MOVING OBJECT INFORMATION FOR LOCATION BASED SERVICES

by

Sabri KANTAR

BS, Computer Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Graduate Program in Computer Engineering

Boğaziçi University

2008

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Assist. Prof. Haluk Bingöl for his guidance, patience and tolerance throughout the preparation of this thesis. He always supported and encouraged me during my studies. His valuable comments and ideas helped me to improve this thesis.

I would like to express my gratitude to Prof. Cem Ersoy and Dr. Gökhan Özdinç for taking part in my thesis committee.

Special thanks to my wife Selime Işık for her cooperation and collaboration during my studies. This thesis has only been possible with her unlimited patience for helping me in finding the best in what I thought and what I did. She supported me with her brilliant ideas.

And finally, I would like to express my gratitude to my parents for their continuous support, encouragement and patience during my academic life and this research. They were my first teachers and helped me to shape my life with their invaluable ideas.

ABSTRACT

MOVING OBJECT INFORMATION FOR LOCATION BASED SERVICES

Delivering personalized services to subscribers is a relatively new challenging area for GSM content providers. One of the most effective ways to personalize mobile services is to make use of the location of the mobile device and to provide location based services.

Example location based services are receiving alerts about a traffic jam, notification of a sale on gas, issuing discount coupons to customers nearing a store and providing information about playing movies in cinemas around. Applications can also be developed that will inform users about other users in close vicinity that have matching profiles, this type of application for example can be used for friend finding. However tracking users is a very expensive operation.

The goal is to decrease the number of location queries sent to GSM operators when tracking users. A solution is presented to track users cost efficiently by predicting future locations of users. The algorithm proposed extracts movement patterns of subscribers and uses these patterns for location prediction. Two types of patterns, stationary and movement, are mined from movement data and two types of predictions, point and path, are used for predicting location.

Real location data of many users provided by a GSM operator are used to evaluate the algorithm. Simulations are also performed with artificially generated log data. Results demonstrated that the number of location queries is reduced significantly while keeping error rate in an acceptable level.

ÖZET

KONUMA DAYALI HİZMETLER İÇİN HAREKETLİ NESNELERİN KONUM BİLGİSİ

Abonelere kişiye özel hizmetler sunmak GSM operatörleri için yeni bir ilgi alanı oluşturmaktadır. Özelleşmiş hizmetler sunmanın en etkili yolu konuma dayalı hizmetler sunmaktır. Müşterinin kullanması muhtemel yollardaki trafik sıkışıklığını haber veren uygulama, müşteriye yakın mağazalardaki indirimlerin haber verilmesi, bir müşterinin yakınlarında aradığı özelliklede başka bir kişinin varlığının bildirilmesi konuma dayalı kişisel hizmetlere örnek verilebilir. Fakat müşterilerin yerlerini takip etmek oldukça pahalı bir işlemdir.

Buradaki amaç GSM operatörüne yapılan konum sorgulama sayısının azaltılmasıdır. Bu tezde hesaplı bir şekilde müşterilerin yerlerini takip edebilmek için çözüm sunulmaktadır. Sunulan çözüm müşterilerin gelecekteki konumlarını tahmit etmeye dayanmaktadır. Konum tahmini müşterilerin hareket alışkanlıklarını çıkartarak yapılmaktadır. Bu tezde iki çeşit hareket alışkanlığı kullanılmaktadır: yolculuk alışkanlığı ve belirli yerlerde uzun süre durma alışkanlığı. Her alışkanlık çeşidi için ayrı tahmin yöntemleri kullanılmaktadır: yol tahmini ve nokta tahmini.

Sunulan çözüm, bir GSM operatörünün sağladığı gerçek konum bilgisi ve yapay olarak üretilen konum bilgisi kullanılarak test edilmiş ve sonuçlar istenilen başarının sağlandığını göstermiştir. Hata payı belirli bir seviyede kalmak kaydıyla konum sorgulama sıklığı önemli ölçüde azaltılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

3D	Three Dimensional
ATM	Asynchronous Transfer Mode
CDR	Call Detail Record
GPS	Global Positioning System
GSM	Global System for Mobile communications
HMM	Hidden Markov Model
MOD	Moving Object Database
MSC	Mobile Switching Center
MT	Mobile Terminal
NAP	Network Access Point
PCN	Personal Communication Network
QoS	Quality of Service
SCM	Spatial Conceptual Map
TA	Timing Advance
TDOA	Time Difference of Arrival

1. INTRODUCTION

Providing location based services to Global System for Mobile communications (GSM) customers is a relatively new issue. Knowing locations of users will enable service providers to provide more personal and location dependent services. Location based services can be categorized into two groups: single user and multi user applications [1]. Single user applications provide services to one user by using only his location information. However multi user services are provided by using location information of many users and they may be used to provide collaboration and interaction between individuals.

Mobile commerce services that have very wide application area for user tracking can be given as an example to single user applications. Mobile users can be informed about a list of restaurants or movies playing in cinemas within certain proximity. Moreover discount coupons can be issued to customers nearing a store. Another application is facilitating calling a taxi by providing location information to the taxi center. Additionally location information of users that make emergency calls like calling an ambulance or police can also be provided to hospitals or police stations. Interest areas and other personal information can be found from moving habits of users. For example if a user goes to a stadium frequently, his favorite team can be learned. If a user goes to a university campus, this user can be identified as a university student or instructor.

In multi user applications location information of many users is used to provide services. Fleet applications are examples for multi user applications. For example a taxi center can send the nearest taxi to customers by knowing the whereabouts of taxies. Another multi user application is matchmaking. Users that are close to each other and that have matching profiles can be informed to each other. For example a user can search for other users that match a set of specific criteria and one of the criteria can be current location. Only users within a specific proximity can be returned as the search result. Social networks can be established by using location information of users.

Furthermore users can be grouped according to their moving habits and different services can be provided to each group. For example information about movies playing at noon can be sent to people that have random movements. On the other hand people that have regular movements are usually working people and they cannot go to a cinema at noon on weekdays.

In order to provide these services, locations of subscribers must be tracked efficiently and accurately. In GSM networks the serving cell that is an indication of the user location is easily available. However this location information is not accurate enough for various location based services [2] because, the longest distance the GSM specification supports in practical use is 35 kilometers which means that there can be very large cells [3]. Furthermore due to shadowing not all the time a Mobile Terminal (MT) is served by the closest cell. As a result the serving cell can be far away from the MT and using location information of the serving cell as a location of the MT can lead to erroneous results. Using GSM phones for localization is discussed in [4] and they conclude that GSM localization systems can achieve median localization accuracies of 5 and 75 meters for indoor and outdoor environments, respectively.

Although a mobile device is served by one cell at a time, several neighboring cells also listen the same mobile device. More accurate location of mobile devices can be calculated by GSM operators using multilateration but, this operation consumes a lot of resources. In multilateration, also known as *hyperbolic positioning* [5], location of mobile devices are found by accurately computing the time difference of arrival (TDOA) of a signal emitted from the mobile host to three or more base stations. There are also other works dealing with positioning in cellular networks [6] and some advance techniques for GSM localization are proposed in [7], [8], [9] and [10].

For tracking facility, GSM operators provide services for querying locations of mobile users, where each query is charged [11].

Since cost is an important issue, keeping track of current locations of mobile device users in an efficient manner is the most crucial problem. The main question is

how frequently query the location of a subscriber and update the database. In this thesis focus is on lowering query frequency while preserving the location accuracy in an acceptable level. If locations of subscribers are queried too frequently this will incur very high query cost. On the other hand if locations of subscribers are queried rarely the precision about the location information will be lost. The objective is to reduce the number of location queries sent to the GSM operator and the constraints are keeping position information accuracy in an acceptable level, using an efficient system and keeping system complexity below a specific level.

It is assumed that many people have regularities in their movements, that is in most of the days they go to same places at the same times. For example working people go to work every weekday at the same times, they stay in their offices until lunch, they go to lunch at the same time and they go to home in the evening at the same time by using the same path. Another group of users that have regular movement patterns are students, the movement profile of each student repeats on a day to day basis. A student goes to the same school at the same times during weekdays. For example on Mondays student X has four lessons and he goes to his school at 11:00 and his lessons finish at 15:00. This cycle repeats for every Monday for this student. Even on weekends most of the students perform regular activities such as working, going to sports courses, etc.

The algorithm proposed in this thesis makes use of these regularities in peoples daily activities. The solution is based on extracting moving patterns of subscribers and using these patterns along with interpolation and extrapolation techniques to predict their locations whenever possible. The main idea is that the next movement strongly depends on the present location, the time of the day and the movement pattern. After extracting repeated movements from a users moving history, if we know that this person is on the way from his workplace to his home there is no need to query his location so often because we know this path from the previous travels. Without querying from the GSM operator, location of the user at the specified time can be predicted using the proposed prediction algorithm. Furthermore if a student has lessons from 11:00 to 15:00 every Monday it is not necessary to query his location during this time period because it can be predicted that this student will stay at the same location until 15:00.

First of all, in order to extract patterns, movement data of subscribers must be collected. When a user subscribes to the system a specified time period is allocated for data collection phase. In this phase the user is queried with maximum frequency in order to collect data. When enough data is collected pattern generation procedure is executed. In this phase firstly, input data is filtered and cleaned because latitude and longitude values can have deviations and noise. Secondly, patterns are extracted from the movement data. There are two types of patterns: movement patterns and stationary patterns. *Movement patterns* represent frequently used paths which are sequences of locations. After extracting frequently used paths time values of points constituting sequences are attached. For example going from home to workplace every weekday in the mornings is a movement pattern. On the other hand if a user stays at the same location for a long time and if this action is repeated for many days, staying at this location between time t1 and t2 represents a *stationary pattern* for that user. For example being at workplace every workday between 9:00 and 18:00 is a stationary pattern.

Once patterns of a mobile user are obtained, future locations of this user can be predicted. Prediction consists of two operations, *point prediction* and *path prediction*. The former gives information about residing at a location during a specified time and the latter gives information about which path among moving patterns is followed. When a prediction is obtained it can be used to answer location queries without sending them to the GSM operator.

The rest of this thesis is organized as follows. In Section 2, related work and the contribution of this paper is presented. Our method for the solution of the problem is proposed in Section 3. In Section 4 simulations and experimental results are presented. We concluded our thesis and mentioned about future work in Section 5.

2. RELATED WORK

The movement pattern mining problem was discussed in [6], [12] and [13]. Work in [6] uses Hidden Markov Model (HMM) based pattern recognition techniques for vehicular position location in cellular networks. HMMs are used for street modeling and for construction of repositories consisting of models for each street element. In this algorithm, by using serving base station and Timing Advance (TA) during a call different repositories are constructed for every TA zone of the radio cell. Moreover for the reported link level observation sequences are extracted and compared with the saved models. Downlink progression is used to identify places and times where the user is not moving. By considering the residence to each TA zone mean velocity can be calculated.

Work in [12] deals with mining moving sequential patterns, where moving sequential patterns are treated as a special case of conventional sequential patterns. First, clustering preprocessing is performed on the data in order to discretize the time attribute, and then Prefix Tree algorithm is proposed to extract moving patterns. The aim of that prefix tree based algorithm is to increase the performance of pattern mining operation. Although time attribute is used in this algorithm, CellIDs are used as location elements and it is not explained how to use these extracted moving patters for location prediction.

Algorithm proposed in [13] extracts user moving patterns from log of call detail records (CDR) of mobile computing systems instead of using moving log that contains each movement of mobile users. The log that contains CDR, reflects only the fragmented moving behaviors of mobile users. Firstly, similar moving sequences are extracted, then CDRs are grouped into several clusters and finally, for each cluster moving functions of user are derived in order to generate user moving patterns. Although there is an algorithm to extract movement patterns from limited log data there is no mechanism to use these patterns for location prediction. Moreover in this paper CellIDs are used too as location elements which is not suitable for our problem.

In [14], [15] and [16] pattern mining is applied for mobility prediction by using CellIDs as locations elements. The algorithm proposed in [14] consists of three phases. Firstly, mining of user mobility patterns is done, and then mobility rules are generated from these patterns. Finally by using these rules next movements of mobile users are predicted with a confidence level. Since location data consists of CellIDs of a cellular network and the topology of the network must be known in order to generate mobility patterns this algorithm is not applicable for location data that consists of latitude and longitude values where a cellular network is not present. Furthermore the proposed algorithm predicts only the next movements and this can reduce the query frequency only 50

In [15] an algorithm for mining of user moving patterns in a mobile computing system and data allocation schemes that use these patterns is proposed. An incremental mining algorithm is used to capture the frequent moving patterns from a movement log data. Moving pattern extraction algorithm uses a roundtrip model [17] where an initial location is selected which is usually home of the mobile user, but a problem arises when the mobile user stays in a different place for a long time (for weeks or months). In this case the algorithm will fail while trying to find a movement pattern for such a mobile user, so an assumption abut the starting point of a moving pattern will not always be valid. Furthermore some other problems in the algorithm proposed in this paper are that there is no time property, only the next location can be predicted and also CellIDs are used as location elements.

Mobility modeling and location prediction for wireless ATM networks is introduced in [16]. Future locations and speeds of mobile users are predicted by a two level prediction algorithm: global prediction and local prediction. Although next possible cells are predicted, no time attribute is used in this work in order to be able to predict future locations of a user given that the user is at location x at time t. The common disadvantage of these algorithms is not using time attribute.

Mobility prediction methods where Global Positioning System (GPS) data is used are presented in [1] and [18]. In [1] a solution to location prediction problem is

proposed by learning significant locations for users. Location information is collected by GPS devices and this location information is used to find important locations for each user. In this algorithm by providing the current location of a specific user the next location is predicted. Also sub locations are used to predict movements inside a location. Because GPS signals cannot reach inside buildings data gaps occur when users are indoor. These data gaps are used to find important locations of users. Predictions are made by using Markov Model. A Markov Model is created for each location with transitions to other locations, which represents the mobility of traveling from one node to another. However this algorithm predicts only the next important location and the path between locations are not known, so path prediction and time attribute are missing in this algorithm.

Work in [18] discusses a location model which is a set of learned destinations expressed by latitude and longitude values. Furthermore routes between the destinations and the time it takes to travel them are also included. The model proposed in this work is based on users' mobility patterns. The model learns important locations for users and predicts future destinations by using route information. Routes also include traveling times which enables estimating time to arrival. Three methods are used for route learning: Bayes Classifier, Hidden Markov Model and Histogram Modeling. After simulations it is found that Histogram Modeling performs the best among the three models. In this model because GPS devices are used, locations can be queried frequently however, this is not possible for our problem where the aim is to reduce number of location queries.

Mobility prediction methods where users moving patterns are not available are presented in [19] and [20]. First, users traveling trajectories and destinations are predicted by means of their preferences, goals and real-world maps. Since the proposed algorithm needs personal information about users, such as their profiles, schedules and preferences along with spatial conceptual maps (SCM) as input, it is not applicable to our problem. An algorithm that depends on maps has limited application area. Such an algorithm can be used only in regions where maps exist. Because there is no possibility to provide such input for all regions, this method is not suitable for a general

and everywhere applicable algorithm.

Architecture of a mobility prediction agent (MPA) is presented in [20]. Instead of using users historical movement data, as in our thesis, this architecture uses knowledge of users preferences, goals and spatial information for mobility prediction. The MPA captures the uncertainty of the users navigation behavior by getting pieces of evidence concerning different groups of candidate future locations. The framework consists of three layers that cooperate to dynamically adapt network behavior to maintain the delivered Quality of Service (QoS) level. The first layer consists of mobility prediction agents (MPA) that utilize the knowledge about the user. The predicted location is then reported to the second layer, where QoS adaptation agents are located to dynamically adapt the network changes. The last layer is designed for getting feedback based on QoS measurements by means of the monitoring agents.

In order to make MT localization more efficient, algorithms for mining and modeling of movement patterns of mobile users in a Personal Communication Network (PCN) are proposed in [21], [22], [23], [24], [25], [26], and [27]. Location prediction is performed for MT localization when a call arrives. The aim in these works is reducing paging and location update costs. A compression technique [28] is used for capturing the movement patterns to make mobility management in cellular networks more efficient in [22] and [23].

In [21] mobility model of users is extracted and this model is downloaded to the mobile host for real time verification and next cell is predicted by using an artificial neural network. The drawback of the algorithm proposed is that the network topology must be known and the other problem is that mobile device is involved in prediction process which is not possible in our problem domain.

Algorithm presented in [24] proposes a user pattern learning strategy that reduces the signaling cost of a location update. The algorithm classifies the users in three different categories according to their daily routines: users who have a very high probability of being accurately predicted, users who have a certain likelihood of being

where the system expects them to be, and users whose next position is unpredictable. A user pattern learning strategy is used to generate a list of location areas for each user where the user is most likely to be located within a given time interval. When a call arrives for a MT, each location within the list is paged sequentially until the MT is found. No location update takes place when a user moves between locations within the list. Artificial neural network with three layers is used to learn users movement patterns by using back propagation training algorithm. The obtained list is stored in Mobile Switching Center (MSC) and in users MTs. Disadvantage of this method is the need to store information in mobile terminals.

In the following methods also MT is involved in the location prediction process. Focus of paper in [25] is on using a subscribers mobility patterns to predict a set of possible locations when a call arrives. The aim is to perform call delivery by paging only the set of predicted locations. The prediction algorithm is based on time specific movement history. In this scheme mobile terminal is responsible for predicting, registering and maintaining its own mobility pattern. A subscriber mobility database that contains mobility patterns is maintained by the mobile terminal. A secondary set of locations is given as input during subscription time along with a route map of roads in the neighborhood of a location area and this information is stored at the base station of home location. The proposed algorithm extends algorithms [27] and [29] by introducing a policy to handle unexpected deviations from registered movement patterns. However this approach also requires modifications in mobile terminals and it requires external inputs such as maps which is not feasible and not suitable to our requirements.

Work in [26] proposes a mobile tracking scheme that uses the predictability of mobility patterns of users in wireless PCS networks. Locations of mobile users are predicted by the network by using location and velocity information gathered from these mobile users. When a call is made the network pages only the predicted location. The same prediction is made also by the mobile and if actual location and the predicted location exceed a specified threshold value the mobile reports its location. In order to perform mobility prediction Gauss Markov mobility model is used. In the proposed algorithm the future location of a mobile is predicted based on the probability density

function of the mobiles location.

In [27] a user mobility pattern scheme is introduced for location update and paging in wireless systems. In this solution mobile terminals maintain their history data in database. User mobility pattern is derived from user mobility history during a location update and this derived pattern is registered to the network. If MT detects that the registered pattern is not followed it performs a location update, otherwise no location update is performed. In order to page a user, a sequential search is performed. Disadvantage of this method is using MT for deciding whether to perform location update or not. Furthermore only mobility pattern mining technique is proposed in this work and there is no algorithm for location prediction.

Disadvantage of these methods is that mobile device is involved in prediction process which is not possible in our problem domain.

Movement pattern mining and location prediction techniques are applied for maintaining Movement Object Databases (MOD) more efficiently in [30], [31] and [32]. In [30] safe region for moving objects are determined by movement patterns, and if these objects remain in their safe region location update information is not sent. The next progress of an object is calculated by taking the average of previous movements, where a window value is used to determine the number of previous movements used for calculation. Also weighted average technique is used to give more weight to recent movements in the history data while calculating the average. We also used the weighted average technique in location prediction in order to give more weight to recent data.

Work in [31] deals with the analysis, pre processing, modeling and storage techniques of traffic data in a MOD. Moreover data extraction techniques are used for prediction of difficult situations such as traffic congestions in a traffic management system. Fundamental concepts of the movement of a vehicle are analyzed and its semantics and properties in terms of a conceptual model are stored in a database. Stored data contains vehicles, routes, trajectories and relations among them. Moving object database consists of spatial data, non spatial data and trajectories. The concentration

in this work is handling trajectory information and to extract further knowledge from this information. Along with known data extraction techniques that are characterization, clustering and association, Spatial Mining Language is introduced to support these functions. One of the key points is to extract and store semantics of movements (speed, heading, size of the moving object, traveled distance and traveled time). Extraction of further knowledge from the trajectory information is adapted to our problem domain and it is used to extract movement behaviors and patterns.

The paper called Accuracy and Resource Consumption in Tracking and Location Prediction [32] deals with maintaining current locations of a large number of moving objects in order to be able to provide location based services. An alternative to distance update policy is proposed in order to minimize location updates while keeping the accuracy in an acceptable level. The proposed algorithm pertains to motion on the road network given by a map, and it improves the performance of tracking by location prediction. Following a location update it is predicted that the object will continue moving on the same street. If there is a deviation from the predicted location the moving object or the network updates the database. In this scheme it is required to provide a map of the geographical region.

Algorithms in [33], [34], [35] and [36] use location prediction for resource pre allocation. Instead of trying to predict the next physical location, work in [33] proposes a solution to predict the base station to which the user will next connect. Not only user behavioral patterns are incorporated in the mobility prediction algorithm, but also wireless link characteristics and the handover decision making mechanisms are used. The probability distribution of all possible next moves from the mobility history is used to derive a prediction. If the first predicted cell does not contain a probability higher than the Prediction Confidence Ration (PCR), extra cells are added to the group of cells in which resources are reserved in advance. This operation continues until the sum of their probabilities exceeds the predefined PCR value. The aim is to avoid predicting the random components of user movements.

Algorithm presented in [34] proposes an adaptive user mobility prediction method that uses only a subset of cells around the user for reservation and configuration procedure in wireless networks to provide QoS. IDs of base stations to which a mobile phone was connected during movements are logged and it is seen that there are random components in traces due to Ping Pong effect between adjacent base stations or some temporary handovers to other base stations. For prediction, the paper uses current location, current direction, current time, patterns and conditional probability of future direction.

A mobile motion prediction algorithm for resource pre allocation is also proposed in [35]. Prediction in this algorithm is based on movement history of users. There are regular and random components in movements that can be matched with patterns or can be simulated by the Markov chain model. The contribution of this paper is introduction of mobile floating agents for resource pre assignment, data pre fetching and service pre connection. Prediction accuracy of this scheme is very high with regular movements however random components decrease the prediction accuracy significantly. Because these algorithms predict a set of possible next cells it is not applicable for predicting a future path. There is noise also in our log data possibly due to Ping Pong effect between adjacent base stations. So, filtering data techniques mentioned in these papers are used in our algorithm.

Work in [36] proposes a profile based next cell prediction algorithm for providing early reservation and adaptation in resource management. Prediction scheme in this paper is based on movement history of users and classification of locations. Mobility is divided into four categories: purely random, mostly random, purely deterministic and mostly deterministic. Locations considered in the paper are office, corridor and common room. However disadvantage is that classification of locations requires advance knowledge about the geographical region and this makes the algorithm applicable to only some regions.

Algorithm in [37] performs mobility prediction in cellular networks by using observed traffic patterns. In this paper a scheme is proposed that uses user mobility

information to optimize the resource reservation in cellular networks. The proposed handover strategy consists of two parts: mobility prediction and advance allocation. Mobility prediction is a scheme that predicts the most probable target cells depending on the users current state. Advance allocation is an allocation strategy that determines the number of handover channels required given the system state and user position. In order to find handover traffic flowing into a cell, mobility of individual users is predicted and the contributions from all the users in the neighboring cells are combined. Prediction is done by using position, speed and direction of movements of users.

In [38] classification is exploited in order to predict next movement of a mobile user. In this paper a context model, based on classification, which deals with location prediction of moving users is proposed. Location prediction is used for resource reservation. The model produced predicts the next movement of a mobile user with certain moving profile and history of movements where spatial and temporal information is included in users profiles. Decision tree induction algorithm is used for classification in the proposed scheme. Current location is the basic context attribute for classification and distribution for cell residence times [39] in these locations is assumed to be the generalized Gamma distribution. Classification rules that represent movement trajectories are produced with the decision tree classifier. Again the disadvantage of this method is that it only predicts next locations of users but not the whole path.

Works in [40] and [41] use vehicle traffic only for location prediction. A prediction algorithm by using highway traffic is proposed in the paper called Shadow Cluster Concept [40]. Prediction algorithm proposed in this paper for resource reservation is based on the users movement history on highways. Highway traffic with various constant speeds traveling in forward and backward directions is considered.

Another work that aims to provide a solution to location management problem is presented in [41]. In this paper prediction is achieved by using movement history of users. Mobility model used in this paper is derived from statistical analysis of actual call traffic traces, vehicle and aero plane traffic data, and government transportation surveys. Movements considered in this paper are random walks and repetitive round

trips. Although bandwidth and memory requirement are increased, most of the calls are served by local database lookups. Because users movement data in our domain consists of both vehicular and pedestrian traffic, these algorithms are not applicable to our log data.

In [42] a class of mobile motion prediction algorithms for supporting global mobile data access is described. The algorithm suggested, called predictive mobility management, predicts the next location of a mobile according to users movement history. From the point of service providers it is very important to give the right service to the right user, meaning service user matching. In order to be able to provide the services with minimum querying and acceptable error rate more intelligent mobile aware applications and efficient routing management should be developed. Different from our proposal random movements and regular movements are considered as two independent predictions where mobile circle and mobile track models are used to predict the regular movement while Markov Model is used for random movement prediction. Also characteristics of mobile user are kept in database in order to be able to use them in predictions, which means more data should be learned in each query and more data should be kept.

In [43] a data mining algorithm is proposed for discovering users behavior patterns in mobile web systems. Three categories of users behavior prediction are provided: location, services and location associated with services. These three categories of prediction are supported by N-gram Model which is a variant of Markov Model. For location prediction the proposed data mining method is called SMAP-Mine that can discover mobile users sequential movement patterns associated with requested services. In order to be able to associate location prediction to a service two types of logs are collected: movement logs and service request logs. Then for each user data integration is performed and services are associated to locations. By using the obtained integrated data, data mining phase is completed where the output of the mining phase is used for predictions. This proposal differs from our thesis in that it uses service logs along with movement logs, which means more logs and more storage. Moreover the algorithms described in that paper are suitable only for web systems.

Work in [44] proposes an algorithm to provide user mobility prediction in hybrid and ad hoc wireless networks. According to this paper past history is not valuable for mobility prediction in ad hoc networks. They give examples about military environments, emergency search and rescue operations where past history is not possible. They extended the algorithm in [45] to hybrid and purely ad hoc networks by introducing the sectorized cluster sector concept. By using this concept the area required for tracking is reduced. Furthermore cluster and sector numbering scheme is proposed for the prediction of user movements. However instead of emergency environments the aim in our work is to extract regularities in peoples daily lives.

Mobility prediction is used in [46] in order to enhance unicast and multicast routing protocols in wireless networks. In this algorithm GPS position information is piggybacked on data packets during a connection and this information is used to estimate the expiration time of the link between two adjacent nodes by predicting future locations of the nodes. The goal is to react before the connection breaks. According to coordinates, speeds and moving direction of two neighbors link expiration time is predicted. Furthermore route expiration time which is the minimum of the link expiration times along the route can be predicted. Routes that stay connected longest are chosen by utilizing the mobility prediction. However because speed and direction are used in prediction, performance of the proposed algorithm decreases dramatically on sudden changes in direction and speed.

Algorithm in [47] uses network elements for extracting the aggregate movement data of many users and to predict the future behaviors of these users. This work presents a neural network prediction system that captures some of patterns exhibited by users movements in a wireless environment. Aggregated data storage method that is used for mobility prediction and path recognition in this paper argue that user mobility is subject to geographical constraints at the place of each Network Access Point (NAP) and thus all users will exhibit similar behavior at a given NAP. Aggregated data storage method differs from per user based storage in that it considers the regularity of network, whereas user based storage deals with the regularity of user mobility. This algorithm seems to be more useful in small geographical areas, since it needs to be aware of the

geographical constraints and predict users locations using these constraints. Moreover in the domain of our problem there is no facility to access network elements in order to gather extra information.

In [48] a comparison of algorithms for path prediction is made to increase the handoff trigger accuracy. The prediction is based on historical data observed from the movements across multiple cells. The main aim of this paper is to find out how accurate the predictions of users future serving base stations can be done and to find out the importance and impact of the input data. The input data for such algorithms, that are trying to detect movement regularities, are historical and simulation data. Several algorithms for pattern recognition are compared: Decision Trees (DT) [49], instance based (IB) [50] nearest neighbor algorithms and support vector machines (SVM) [51] comparable to neural networks [52]. All these algorithms being compared in that paper offer one possibility to detect repeating behaviors in users movements. It has been seen that all mentioned three algorithms have similar behaviors depending on the maximum length of available path sequences. It is demonstrated that a combination of more than one algorithm reduces the overall rate of erroneous predictions. There are also proposed algorithms for generating artificial test data. In [53] a Realistic Mobility Pattern Generator (RMPG) is introduced. RMPG, a platform for the simulation of mobility functions, is based on the assumption that real humans exhibit considerable spatial and temporal regularity in their moves. The overall architecture of RMPG consists of two parts: area map creation and java simulation platform. In the map creation part movement patterns are produced by tool provided in the Network Simulator 2 (NS2) [54]. This part differs from the generator introduced in our thesis in that, we generated points, whereas RMPG produces movement patterns with NS2 provided tool. After mobility map is constructed there comes the second part of RMPG, simulation of movement of users in the given area. This is the part that our thesis extends from. Each user may have a different action profile (worker, student, etc.), so can exhibit a quite different movement behavior from others. Different action profiles can be defined for distinct group of users. Moreover the day of the week, random events and the action profile are used for determining the users daily action, as in the case in our thesis.

In [55], which is a short version of [56], a new moving objects generator for spatiotemporal data is presented. This generator combines a real network with user-defined properties of the resulting dataset. Different from our thesis, in this paper the generation of spatiotemporal data is based on a network. Simulation of the motion of a huge number of moving objects is examined by taking into account the user-defined parameters, meaning traffic jams, speed, speed influenced by other objects, weather conditions, etc. Furthermore, time modeling, speed and choosing a starting node concepts are used in our thesis too. Because the road network must be provided to this generator we cannot use it to generate data for our algorithm.

To sum up none of the algorithms mentioned above can provide solution to the problem of lowering the number of location queries sent to the GSM operator. Contribution of this thesis is to provide a solution to this problem by combining useful aspects of the algorithms mentioned above; such as generating patterns from latitude and longitude values and attaching time attribute to these patters. This thesis also introduces separating patterns into two groups, merging similar sequences technique, methods for comparing point sequences and methods for predicting future locations of users.

3. PATTERN MINING AND MOBILITY PREDICTION

In order to decrease the number of location information requests to the GSM operator, firstly patterns of users are mined from their historical data and then these patterns are used for prediction of future locations. As a result, if a prediction can be made about future locations of a user, instead of sending location information requests to the operator, location queries can be responded by using this prediction. The solution proposed in this thesis consists of two parts, *pattern mining part* and *prediction part*.

3.1. Pattern Mining

In this section patterns are extracted from the movement log data which is provided by the GSM operator and it consists of user ID, time stamp, latitude and longitude information. In Table 3.1 example log data of four different users is shown. Here latitude and longitude values are floating point values with very high precision. However the actual resolution of the provided data is about 47 meters. This value is found by calculating the minimum distance between two consecutive and different location readings of a user. Furthermore query period in the provided log data is about 10 minutes.

There are two types of patterns a user can have, movement patterns and stationary patterns. *Movement patterns* represent frequently used paths by individual users and these patterns are sequences of points. For example path from home to schools or path from home to workplace can be a movement pattern for a user. *Stationary patterns* however represent places that are personally important to users. Stationary patterns contain information about when this user arrives at this location and when he leaves the location. These two types of patterns are mined separately but, before mining operation noise is removed from the movement log data.

Table 3.1. Example movement log data

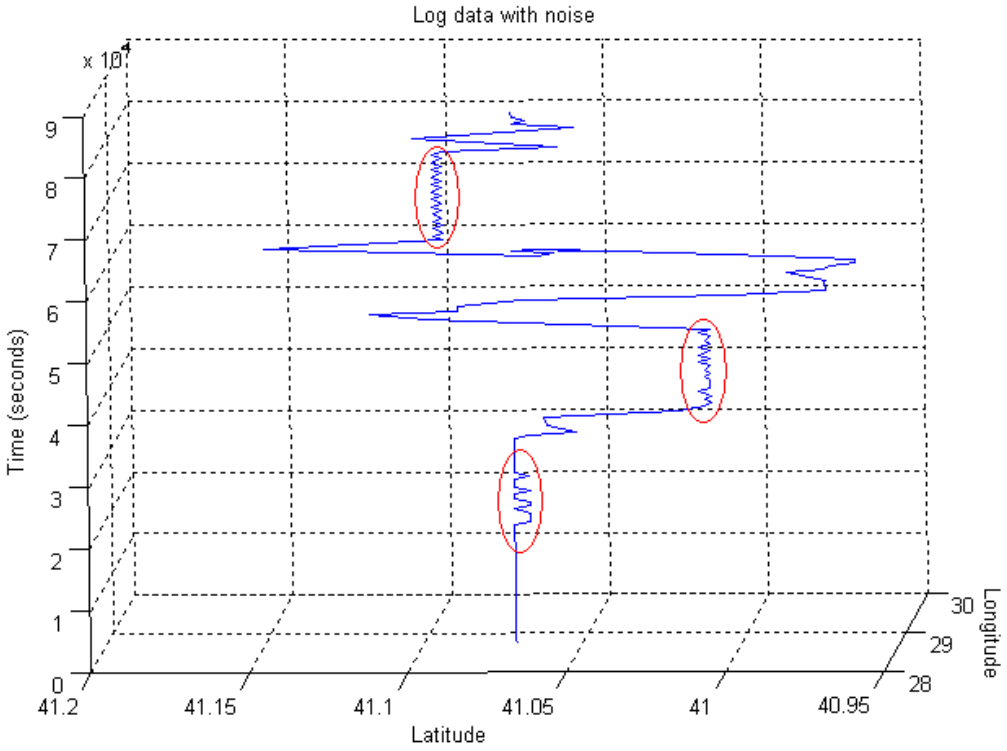
User ID	Time Stamp	Latitude	Longitude
1	01/10/2007 17:46:12	41.0394444444444	28.9758333333333
3	01/10/2007 17:46:43	41.1122222222222	29.0225
4	01/10/2007 17:47:01	41.0788888888889	29.0094444444444
5	01/10/2007 17:47:18	40.9122222222222	29.1758333333333
1	01/10/2007 17:56:10	41.0394444444444	28.9758333333333
2	01/10/2007 17:56:13	41.0772222222222	29.0272222222222
3	01/10/2007 17:56:17	41.1122222222222	29.0225
4	01/10/2007 17:56:20	41.0788888888889	29.0094444444444
5	01/10/2007 17:56:24	40.9122222222222	29.1758333333333

3.1.1. Removing Noise

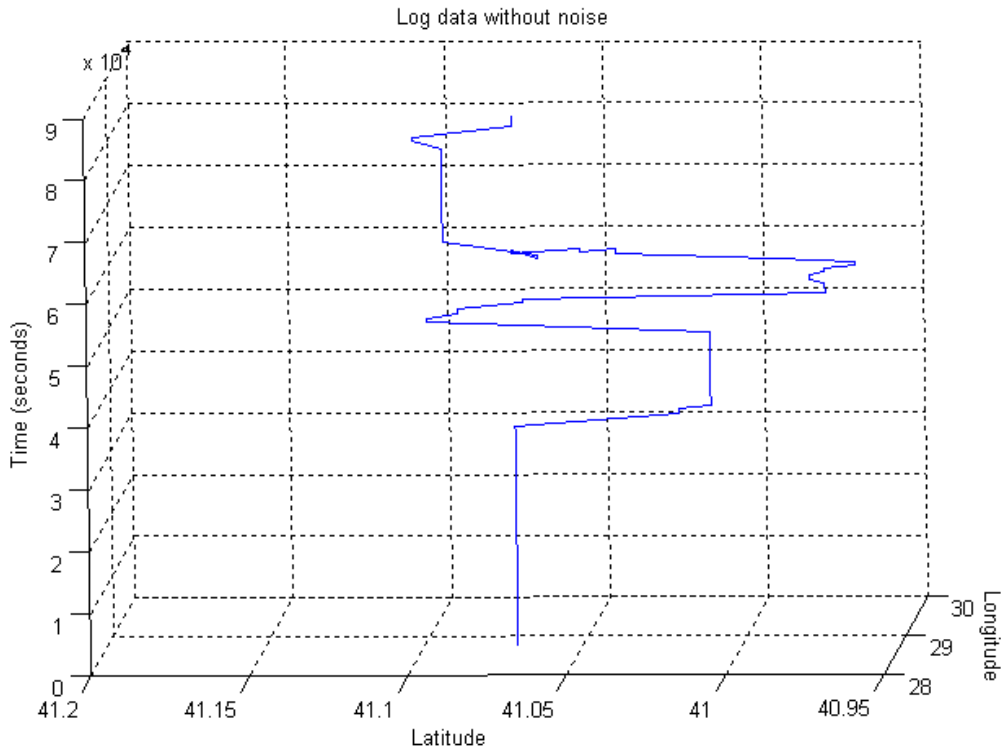
In order to obtain better results, noise in the log data must be removed. The simplest noise removal operation is rounding latitude and longitude values. However even after rounding, some noise exists in the data. In Figure 3.1(a) one day log data of one user is shown on a 3D graph where x and y axes show latitude and longitude values and z axis shows number of seconds elapsed from the beginning of the day. Vertical lines on the graph represent residing at a fixed location and horizontal ones represent fast moving. As it can be seen from the graph although rounded data is used, noise exists when the user stays at a fixed location. Because of this noise there are problems with determining important locations for users. Important locations are locations where users spend time more than the threshold value t_{time}. In order to be more precise, the important location definition is given in Definition 1.

Definition 1. Threshold time for determining important locations is t_{time}. If a user stays in a location more than t_{time}, then this location is called an *important location* for that user.

For example home, workplace, school, fitness club, etc. can be considered as important locations. In Figure 3.2(a) important locations of one user are shown. In



(a)



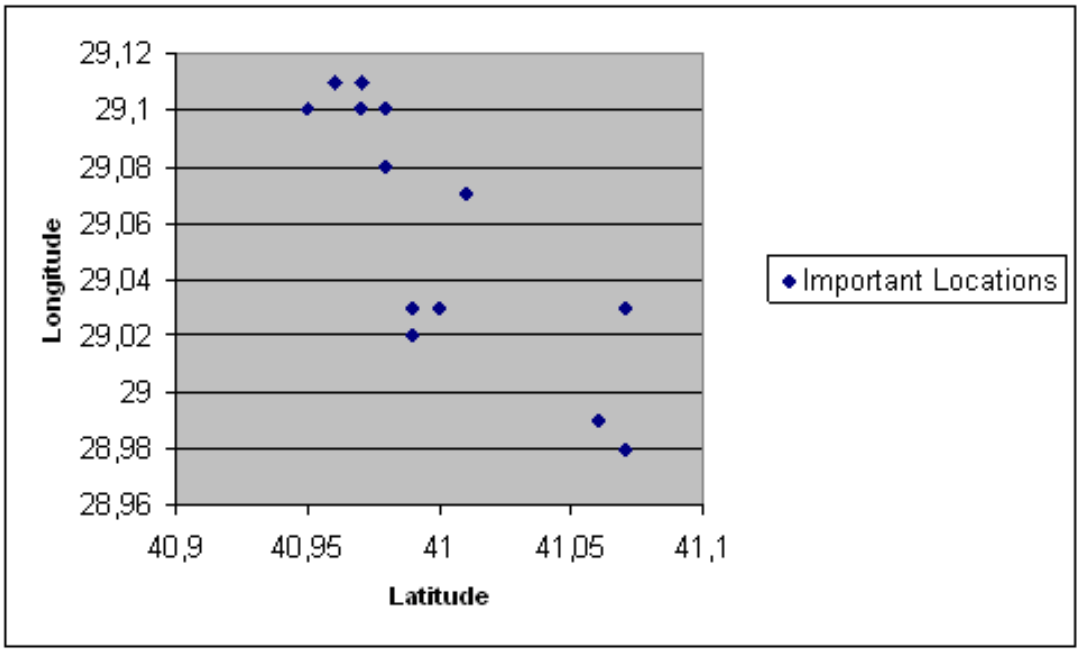
(b)

Figure 3.1. Visualization of one day log data of one user, (a) Before noise reduction, (b) After noise reduction

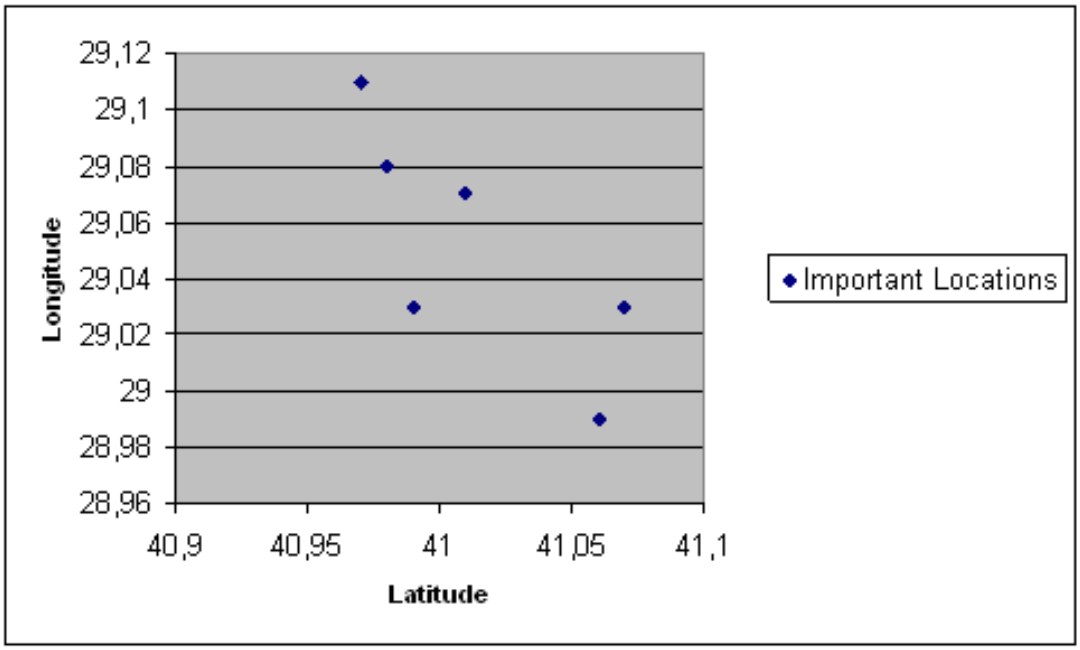
this figure it can be seen that there are many important locations very close to each other which represent the same location. Dense regions are regions where there are many important locations close to each other. It is necessary to align all points in a dense region to a single point in order to represent an important location with same coordinates. These dense regions are assumed to be circle shaped and aligning all points in a dense region to the center of the circle is the ideal case. In order to solve this problem the following alignment operation is performed. Firstly, points that are to be center points of the dense regions are obtained from the log data. By scanning the log data, points at which a user resided for a long time are determined and these points are chosen as center points. Time spent at a point is found by calculating the time span between entering time to the point and leaving time from the point.

Next, for each center point a circular region with radius r is formed as shown in Figure 3.3. After that, the log data of the user is scanned again and each point read is compared with all circles obtained above. If a point falls inside one of the circles, the coordinates of the point being compared are replaced with the coordinates of the matched circle center as shown in Figure 3.3. If a matching circle cannot be found coordinate values of the point being compared are only rounded. Details of the rounding operation are explained Section 3.1.2.

Value of radius r of the formed circles is adaptive; there is a table that keeps different r values for different geographic regions. For example, small value of r can be used in urban areas for higher precision where larger r values can be used on rural areas. Chosen r value determines the maximum location error in distance. For example if r=500m is chosen location of a user can be aligned to another location that is 500m away, in other words r defines the maximum deviation from the actual location. In downtown areas where there are many shops, cinemas, theaters etc. close to each other, higher precision is needed so, small r values must be used for this type of regions. However assigning values to r less than resolution of the log data is useless and assigning too large values will decrease the location information accuracy. Additionally cell sizes in a region can be used as a reference for choosing r value for this region. Using small r values for regions with small cell sizes, and using large r values for regions



(a)



(b)

Figure 3.2. Important locations of a user found before filtering and after filtering phases

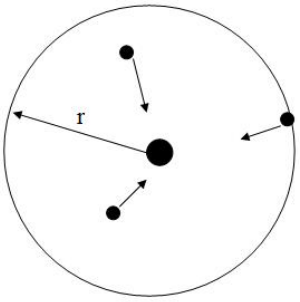


Figure 3.3. Point alignment operation

with large cell sizes will make the algorithm applicable to different geographic areas.

Figure 3.2(b) shows locations found by using filtered data of the same user as in Figure 3.2(a) and the Figure 3.1(b) shows the log data obtained after this noise removal phase.

3.1.2. Rounding

Rounding operation is performed by changing the scale of the coordinate values. A coordinate is rounded in such a way that geographical distance between the rounded value and the original value will not exceed a specified maximum distance. The geographical distance mentioned in this thesis is the shortest distance over the earths surface. It is found by using "Haversine" [57] formula that calculates great circle distances between the two points on a sphere.

There is a table that keeps different maximum distances for different geographic regions. The aim is to be able to adjust the rounding level for different regions. For example small values for maximum distance can be used in urban areas for higher precision where, by using large values for maximum distance, coordinates on rural areas can be rounded more. The maximum distance value can be determined according to location information precision requirements.

3.1.3. Mining Stationary Patterns

Stationary patterns represent residing at important locations. For example a user goes to work at 9:00 in the morning and stays there until 18:00 and this user repeats this behavior in all weekdays. From this repeated behavior it can be inferred that if the user is at work this user will stay there until 18:00. So, these types of repeated behaviors are named stationary patterns. If it can be predicted with a certain confidence that a mobile user will stay at his current location until a specified time there is no need to send a location query to the GSM operator.

Stationary patterns are mined by using important locations and by using residing times at these locations. Stationary patterns contain location information that is a point, arriving time to this point and leaving time from this point. Consequently crucial point is to determine these locations that are important for users. For example for a student, his home, school, sports club are the most important locations. Important locations are found by considering time spent at all points. If a user spends more time than a threshold value t_{time} at a point, this point is identified as an important location and is added to the stationary patterns list of this user. For example suppose a movement log data for a user to be: "a-a-a-a-a-b-t-t-m-r-k-k-k-k" where letters represents points that the user passed during a movement. From this sequence it is understood that this user resided at point "a" for some time, then traveled through points "b", "t", "m", "r", "k" and resided at point "k" for some time. If this user stays at points "a" and "k" more than threshold value t_{time} then, "a" and "k" are important locations for that user. This user resided also at location "t" but because time spent in "t" is less than t_{time} value point "t" is not considered as an important location. In this example time spent in point "a" is found by calculating time difference between time attribute of the first point "a" and time attribute of the point "b". Because time spent in "a" is more than t_{time} value point "a" is identified as an important location. By incorporating this important location with arrival and leaving times, a stationary pattern is obtained. *Arriving time* for location "a" is time attribute of the first "a" and *leaving time* from this location is time attribute of the point "b".

Table 3.2. Example data of stationary patterns

User ID	Latitude	Longitude	Arriv.Time	Arriv.Day	Leav.Time	Leav.Day
4	40.97	29.05	38219	4	74226	4
4	41.025	29.02	48411	4	66417	4
2	40.97	29.05	68297	4	44369	5

Definition 2. *Time attribute* of a point is the time of visiting this point. Furthermore this time value is the elapsed time from the beginning of the day (number of seconds).

All log data is scanned in this manner in order to find important locations of users. It is possible that a user arrives a location, stays there and leave it on the next day as in the case of going home in the evening and leaving in the next morning. Arriving and leaving day values are also stored along with other attributes to be able to resolve this. Attributes of a stationary pattern are shown below.

Stationary pattern:

- *latitude*
- *longitude*
- *arriving_time*
- *arriving_day_of_week*
- *leaving_time*
- *leaving_day_of_week*

Example data of stationary patterns is shown in Table 3.2.

3.1.4. Mining Movement Patterns

Movement patterns are frequently used paths by users and they can be extracted basically by finding frequently occurring movement sequences in the log data. A movement sequence is a time ordered sequence of points visited during traveling between

two important locations.

Definition 3. Let p_t be the point a user is in at time t. Then sequence of points p_{t1}, p_{t2}, p_{t3}, ..., p_{tn} is called a movement sequence where $t1 > t2 > t3 > \ldots > tn$.

Consider a user going from home to work. Let C = $< c_1, c_2, c_3, c_4 >$, B = $< b_1, b_2, b_3, b_4 >$ and A = $< a_1, a_2, a_3, a_4 >$ be the movement sequences he makes on Monday, Tuesday and Wednesday respectively. If A = B = C then these three movement sequences will be equal and the movement sequence A = $< a_1, a_2, a_3, a_4 >$ will be found three times in the log data.

Definition 4. Assume that we have two movement sequences A = $< a_1, a_2, \ldots, a_n >$ and B = $< b_1, b_2, \ldots, b_n >$. A = B, iff $a_k = b_k$, for all k, where $1 \leq k \leq n$.

For example if our occurrence threshold to consider a movement sequence as a movement pattern ($t_{occurence}$) is 3, these movement sequences will be identified as a movement pattern. However due to different traffic conditions in different days this user can be in different points at the same times. It is not always possible A = B = C. So, the same path is represented by different point sequences on different days. Although these three movement sequences represent the path from home to workplace as shown in Figure 3.4 they are different sequences because they consist of different points. As a result, we cannot come up with a movement pattern by using these sequences. There is a need to extract the information that the user used the same path in movement sequences A, B and C even if the points in these movement sequences are all different.

Solution proposed in this thesis is to merge these movement sequences. Daily point sequences of the same week are merged with each other by using the merging rule shown in Figure 3.5, where sequence B is merged into sequence A.

Definition 5. *Daily point sequence* is a movement sequence representing a user's location information during one day period. Daily point sequences start from 00:00

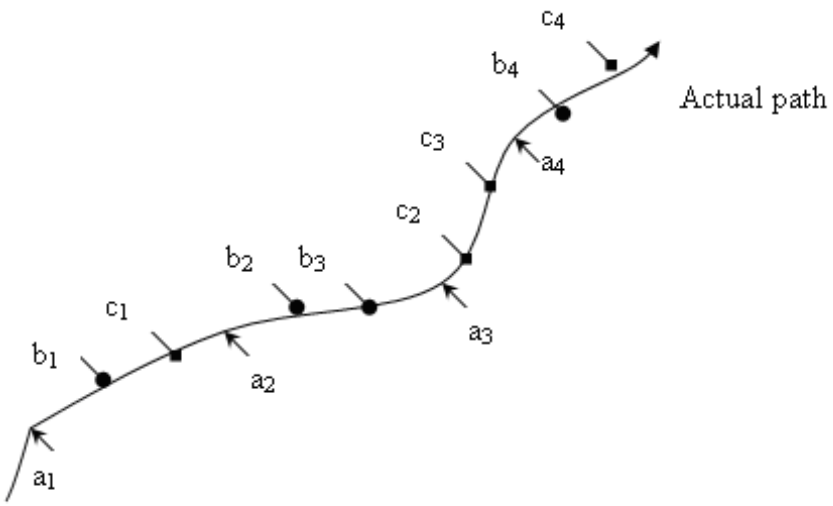


Figure 3.4. Different point sequences on the same path

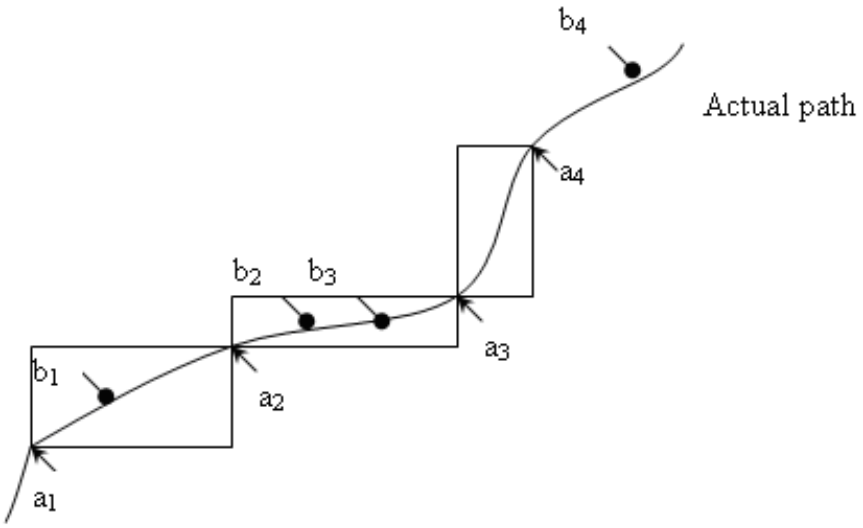


Figure 3.5. Inserting sequence B into sequence A

o'clock and end in 24:00 o'clock.

Virtual rectangles as shown in Figure 3.5 are formed between two points that are used as merging condition. Merging condition is explained below.

Suppose P1, P2 and P3 are points on the earth.

if ((P1.latitude < P2.latitude < P3.latitude) ∨ (P3.latitude < P2.latitude < P1.latitude)) ∧ ((P1.longitude < P2.longitude < P3.longitude) ∨ (P3.longitude < P2.longitude < P1.longitude)) **then**

P2 can be inserted between P1 and P3

end if

In our example case if point b_1 falls inside the rectangle formed by points a_1 and a_2, point b_1 is inserted between a_1 and a_2. If not, then point b_1 is checked for the rectangle formed by a_2 and a_3 and so on. The resulting sequence after this step is $< a_1, b_1, a_2, a_3, a_4 >$. This continues for all points in sequence B and the result below is obtained. $< a_1, b_1, a_2, b_2, b_3, a_3, a_4 >$ Next, sequence C is merged into the resulting sequence as shown in Figure 3.6.

$< a_1, b_1, c_1, a_2, b_2, b_3, a_3, c_2, c_3, a_4 >$ is the resulting sequence for A. Now the operation with sequence A is finished. The same procedure is applied to sequence B, where sequence A is merged into sequence B and then sequence C is merged into the resulting sequence. At the end of the merging process for the three movement sequences, obtained movement sequences are shown below.

$< a_1, a_2, a_3, a_4 > \Rightarrow a_1, b_1, \mid c_1, a_2, b_2, b_3, a_3, c_2, c_3, a_4 \mid$

$< b_1, b_2, b_3, b_4 > \Rightarrow \quad b_1, \mid c_1, a_2, b_2, b_3, a_3, c_2, c_3, a_4 \mid, b_4$

$< c_1, c_2, c_3, c_4 > \Rightarrow \quad \mid c_1, a_2, b_2, b_3, a_3, c_2, c_3, a_4 \mid, b_4, c_4$

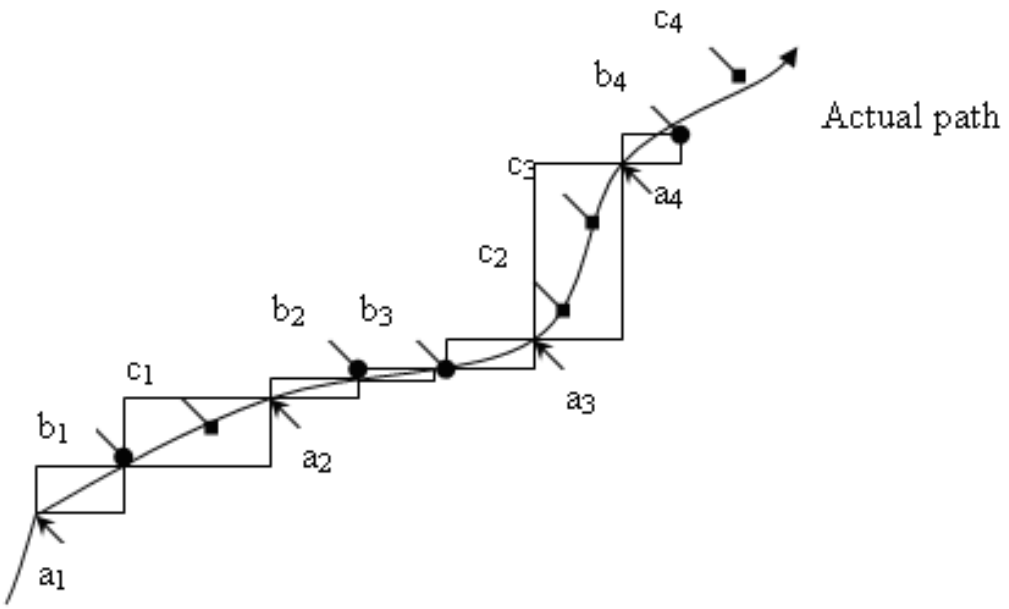


Figure 3.6. Inserting sequence C into the resulting sequence

Three totally different sequences are converted into three very similar moving sequences that have parts in common. As it can be seen from the example above subsequences between pipes are the same. Because our $t_{occurence}$ is 3, sequence $< c_1, a_2, b_2, b_3, a_3, c_2, c_3, a_4 >$ can be identified as a movement pattern.

In this merging algorithm, rectangles are getting smaller as more point sequences are inserted, so merging conditions for these sequences get stricter. For example, smaller rectangles are used as merging condition when inserting point sequence C in Figure 3.6. Because rectangles will be very small after a number of iterations, inserting a sequence will be almost impossible. On the other hand this is unavoidable because coordinate order of points must be preserved in the resulting sequence. Since this merging operation is performed for all days in a week, at most six sequences are inserted into each sequence. There are $n-1$ iterations in a merging operation where n is the number sequences so, totally $n*(n-1)$ merging operations are performed. In our case daily point sequences of the same week are merged with each other so, $n = 7$ and there are $7 * 6 = 42$ merging operations for each week. Alternatively instead of merging all

days in a week, only same weekdays of different weeks can be merged with each other. For example, Mondays of subsequent weeks can be merged with each other. However, after simulations with both methods, the better results (smaller query rate and error rate) are obtained with the first approach.

Since merging order affects the result of the operation descending date order is used for determining the merging order. For instance merging B and then merging C into A can give a different sequence as a result than merging C and then merging B into A. Each daily sequence has a *day* attribute which is a day of week number. Day attribute is 1 for Monday, 2 for Tuesday, and continues in the same order where day attribute is 7 for Sunday. In our example, day attribute for daily sequence A is 3 because this daily sequence is realized on Wednesday, for B is 2 (Tuesday) and finally for C is 1 (Monday). Suppose that daily sequences S1, S2, S3, S4 are realized on Thursday (4), Wednesday (3), Tuesday (2) and Monday (1) respectively. As a result S1.day = 4, S2.day = 3, S3.day = 2 and S4.day = 1. These sequences are ordered according to their day attributes (weekdays) as follows: S1.day>S2.day>S3.day>S4.day. Merging order for S1 will be S2 then S3 and then S4 which means that firstly S2 will be merged into S1, then S3 will be merged into S1 and finally S4 will be merged into S1. Similarly merging order for S2 will be S1 then S3 and then S4, for S3 will be S1 then S2 and then S4 and for S4 will be S1 then S2 and then S3. The reason of using this order is to give more importance to the recent daily sequences.

On the other hand, in some cases instead of enhancing the sequences, merging operation can introduce noise to the initial sequences. This generally occurs when two paths are intersecting only at few points as shown in Figure 3.7. If sequence B is merged into sequence A, sequence A will become $A^1 = < a_1, a_2, b_2, a_3 >$ and this will change the path represented by the sequence A significantly (dashed lines in the figure). As a solution to this problem $t_{matching}$ parameter is introduced according to which this merging decision can be made. In this algorithm merging operation is performed but, the resulting sequence is stored if and only if number of merged points is equal to or more than $t_{matching}$ value. In the case where the number of matched points is less than $t_{matching}$ value the resulting sequence is discarded and the original ones are preserved.

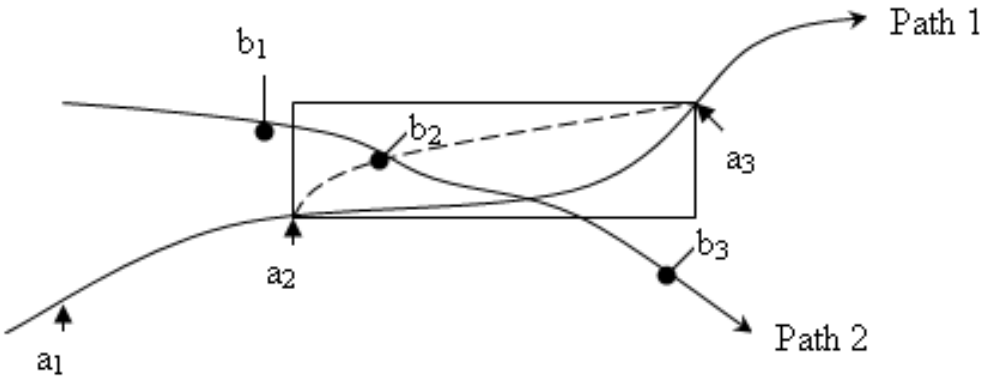


Figure 3.7. Two different and intersecting paths

if (number of points in A^1 – number of points in A) $\geq t_{matching}$ **then**
  store A^1
else
  store A
end if

For instance, suppose $t_{matching}$ value is 3. In the example case shown in Figure 3.7 number of points in A = 3 and number of points in $A^1 = 4$ so, Boolean expression 4 – 3 $\geq$ 3 is false. Because only one point which is "b_2" is merged in daily sequence A, merging operation of B into A is not performed.

However, in some cases, very little or no common parts can be obtained after the merging process. Especially when there are sharp direction changes on the path, the merging algorithm performs poorly. In Figure 3.8, merging of two movement sequences A = $< a_1, a_2, a_3, a_4 >$ and B = $< b_1, b_2, b_3 >$ is shown. Rectangles with solid lines are rectangles generated by points in sequence A and rectangles with dashed lines are

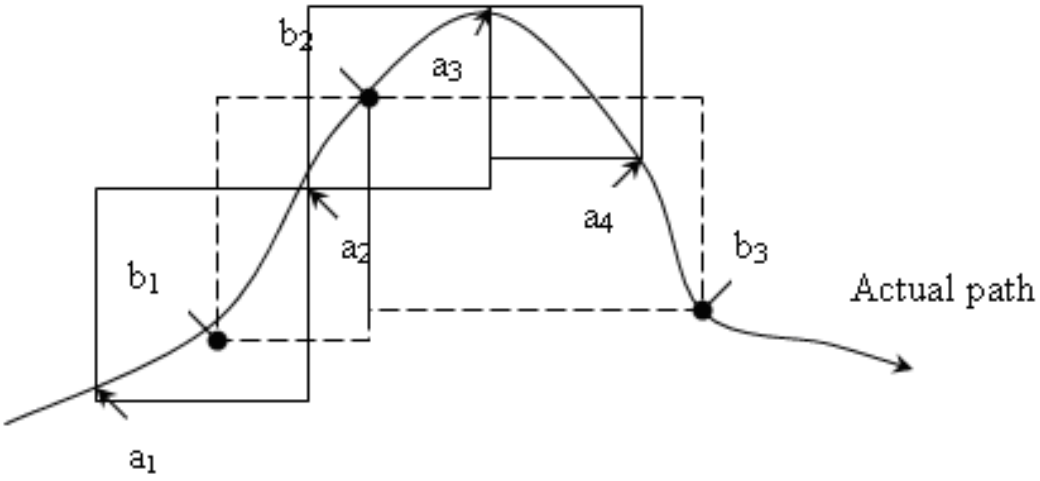


Figure 3.8. Merging in a path with sharp direction changes

generated by points in sequence B. Because there is a sharp direction change point "a_3" is missed when sequence A is merged into sequence B. Resulting sequences will be: $A^1 = < a_1, b_1, a_2, b_2, a_3, a_4 > B^1 = < b_1, a_2, b_2, b_3 >$

Since points in daily sequences also have time attributes, this also affects the merging rule. Because movement sequences are ordered in increasing time order, daily sequences obtained after the merging phase must be also ordered. For this reason time order of daily sequences must be preserved during merging operation. In order to fulfill this requirement time constraint is added to the merging rule. As shown in Figure 3.9 in order to insert point "b_1" between points "a_1" and "a_2", value of time attribute of point "b_1" must be between time attributes of points "a_1" and "a_2". However not all the times a user passes through a point at the same time during travels. Log data of different days are plotted on a 3D graph in Figure 3.10 for a user. When the log data is examined visually on the graph it is seen that the same path traveled in different days is shifted in time axis. For example sometimes there can be traffic congestions on the way from home to workplace. For this reason the *flexibility* values is used when

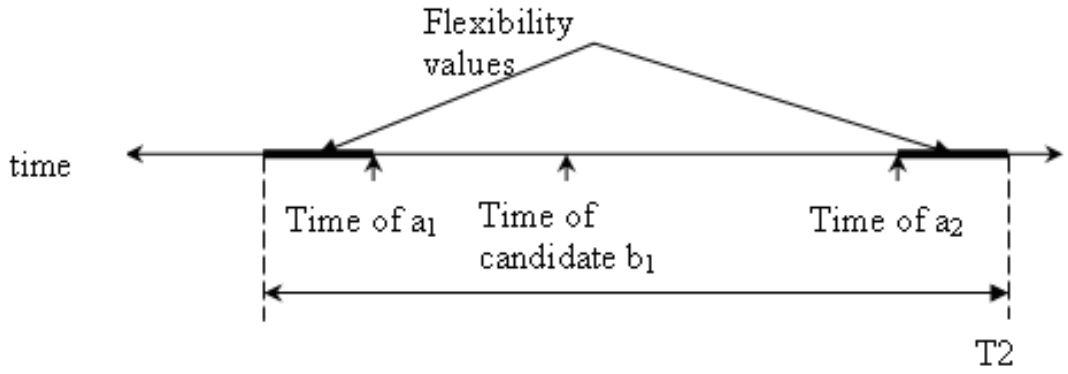


Figure 3.9. Comparing time attributes when inserting a point from one sequence into another

comparing times so, time values T_1 and T_2 shown in Figure 3.9 are used for comparison with time attribute of point "b_1".

Time constraint in the merging operation:

$T_1 = a_1.time$ - $flexibility$

$T_2 = a_2.time + flexibility$

if $T_1 < b_1.time < T_2$ **then**

Point b_1 can be merged between a_1 and a_2

end if

After similar sequences are modified to match each other the next step is to extract frequently used paths. This is achieved by finding frequently occurring point sequences. The algorithm for pattern mining presented in this thesis is based on the incremental mining for moving patterns algorithm presented in [15]. The procedure for mining moving patterns consists of three phases. The first one is data collection phase

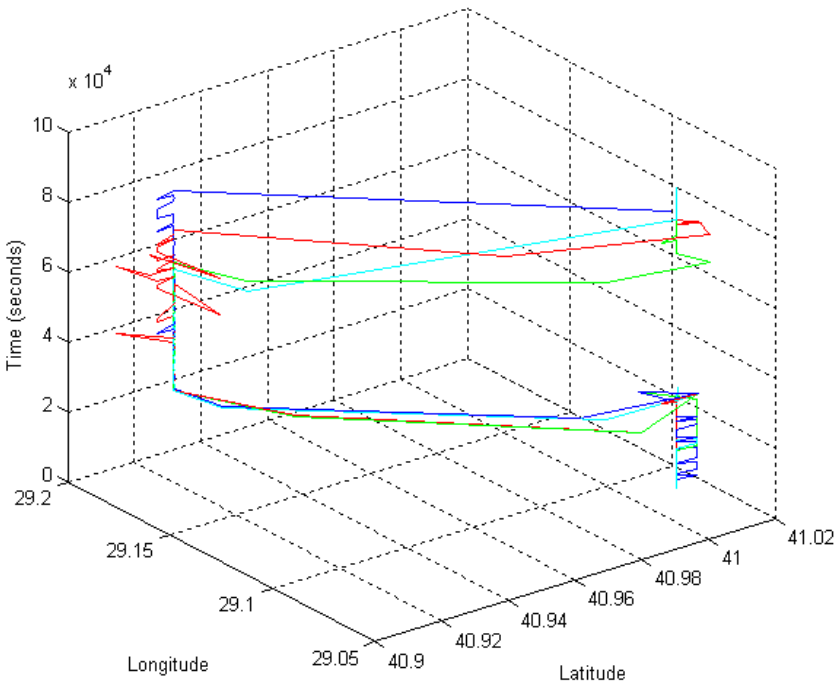


Figure 3.10. Time shift in the same path

where maximal moving sequences and occurrence count of moving pairs are determined from the movement log data. The second phase is incremental mining phase where large moving sequences are obtained from maximal moving sequences found in the first phase. The final phase is pattern generation phase where user movement patterns are determined from large moving sequences found in the second phase.

In the original algorithm a roundtrip model [17] is used where an initial starting point "S" for each user must be determined which is generally home place of users. Maximal moving sequence is the travel sequence starting from "S" and ending at "S". However when a user goes to a different place for a long time, for weeks or months, this algorithm will fail to find patterns of that user. For this reason maximal moving sequences in our algorithm are movement sequences starting from one important location and ending at another important location. As mentioned previously, important locations for a user are locations where this user spends time. The same example sequence a-a-a-a-a-b-t-t-m-r-k-k-k-k given in section 3.1.3 can be used for illustrative purposes. Important locations for this sequence are points "a" and "k". Because

point sequences between important locations are maximal movement sequences, point sequence "a-b-t-m-r-k" is obtained from this example log data as a maximal moving sequence.

After the set of maximal moving sequences is obtained, large moving sequences are found by using the algorithm in [15]. A large moving sequence can be determined by finding its occurrence count among all maximal moving sequences. In order a movement sequence to be a large moving sequence, occurrence count of this sequence in all maximal moving sequences must exceed the threshold $t_{occurence}$.

Definition 6. Occurrence threshold to consider a sequence as large moving sequence is $t_{occurence}$. If movement sequence $< p1, p2, p3, p4 >$ occurs more than $t_{occurence}$ times in all maximal moving sequences then this sequence is a large moving sequence.

Since points do not have time attributes in [15], points in the obtained large moving sequences also do not have time attributes. We introduced an additional phase to add time attributes to points in the large moving sequences. At the end of this phase movement patterns are obtained.

The problem in adding these time attributes is that all occurrences of a large moving sequence among all maximal moving sequences have different time values. In the example case shown in Table 3.3, large moving sequence $< a, b, c, e, f >$ is found 5 times among all maximal moving sequences and all occurrences have different time values. Solution proposed for this problem is to store different time values of a point in a list called *time_list* that is a property of the point. Each column in the table is a *time_list* for each point. In the prediction phase time attributes of points will be calculated by taking the arithmetic average of time values in *time_list* of the corresponding point. Time attributes are elapsed seconds from the beginning of the day. For instance time attribute of point "a" used in prediction phase is calculated in Equation 3.1.

$$\begin{aligned} Time\_attribute &= (20120 + 19990 + 60123 + 20240 + 60150)/5 \\ &= 36124 seconds \end{aligned} \tag{3.1}$$

Table 3.3. Different time attributes for movement sequence $< a, b, c, e, f >$

Point	a	b	c	e	f
Time attribute 1	20120	21000	22120	23500	24790
Time attribute 2	19990	20830	22300	23490	25000
Time attribute 3	60123	60250	60331	60481	60555
Time attribute 4	20240	21150	22220	23660	25010
Time attribute 5	60150	60289	60313	60444	60542

However from Table 3.3, it can be seen that three of the occurrences of the $< a, b, c, e, f >$ large moving sequence (1, 2, 4) took place in the mornings and two of them (3, 5) took place in the evenings. For example, a student can go from school to home in the evening one day and, in the noon other day. In this case the same path is traveled in totally different times and consequently when average of these times is taken an irrelevant value will be obtained as in the case in our example where time attribute of point "a" is found to be 36124 seconds. In order to eliminate this problem, time attributes of the large moving sequence $< a, b, c, e, f >$ must be split in two separate groups, one group containing time attributes 1, 2, 4 and the other containing time attributes 3, 5.

Solution to this problem is to group similar time values and to introduce separate large moving sequences for each group. Time attributes of the first point of the large moving sequence are used in the grouping process. For the example in Table 3.3, time attributes of point "a" are used for grouping. As a result of this process, two $< a, b, c, e, f >$ large moving sequences are obtained for each time attribute group.

All these large moving sequences obtained at the end of the mining movement patters process are movement patterns of mobile users. These patterns then will be used for predicting future locations of mobile users.

3.2. Location Prediction

In order to decrease the number of location query requests sent to the GSM operator, future locations of users are predicted using their movement and stationary

1: Input: Array a {time attributes of the first point in a large moving sequence}
2: Output: Set *group* {set of different groups}
3: Sort a in ascending order
4: Set $currentGroupNum$=0
5: **for** $i = 0$ to $a.size - 1$ **do**
6: **if** $i = 0$ **then**
7: Put $a[i]$ into $group[currentGroupNum]$
8: **end if**
9: **if** $(a[i+1] - a[i]) > t_{grouping}$ **then**
10: $currentGroupNum = currentGroupNum + 1$
11: **end if**
12: Put $a[i+1]$ into $group[currentGroupNum]$
13: **end for**

Figure 3.11. Algorithm for grouping time attributes

patterns. When location of a user is queried, firstly existence of a prediction result is checked for that user. If a prediction result exists, current location of this user is returned by using the information in the prediction result. If prediction result does not exist, the location of the user is queried from the GSM operator and obtained location is added to the log data of that user. Furthermore it is attempted to construct a prediction result by making a prediction about future locations of the user. Prediction for a user is made by using patterns of that user, current location of the user and the current time. Since there are two types of patterns, *stationary patterns* and *movement patterns*, there are two types of predictions, *point prediction* and *path prediction*. In point predictions, it is predicted that users will stay at a point for some time. For example if a person is at work, a prediction result containing leaving time from the workplace can be constructed by using stationary patterns of that person. In path prediction it is predicted that users will move on a path. For example if a student is on his way from school to home, a prediction result containing point sequence that represents the path from school to home can be constructed by using movement patterns of this student.

A prediction result has a confidence value that indicates trust level for that pre-

1: Input: Current point P
2: Input: Current time t
3: Output: PredictionResult
4: PredictionResult $result = null$
5: $result$ = predictPoint(P, t)
6: **if** ($result.confidence < c\_point$) **then**
7: $result$ = predictPath(P, t)
8: **end if**
9: **if** ($result.confidence < c\_path$) **then**
10: $result = null$
11: **end if**
12: **return** $result$

Figure 3.12. Algorithm for location prediction

diction. When constructing a prediction result firstly point prediction is made and if confidence value of this prediction is above the threshold value *c_point* this prediction result is used. If confidence level of the point prediction is lower than the threshold value *c_point* path prediction is made. If confidence value of this path prediction is above the threshold value *c_path* this prediction result is used otherwise no prediction result can be obtained. The algorithm is given in Algorithm 2.

3.2.1. Point Prediction

In this section, provided a user's current location p_{now}, current time t_{now} and current day of week d_{now} a prediction is made about leaving time of the user from this location. Firstly stationary patterns of a user those have the same coordinates as the current location p_{now} and the same day of week attribute as the current day of week d_{now} are read. For example, if a point prediction is done on Monday (d_{now} = Monday), only stationary patterns from previous Mondays are used. This approach is used because it is assumed that people have repeated behaviors on day of week basis. As mentioned before, stationary patterns have arriving time, arriving day of week, leaving time and leaving day of week attributes. Therefore in the day criteria, locations whose arriving

1: Input: Set of stationary patterns P_{all}
2: Input: Current time t_{now}
3: Output: Set of stationary patterns used for prediction P_{result}
4: **for** each stationary pattern $p \in P_{all}$ **do**
5: **if** $(p.arriving\_time < t_{now} < p.leaving\_time - t_{ignorance})$ **then**
6: $P_{result} = P_{result} \cup p$
7: **end if**
8: **end for**
9: **return** P_{result}

Figure 3.13. Algorithm for stationary pattern selection

day of week and locations whose leaving day of week are equal to the current day of week d_{now} are selected. Arriving and leaving days are used because a user can arrive at a point in one day and leave it in the next day. Homes of users are examples of such a case which is a most often case for most of the users. After retrieving points that match the criterion above, a second filter is applied. As shown in Figure 3.14, only points whose time intervals from arriving time to leaving time contain the current time tnow are chosen. Furthermore in order to obtain better predictions, points that have remaining time less than the threshold value $t_{ignorance}$ are ignored. The algorithm for stationary pattern selection is shown in Figure 3.13.

Arriving and leaving time attributes are times elapsed from the beginning of the day.

When the list of matching patterns P_{result} is obtained, remaining time to leave the point along with a confidence value for this prediction is calculated. Remaining time to leave is difference between leaving time attribute of the pattern and the current time. Remaining time is the average of remaining times of the selected stationary patterns. Furthermore a weight is assigned to each stationary pattern and these weights are used when taking the average of the remaining times. Stationary patterns that have date values close to the current date have higher weights. This makes the algorithm able to adapt to pattern changes that can occur over time.

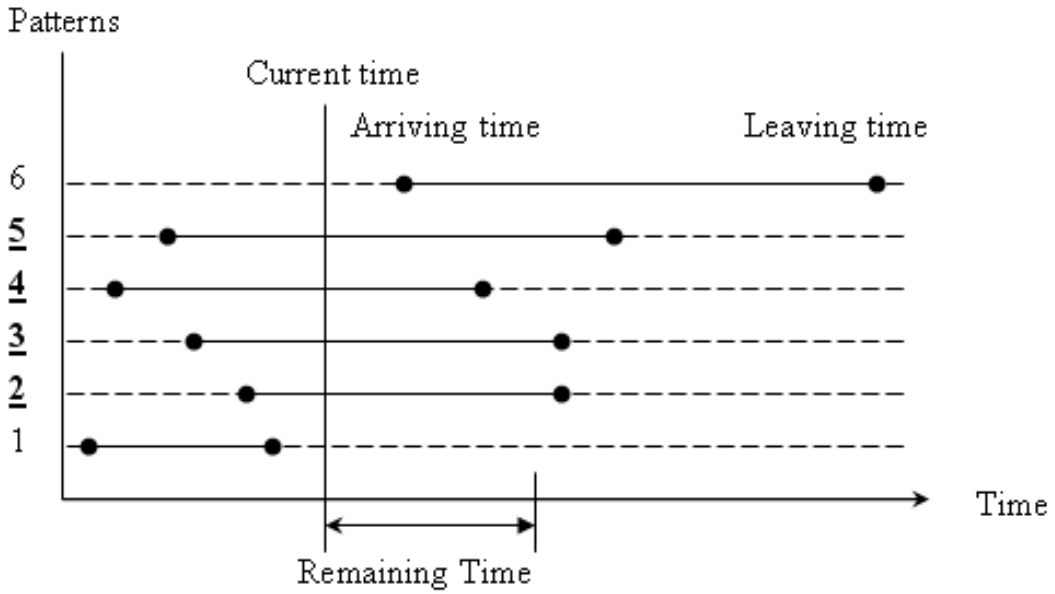


Figure 3.14. Choosing stationary patterns for point prediction

There are various weight calculation methods but all of them are based on time order of the patterns. Because all selected patterns have the same day of week attribute, consecutive patterns are equally distanced in time (one week), distance in weeks can be used as time order value. For example the simplest formula that can be used to calculate weights is $\frac{1}{distance^m}$ where $m \in R^+$. In the weight formula *distance* is distance in weeks and m is a positive real number used as exponent. If large m values are used more importance is given to recent patterns. However, even if m=2 a pattern that has distance value 4 will have weight value $\frac{1}{16}$ which implies that this pattern has almost no contribution to the average. For this reason, it is decided to use m=1 in the weight calculation formula.

$$Average\ of\ remaining\ time = \frac{\sum_{i=1}^{n} remaining\ time\ of\ p_i * weight\ of\ p_i}{\sum_{i=1}^{n} weight\ of\ p_i} \tag{3.2}$$

$$Confidence\ values = \frac{\sum_{i=1}^{n} weight\ of\ p_i}{\sum_{i=1}^{n} \frac{1}{i}} \tag{3.3}$$

$$Weight\ of\ p_i = \frac{1}{i}\ if\ p_i \in P_{result},\ 0\ otherwise \tag{3.4}$$

In the equation above, n is the total number of weeks, p_i is the i^{th} stationary pattern in the obtained pattern list and i is the distance in weeks from the current time. The resulting confidence value is a value between 0 and 1 and depends on the number of stationary patterns found.

For example, suppose that recent four weeks of the history log data are used in pattern generation phase (n=4), the week before from the current week has distance 1 and weight 1, the week two weeks before has distance 2 and weight $\frac{1}{2}$, the week three weeks before has distance 3 and weight $\frac{1}{3}$, and the oldest week has distance 4 and weight $\frac{1}{4}$.

Finally, a prediction result is obtained that has a remaining time value, a confidence value and a point value. Furthermore time of making this prediction is also stored. The meaning of this prediction result is that the user will stay at this point for a time indicated by the remaining time value since prediction making time. Attributes of a prediction result are shown below.

PredictionResult:

- prediction_time
- remaining_time
- confidence
- point

When a location query is issued if there is a point prediction result and if difference between querying time and prediction making time is smaller than the remaining time value, point in the prediction result is returned as an answer. Algorithm for point prediction is given below.

Suppose location of a user is queried at time t_{now} and there is a point prediction

result P. Value of t_{now} is the time elapsed from the beginning of the day.

if $P.confidence > c\_point$

$\wedge$

$P.prediction\_time < t_{now} < P.prediction\_time + P.remaining\_time$ **then**

return $p.point$

end if

3.2.2. Path Prediction

If a point prediction with confidence level greater than the threshold value *c_point* cannot be done, then path prediction is used to obtain a prediction about future locations of users by using their movement patterns. A movement pattern is chosen and a prediction result is built by using this pattern. The success of path prediction mainly depends on the chosen pattern. In order to determine which pattern the user is following, a movement sequence $S_{history}$ that is the sequence of last *history_num* points visited by the user is used. This sequence is compared with all movement patterns of a user and the pattern that matches best is chosen for prediction.

Sequence matching operation consists of matching of all points in the movement sequence $S_{history}$ with the points in movement patterns. The matching algorithm is very similar to the Mining Movement Patterns algorithm presented in Section 3.1.4. Suppose last *history_num* points visited by the user are $S_{history} =< b_1, b_2, b_3 >$ and A $= < a_1, a_2, a_3, a_4 >$ is a candidate movement pattern, if point "b_1" falls inside the rectangle generated by the points "a_1" and "a_2" as shown in Figure 3.5. this means that point "b_1" matches with the pattern A. Next, point "b_2" is checked for matching and then point "b_3" is checked for matching. Suppose that point "b_2" falls into the rectangle generated by "a_2" and "a_3", then matching of point "b_3" starts from the rectangle generated by points "a_2" and "a_3" because order of points in the movement sequence must be considered. A counter is used to keep the number of matched points in $S_{history}$. This counter then will be used in calculation of *confidence* of the prediction result.

Table 3.4. Arithmetic averages of points a_1 and a_2

Points	a_1	a_2
Time attribute 1	2012	2100
Time attribute 2	1999	2083
Time attribute 3	2024	2115
Average	2012	2099

In order to choose the best matching pattern a matching degree is calculated for each candidate pattern. Penalty value for each candidate pattern is calculated in order to obtain the matching degree. Penalty value is calculated by taking summation of penalties of each matching operation. For example penalty value of matching operation of point "b_1" is calculated by using time attributes of "a_1" and "a_2" given in Table 3.4. because "b_1" is matched between "a_1" and "a_2". If a point is not matched a constant value which is very large is assigned as its penalty value.

$$Penalty\ of\ b_1 = \frac{time of a_1 + time of a_2}{2} - time of b_1 \tag{3.5}$$

$$Penalty\ of\ pattern\ A\ =\ penalty\ of\ b_1 + penalty\ of\ b_2\ +\ penalty\ of\ b_3 \tag{3.6}$$

Suppose time attribute of point "b_1" is 2080, in this case penalty value of point "b_1" = $\frac{2012+2099}{2}$ - 2080 = 25. Lower penalty values indicate better matching and at the end the pattern with the smallest penalty values is chosen to be used in the prediction result.

As mentioned before, each point in a pattern has a *time_list* attribute that stores all time attribute of the point. For this reason time attributes of points "a_1" and "a_2" are calculated by taking arithmetic averages of their time attributes as show in Table 3.4.

After determining the best matching pattern a prediction result is constructed by using this pattern. Moreover a confidence value for this prediction result is calculated

and a *start_index* attribute is assigned to the prediction result.

$$Confidence\,Value = Num\,of\,matching\,points\,/\,history\_num \tag{3.7}$$

When a location query is issued if there is a path prediction result, location is predicted using this prediction result. Starting from the *start_index* of the pattern, time attributes of points of the pattern are compared with the current time.

Attribute *start_index* indicates at which point of the pattern the user is. For example the best matched pattern is $< a_1, a_2, a_3, a_4 >$ and the user currently is at a point between "a_2" and "a_3", starting index for this case is 3. By using a starting index value, when predicting future locations of the user only points "a_3" and "a_4" are used and points "a_1" and "a_2" are ignored because the user has already passed these points.

The point that has the closest time value to the current time is returned as an answer to the location query. An example is shown in Figure 3.15. where point "a_3" is predicted and is returned as an answer to the location query. Start index value is also updated to be the index of the returned point. If time attribute of the last point of the pattern is smaller than the current time, no point is returned that means a prediction can not be done and a location query from the GSM operator is issued.

If locations of users are queried very frequently from the GSM operator the charging amount increases so, a querying frequency limit is added to the proposed infrastructure. A minimum query period value for sending queries to the GSM operator is defined. Queries that are issued before the end of this period are not sent to the GSM operator and the last predicted point is returned as an answer.

The most important factor in prediction accuracy is regularity in movements. If users repeat the same movements regularly, their future locations can be predicted accurately. However, changing traffic conditions degrade regularity degree of movements. For example, using alternative routes and delays due to traffic congestions causes wrong

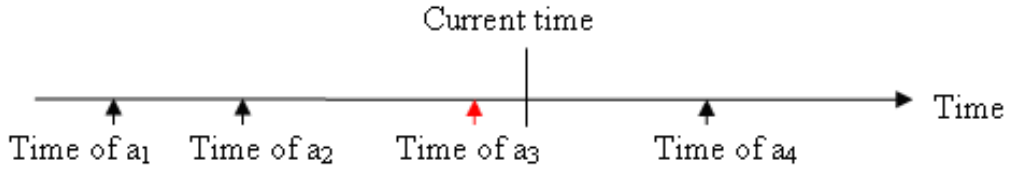


Figure 3.15. Predicting a point

predictions. If a person uses alternative route from home to workplace, predicting his future locations will be impossible because this route does not exist among patterns of this user.

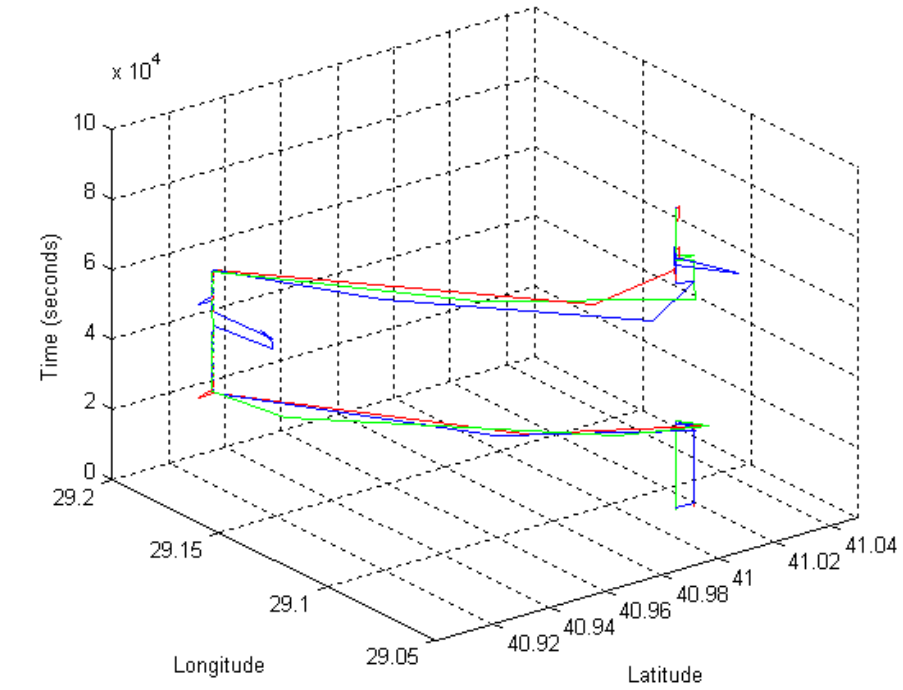
4. EVALUATION AND EXPERIMENTAL RESULTS

4.1. Test Data

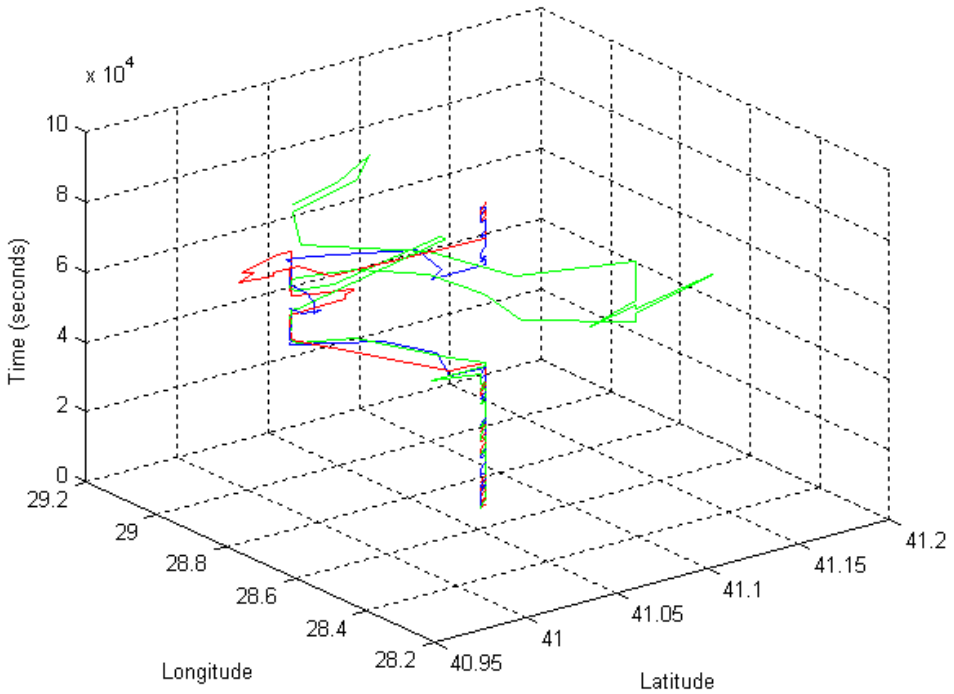
There is a five week movement log data of eight users. Log data consists of user ID, time stamp, latitude and longitude information as shown in Table 3.1. This location data is provided by a GSM operator and it is obtained as a result of calculations by using information from many base stations that listen a mobile device simultaneously [2]. A Matlab script is written in order to analyze the data graphically and to see whether there are similarities between movements of a user in different days. The script plots movements of a user on a 3D graph where x and y axes show latitude and longitude values and z axis shows time (number of seconds) elapsed from the beginning of the day. Movements of the same day in different weeks are plotted on a same graph, for example log data of Mondays are plotted on a same graph. Vertical lines on the graph can be interpreted as residing at a fixed location and horizontal ones as fast moving. In Figure 4.1 movements of two different users are shown. Movement of User 5 that is shown in Figure 4.1(a) is very regular and User 6 whose data is shown in Figure 4.1(b) has random movements.

Moreover in order to obtain artificial movement data for simulations, we implemented a data generator. Java based, object oriented generator is used to generate movement log data for different types of users. We adopted profile usage from [53] and created two different user profiles: users that move randomly and users that have movement regularities in daily basis. The movements of users between distinct points are simulated, where each user can have one of the profiles mentioned above.

Firstly a rectangular region is defined as the simulation area where all movements occur. Secondly a specified number of homes are generated where coordinates of homes are determined randomly within the simulation area [55]. Then, a specified number of users are generated. Finally a home and a profile are assigned to each user randomly.



(a)



(b)

Figure 4.1. Visualization of movements of two users, one with regular movements and one with random movements

A profile, called *RandomUserProfile*, is created in order to determine daily actions of randomly moving users. This type of user visits randomly generated places during all day. He leaves home in the morning and then goes to a randomly determined place, where he stays for a randomly determined amount of time. At the end of this residing time, a new location is determined randomly and the user travels to this new location. This process is repeated all the day and finally the user returns home in the evening.

In the *RandomUserProfile* earliest time to leave home, latest time to leave home, earliest time to return home and latest time to return home are specified. For each day, time to leave home is specified randomly between earliest time to leave home and latest time to leave home by using uniform distribution. Furthermore time to return home is specified randomly between values of earliest time to return home and latest time to return home. When it is time to leave home, a destination and residing time in this destination is generated randomly. We defined minimum and maximum limits for the residing time where the residing time is chosen randomly between these limits. Next, the user travels to this destination with a randomly determined speed and then stays in this destination. When the specified residing time finishes another destination and residing time is chosen randomly. Then the user travels to this destination and stays there for a time specified by the residing time value. This process continues in the same manner until current time passes the time to return home value. At this time the next destination of the user is specified to be his home. Finally the user travels to his home and stays there until the next morning.

Other profile, called *RegularUserProfile*, is created in order to determine daily actions of regular users. In this profile if it is a weekday a regular user goes to work, at lunch time he goes to lunch, after the lunch he comes back to work and at the evening he returns to his home. If it is a weekend regular users use *RandomUserProfile* and they behave like random users.

A predefined number of workplaces are generated in the beginning of the simulation where coordinates of workplaces are determined randomly within the simulation area. When a workplace is generated three places for lunch are also generated randomly

Table 4.1. Parameter values used in data generator

Number of Regular Users	100
Number of Random Users	20
Length of Training Data	6 weeks
Length of Test Data	2 weeks
Query Period	10 minutes
Number of Home Places	100
Number of Work Places	70

around this workplace.

When a user having *RegularUserProfile* is created a workplace is chosen randomly among all workplaces and it is assigned to this user. In this profile, time to leave home, time to go lunch, time to return from lunch and time to go home values are specified randomly between their corresponding lower and upper limits for each day. When it is time to leave home, the user starts traveling from home to workplace. The user stays in his workplace until lunch time and at lunch one of the three lunch places is chosen randomly. The user then stays at the chosen lunch place and when it is time to return from lunch the user goes back to his workplace. Finally, when it is time to go home the user starts traveling from his workplace to his home. It is assumed that regular users follow the same paths everyday. For example when a user travels from home to workplace this user always follows the same path but traveling times are different.

Furthermore because noise caused by the positioning system exists in the real data, artificial noise is introduced to the generated data. Coordinates of the generated points are modified with randomly generated deviation amount. An upper limit for the deviation amount is specified in order to avoid generating too large deviations.

Parameter values used in our simulations are given in Table 4.1. Query period is determined to be 10 minutes because query period in the provided log data is 10 minutes.

Movement data of different weekdays of two regular users are shown in Figure 4.2 and movement data of different days of two randomly moving users are shown in Figure 4.3.

4.2. Implementation

The algorithm proposed in this paper is implemented in Java programming language and MySql database is used. For simulations with the real log data the data of the first four weeks is used for training and the data of the last five days is used for testing. For simulations with the generated log data the data of the first six weeks is used for training and the data of the last two weeks is used for testing. A simulation program is implemented for testing the system. The program firstly generates patterns from the training data of a user. Then, points are read from the test data in increasing time order. For each point read from the test data, location of the user is queried. These queries are processed and answered by the proposed algorithm. Results of queries are compared with actual points and geographical distance between them is calculated. A parameter called $maxDeviation$ is defined that keeps the maximum threshold distance. If there is distance more than $maxDeviation$ between the actual point and the query result, this result is count as an erroneous result.

In the program, queries sent to the GSM operator are simulated by reading the actual point from the database. The program reports number of erroneous results and mean distance between the actual location and the predicted location. The program also reports number of location queries, and number of queries sent to the GSM operator.

Complexity of the proposed algorithm consists of complexity of training (pattern generation) and complexity of estimation (prediction) modules. There are arithmetic operations, comparisons, assignments and database accesses in the modules. Consequently because database access is the slowest operation among the operations mentioned above, complexity is indicated in terms of number of database accesses for each user. Each database access has 1 unit cost. Pattern generation operation is performed

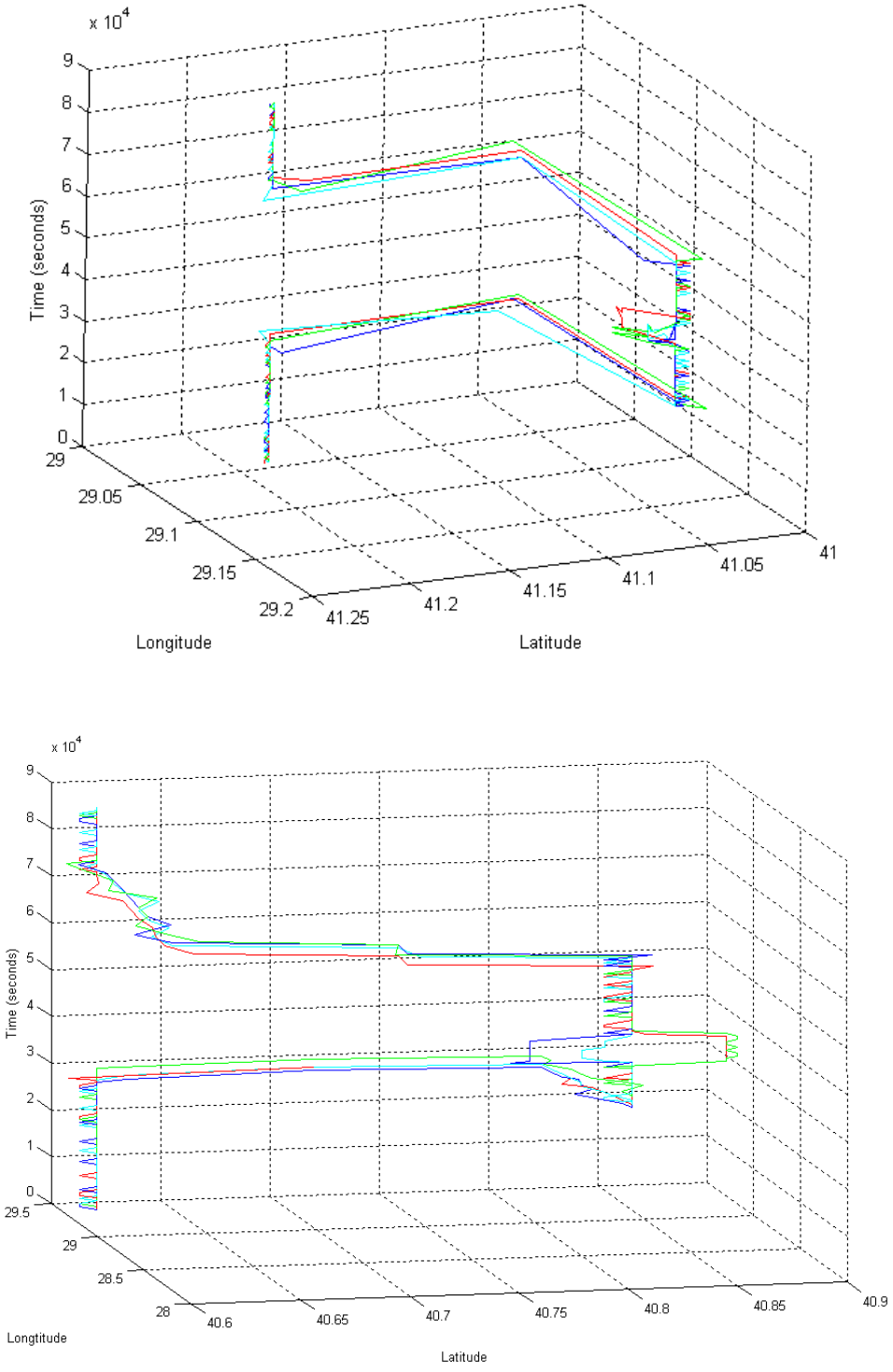


Figure 4.2. Movement data of two different regular users

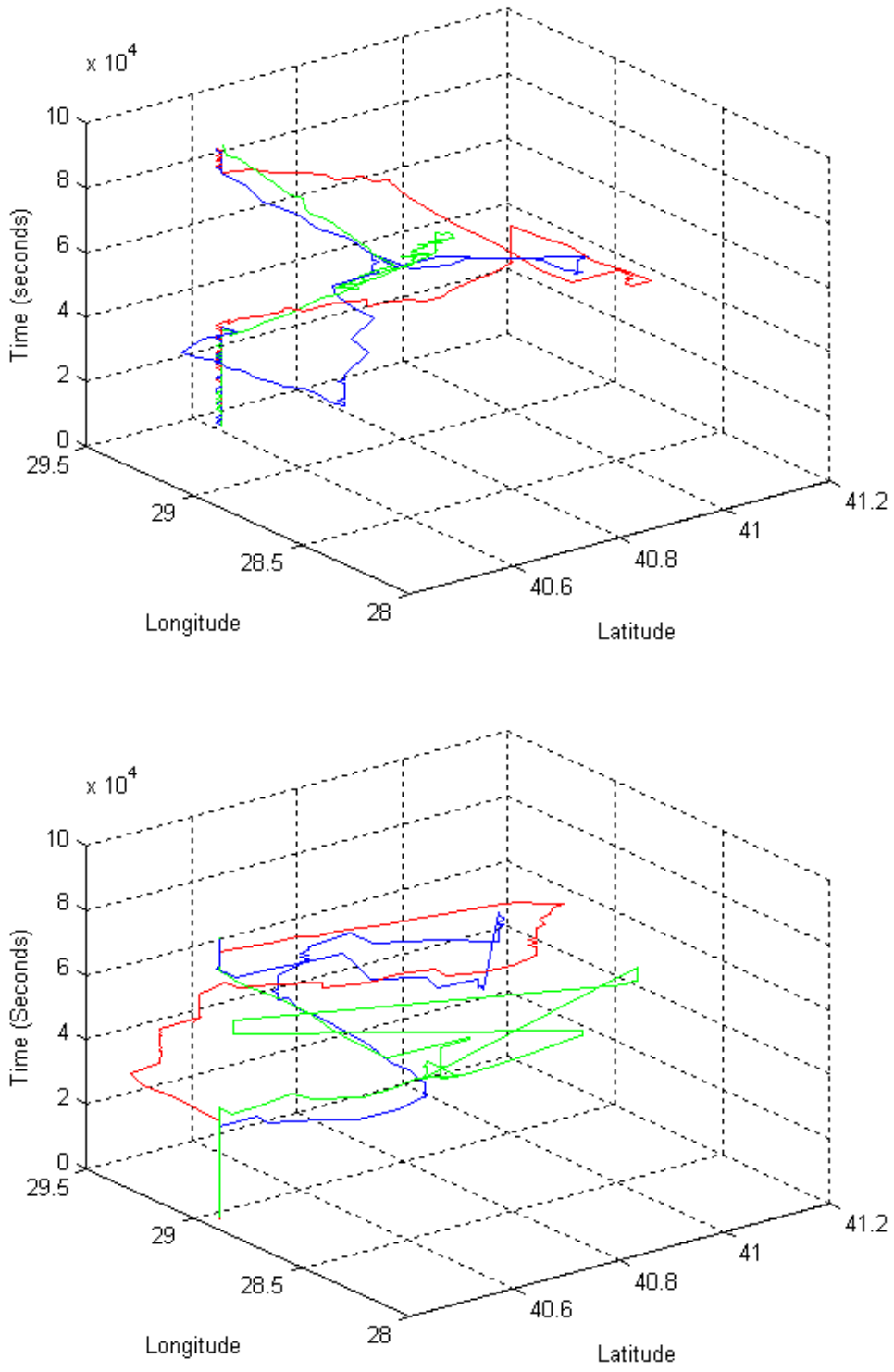


Figure 4.3. Movement data of two different randomly moving users

at the end of each day so, during a day there is no database access.

Complexity of training:

- Noise removal: All log data is read 2 times. (*Cost: 2*).
- Merging operation: Number of weeks in the log data is read (Cost: 1). Secondly the log data is read and log data obtained after merging operation is written. This operation is performed for each week (*Cost: 1*number of weeks*).
- Finding important locations: All log data is read (*Cost: 1*).
- Extracting movement patterns: All log data is read (*Cost: 1*).

Total Cost: *5 + number of weeks in the log data*

Complexity of estimation: All stationary patterns and all movement patterns are read after the pattern generation phase. As a result, there are 2 query executions.

4.3. Parameter Values

Optimal values of most of the parameters are found by running simulations for different parameter values with the real log data. The aim is to reduce the number of queries sent to the GSM operator while keeping the error rate in an acceptable level. However these two goals are conflicting. Decreasing number of queries will increase the error rate and in order to decrease the error rate, location of users must be queried frequently. Because there is a cost of making error and there is a cost of querying a location from the GSM operator, objective function is to decrease the total cost.

- Cost of issuing one location querying from the GSM operator is $cost_{query}$
- Cost of making one error is $cost_{error}$
- Total cost = Number of queries * $cost_{query}$ + Number of errors * $cost_{error}$

Because there are two conflicting success criteria, choice for optimum values of the parameters depends on the $cost_{error}$ and $cost_{query}$ values. In the case where $cost_{query}$

Table 4.2. Symbol table for parameters used in simulations

Parameter	Definition	Default values
$t_{occurence}$	Occurrence threshold to consider a sequence as pattern	5
t_{time}	Threshold time for determining important places	10 min
c_{point}	Threshold confidence value for point predictions	0.4
c_{path}	Threshold confidence value for path predictions	0.4
$t_{matching}$	Minimum number of matched points for matching	3
$t_{grouping}$	Threshold time for grouping time attributes	60 min
$flexibility$	Flexibility time used in comparison of time attributes	15 min
$t_{ignorance}$	Minimum remaining time to use a stationary pattern	10 min
$history\_num$	Number of points in the recent moving history	3
query period	Minimum query period	15 minutes
$maxDeviation$	Threshold distance for error decision	1 km

is larger optimum values will be different than the case where $cost_{error}$ value is larger.

In our simulations, we assumed a scenario where $cost_{query} = 2$ units and $cost_{error} = 3$ units. These values are chosen in order to make both number of queries and number of errors affect the total cost equally. When simulation results are examined, it is observed that if larger $cost_{query}$ is used total cost follows the trend of number of queries, alternatively if larger $cost_{error}$ is used total cost follows the trend of number of errors. The effect of using different cost values is shown in Section 4.3.1.

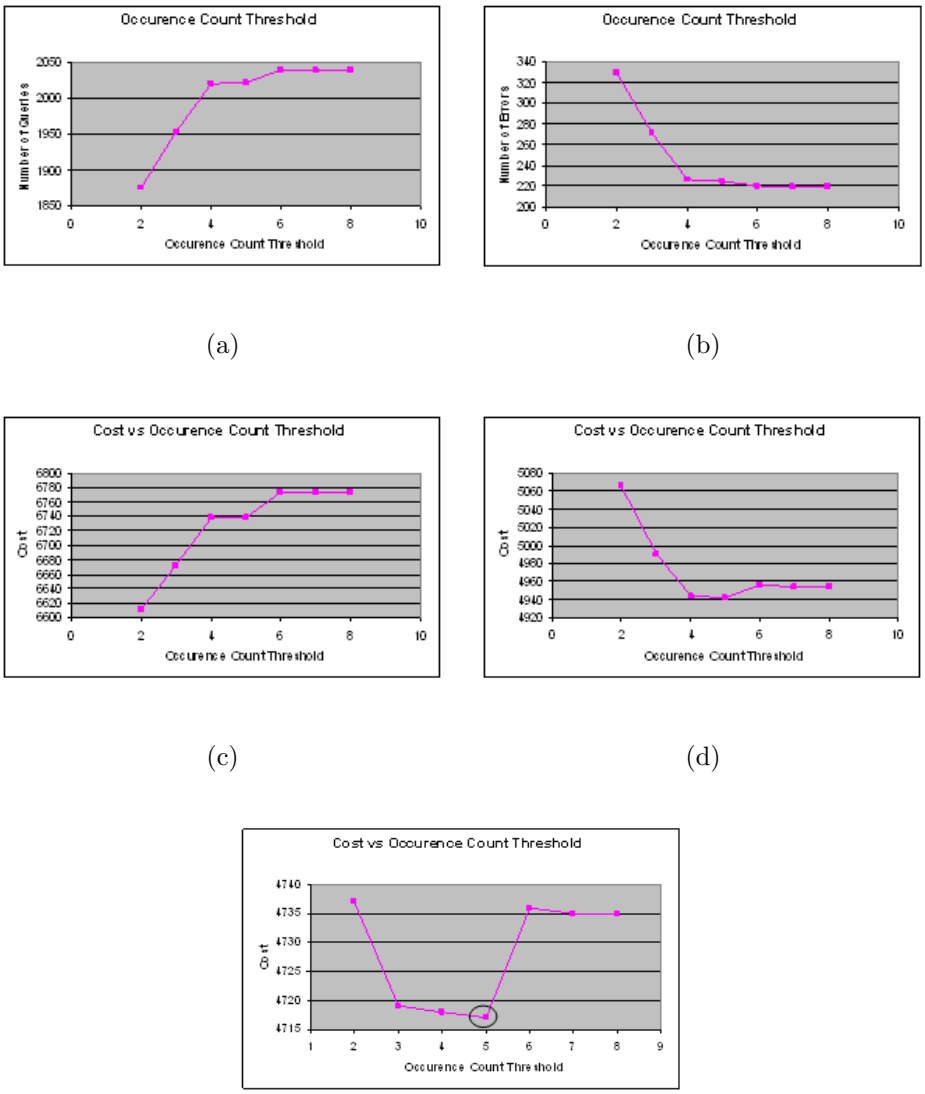
All log data provided by the GSM operator is used in simulations for determining parameter values. Parameter values used in simulations are shown in Table 4.2. However, the values written in the table are not always the optimal ones, as mentioned before optimal values can be different for various cost values. Furthermore for different data sets those contain movement behaviors of different users, optimal values can be different. The aim of presenting these values is to provide a reference point for optimization.

4.3.1. Occurrence threshold to consider a sequence as pattern

Occurrence threshold indicates minimum occurrence count of a movement sequence to be a movement pattern. If high threshold values are used number of mined movement patterns will be low but, obtained patterns will be very sound. In this case only very frequently repeated paths are chosen as patterns. Alternatively if low threshold values are used many movement patterns will be mined but, some of these patterns will be unsafe. Consequently simulations are performed on the test data in order to determine optimal threshold value. Results of these simulations are plotted in Figure 4.4. Resulting cost curve by using $cost_{query} = 3$ units and $cost_{error} = 3$ units is shown in Figure 4.4(c). Cost curve with these cost values follows the trend of cost of number of queries. On the other hand resulting cost curve by using $cost_{query} = 2$ units and $cost_{error} = 4$ units follows the trend of cost of number of errors as shown in Figure 4.4(d). In order to make both costs contribute equally to the result, we decided to use $cost_{query} = 2$ units and $cost_{error} = 3$ units as shown in Figure 4.4(e). As a result of these simulations value of $t_{occurence}$ is chosen to be 5 because, total cost is minimum for this value as shown in Figure 4.4(e).

4.3.2. Threshold time for determining important places

Threshold time for determining important places affects mining of stationary patterns. In order to find optimal value for this parameter the numbers of places found for many values of t_{time} are plotted on Figure 4.5. However, an obvious point could not be found from the graph so, the ttime value is found by simulations. Simulations with different t_{time} values are shown in Figure 4.6. Increasing this value decreases number of important places so, number of stationary patterns decreases. Small number of stationary patterns leads to less point predictions. On the other hand using small t_{time} values causes finding of large number of important places so, making many point predictions. Minimum limit for this value is the query period. Because query period in our test data is ten minutes it is not possible to use smaller values for t_{time}. According to simulation results t_{time} value is chosen to be ten minutes for this test data because, total cost is minimum for this value as shown in Figure 4.6(c). Furthermore in [1], also



(a) (b)

(c) (d)

(e)

Figure 4.4. Simulation results for various $t_{occurence}$ values; number of queries vs $t_{occurence}$, number of errors vs $t_{occurence}$, cost vs $t_{occurence}$ with larger $cost_{query}$, cost vs $t_{occurence}$ with larger $cost_{error}$ and cost vs $t_{occurence}$

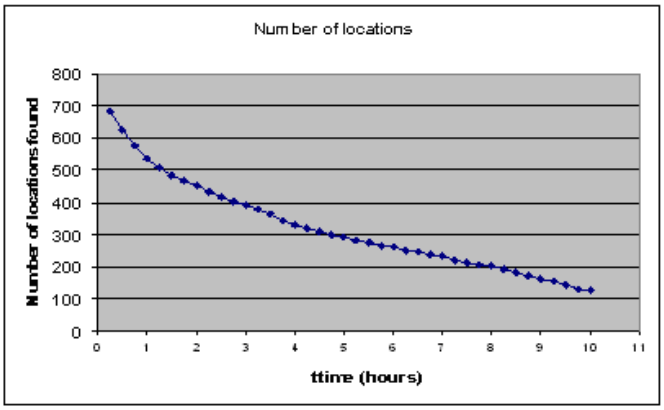


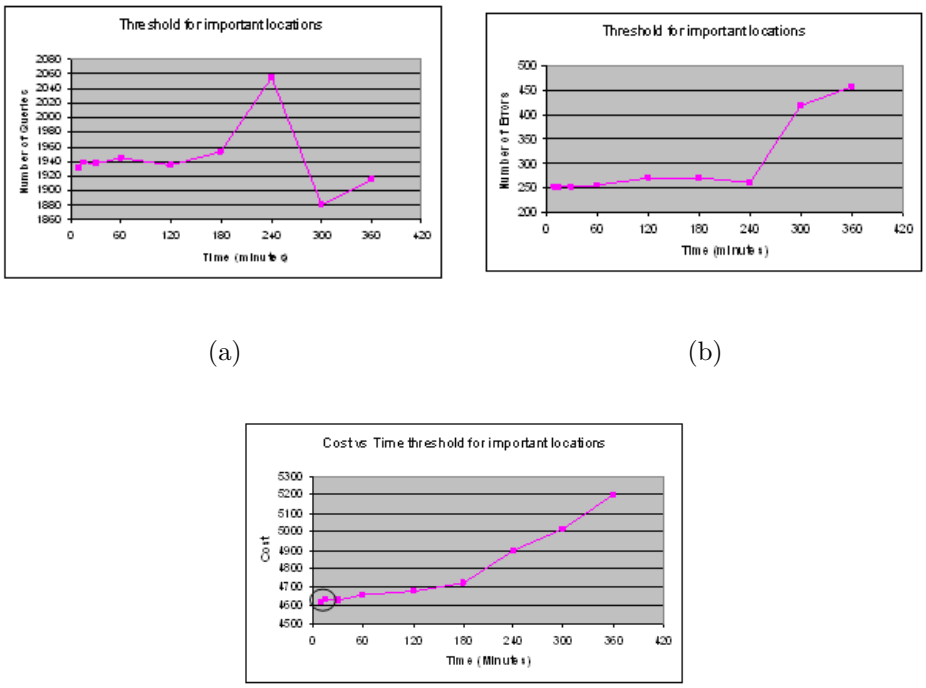
Figure 4.5. Number of important places for all users

value of ten minutes is used as stopping time to determine important locations.

4.3.3. Threshold confidence values for point and path predictions

Threshold confidence value for point predictions and threshold confidence value for path predictions are parameters that are used to decide whether to trust and use a prediction or not. As indicated previously, when a point prediction is made its confidence value is compared with the threshold value c_{point} and if this confidence value is larger than the threshold the prediction is used for predicting future locations. When confidence value does not exceed the c_{point} value, path prediction is made. Then threshold value c_{path} is used to decide whether to use the path prediction or not.

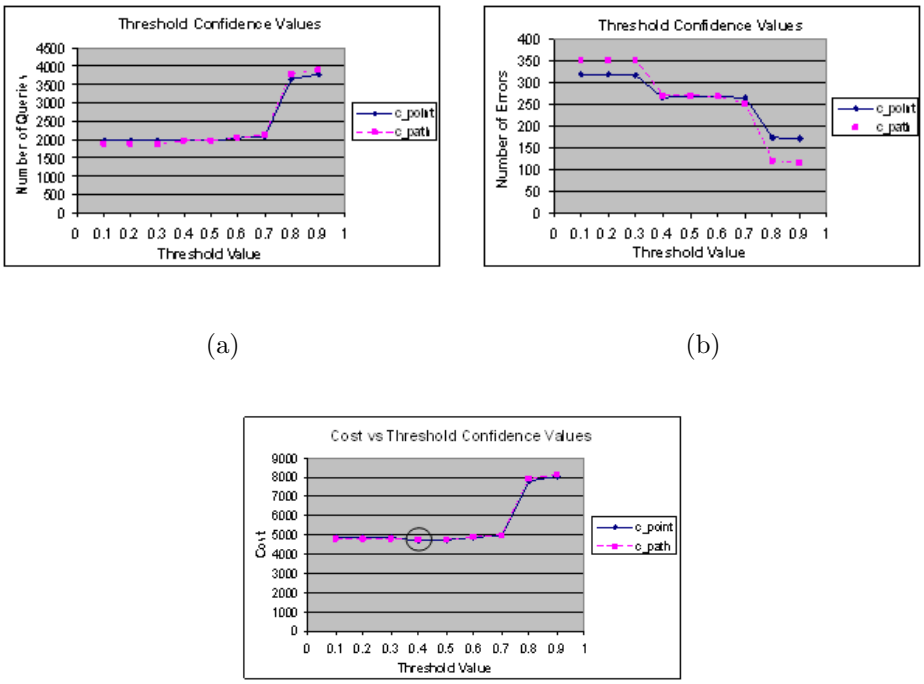
In the case where these threshold values are increased, only sound predictions are used so, error rate decreases but number of queries sent to the GSM operator increases. On the other hand when these threshold values are decreased making prediction becomes easy but the probability of having some incorrect predictions gets higher. Optimal values of these parameters are found by simulations whose results are shown in Figure 4.7. The dashed lines on figures show c_{path} values and the solid lines show c_{point} values. After inspecting these results optimum confidence threshold for point prediction (c_{point}) and confidence threshold for path prediction (c_{path}) are decided to be 0.4 because, total cost is weak minimum for this value as shown in Figure



(a) (b)

(c)

Figure 4.6. Simulation results for different t_{time} values; number of queries vs t_{time} , number of errors vs t_{time} and cost vs t_{time}



(a) (b)

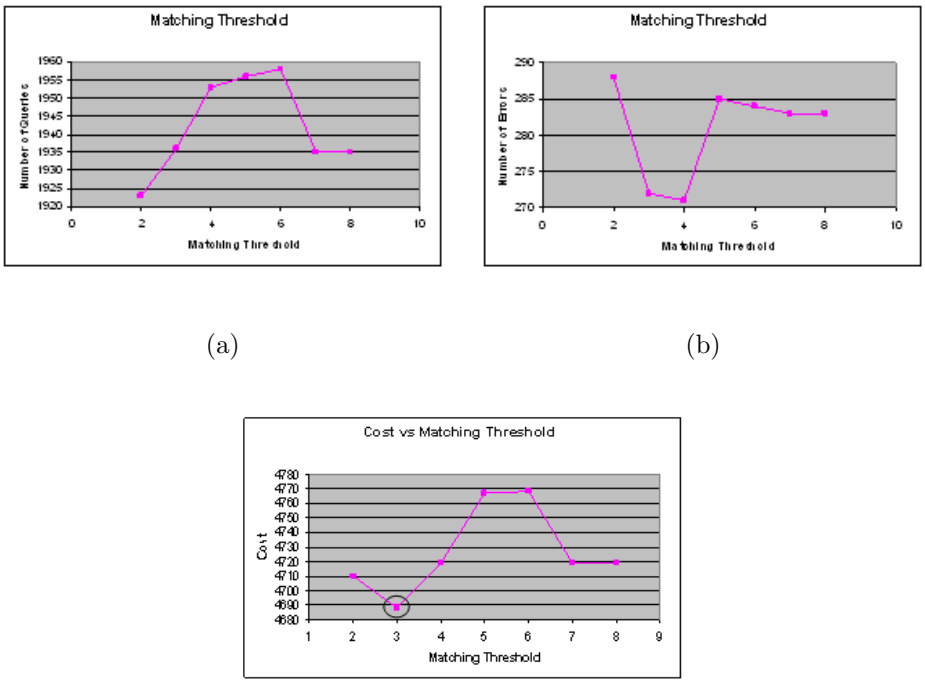
(c)

Figure 4.7. Simulation results for various threshold confidence values; number of queries vs threshold confidence, number of errors vs threshold confidence and cost vs threshold confidence

4.7(c).

4.3.4. Minimum number of matched points to consider two sequences matching

The $t_{matching}$ parameter is used to prevent introduction of additional noise during sequence merging operation. This parameter is used in mining of movement patterns, so it effects path predictions. If the value of this parameter is increased too much there will be no merging operation between sequences. On the other hand if this value is decreased too much merging operation will be performed with irrelevant sequences and this will introduce additional noise. Optimal value for this parameter also is found after simulations with different $t_{matching}$ values. Results of these simulations are shown in Figure 4.8 and according to these simulation results $t_{matching}$ value is chosen to be 3. When total cost is considered minimum value is obtained with matching threshold



(a) (b)

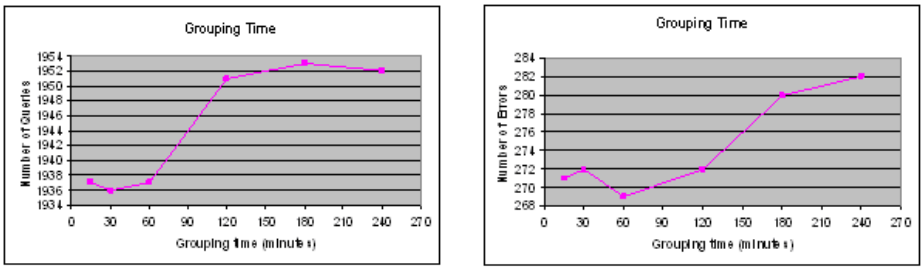
(c)

Figure 4.8. Simulation results for various $t_{matching}$ values; number of queries vs $t_{matching}$, number of errors vs $t_{matching}$ and cost vs $t_{matching}$

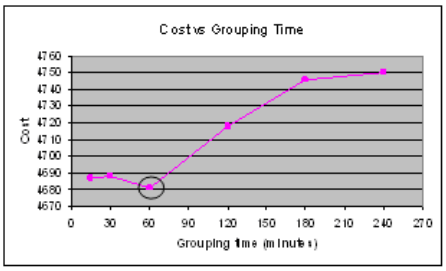
value 3 as shown in Figure 4.8(c).

4.3.5. Threshold time for grouping time attributes

Threshold time for grouping time attributes is the key element of time grouping operation. Number of time groups is determined by $t_{grouping}$ value. If the value of this parameter is increased, more values will fall into the same group and the number of groups will be lower. On the other hand, if this value is decreased only close values will be able to form a group and the number of groups obtained will be larger. Simulation results for different $t_{grouping}$ values are shown in Figure 4.9. Because this parameter is used in mining of movement patterns it effects only path predictions. For this reason, this parameter has little impact on the results. After examining simulation results, a value of 60 minutes is chosen for this parameter. As it can be seen from Figure 4.9(c) minimum cost is obtained when $t_{grouping}$ value is 60 minutes.



(a) (b)



(c)

Figure 4.9. Simulation results for various $t_{grouping}$ values; number of queries vs $t_{grouping}$, number of errors vs $t_{grouping}$ and cost vs $t_{grouping}$

4.3.6. Flexibility time used in comparison of time attributes

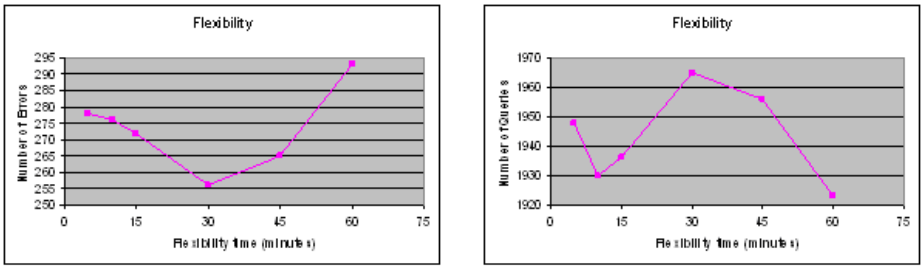
The effect of $flexibility$ parameter is shown in Figure 3.9. It is used to provide some degree of flexibility when comparing time attributes of points during movement sequence matching operation. If the value of this parameter is increased too much difference between upper and lower time boundaries will be very big so, the time constraint will be ineffective. On the other hand if the value of this parameter is decreased too much the time constraint will be too tight and the $flexibility$ parameter will be ineffective. This parameter effects mining of stationary patterns and hence path prediction. Because percentage of path predictions is much less than percentage of point prediction among all prediction, the effect of this parameter on the results is small. Optimal value for this parameter is found by simulations with different $flexibility$ values. Results of the simulations are shown in Figure 4.10 and according to these results optimal value for this parameter is chosen to be 15 minutes. The reason of choosing this value is that total cost is minimum for this $flexibility$ value as shown in 4.10(c). The optimal value found for this parameter is a reasonable one because in daily trips to home, to work and to school, etc. a person can have such time deviations as shown in Figure 3.10.

4.3.7. Minimum remaining time to use a stationary pattern

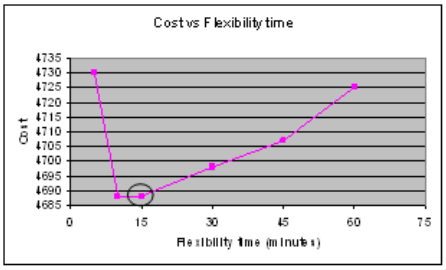
The aim of using threshold value $t_{ignorance}$ is to avoid from using stationary patterns that have very small remaining time attributes. Because these patterns will be invalid after a short time, they are ignored in stationary pattern selection phase. Using a stationary pattern that will be invalid before the next query is useless, for this reason a value at least as query period must be chosen. Query period used during simulations is 10 minutes for this reason value for $t_{ignorance}$ is chosen to be 10 minutes.

4.3.8. Number of points in the recent moving history

This parameter is used when deciding which movement patterns the user is following. Before deciding to a pattern result of *history_num* location queries must be

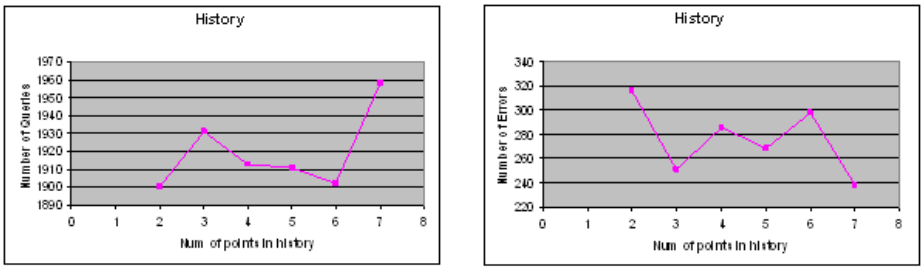


(a) (b)

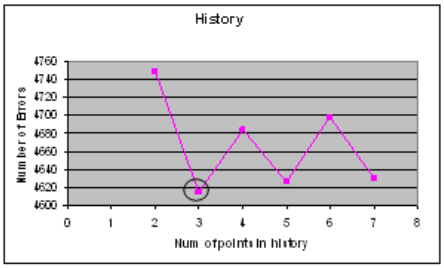


(c)

Figure 4.10. Simulation results for various *flexibility* values; number of queries vs *flexibility*, number of errors vs *flexibility* and cost vs *flexibility*



(a) (b)



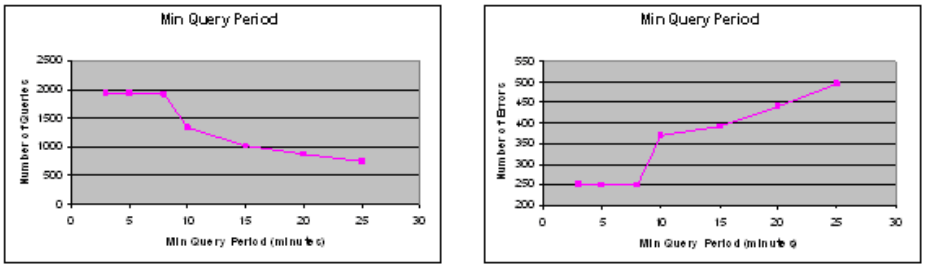
(c)

Figure 4.11. Simulation results for various *history_num* values; number of queries vs *history_num*, number of errors vs *history_num* and cost vs *history_num*

obtained. If very large *history_num* values are used deciding to a movement pattern will be very difficult but, probability of choosing the right pattern will be high. On the other hand if small *history_num* values are used choosing a pattern will be easy but, there will be a high probability of choosing a wrong movement pattern. This parameter has only effect on path predictions. Optimal value for this parameter is found by simulations with different *history_num* values. Results of the simulations are shown in Figure 4.11 and according to these results optimal value for this parameter is chosen to be 3. The reason of choosing this value is that total cost is minimum for this *history_num* value as shown in Figure 4.11(c).

4.3.9. Minimum query period

Although by the GSM operator minimum query period is defined to be 15 minutes, behavior of the algorithm for different minimum query periods is shown in Figure 4.12. Because requirements can change over time it is important to know system re-



(a) (b)

Figure 4.12. Simulation results for different minimum query periods; number of queries vs min. query period and number of errors vs min. query period

sponse to requirements changes. Number of queries sent to the GSM operator and number of errors are plotted for different minimum query periods. Because query period in the provided log data is 10 minutes a knee can be seen on the graphs for this value. Results for smaller query periods than 10 minutes are the same because, the maximum query period that can be achieved with the provided log data is 10 minutes. On the other hand, for larger values, number of queries sent to the GSM operator decreases almost linearly when minimum query period is increased and number of errors increases almost linearly when minimum query period is increased.

4.4. Simulation Results

Error count results of simulation with different *maxDeviation* values are shown in Figure 4.13. A knee exists where *maxDeviation* is 1 km and for smaller values than 1 km number of errors increases dramatically. Furthermore by using distances between the actual locations and the predicted locations in each location query, average distance is calculated. For all predictions average deviation from actual locations is found to be 0.8 km so, if a value slightly greater than the average value is used for *maxDeviation* much more deviations will be less than the *maxDeviation* and number of errors will be small. This result explains the knee observed when maxDeviation is 1km so, in simulations value of 1 km is used for *maxDeviation* value. Results of simulation with real data for each user are show in Table 4.3. It can be seen that error rate is 8.34% and only 21.42% of the queries are sent to the GSM operator. Furthermore, the average

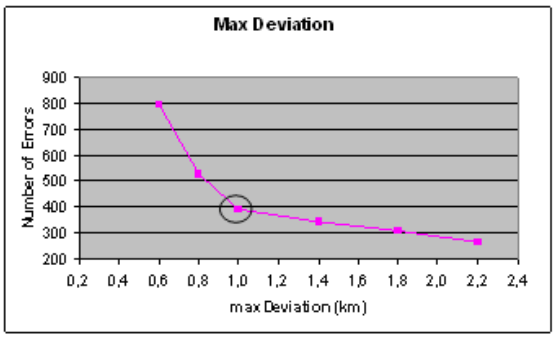


Figure 4.13. Simulation results for various *maxDeviation* values

distance between the actual location and the predicted location is 0.79 km. A script is written in Matlab to plot the actual path and the predicted path on a 3D graph. By using these graphs simulation results can also be examined visually as shown in Figure 4.14. More graphs those show simulation results are presented in Appendix A. The solid lines on the figures are actual paths and the dashed lines are the predicted paths. Furthermore, overall simulation results for both real data and generated data are shown in Table 4.4. It can be seen that both results are similar.

Results are affected mainly from users' regularities in movements. Because future location of users that have regular movement patterns can be predicted easily, the proposed algorithm performs well with these users. However users that move randomly most of the time must be queried frequently for accurate tracking. Differences in performance results for various users define the fairness of the system. Fairness of the system for different users is observed by comparing the minimum and the maximum error rates and the minimum and the maximum percentages of queries sent to the GSM operator. Large difference between these minimum and maximum values implies an unfair system whereas if the difference is smaller this means that system is fair for all users. Fairness results of real data are show in Table 4.5 and fairness results of generated data are shown Table 4.6.

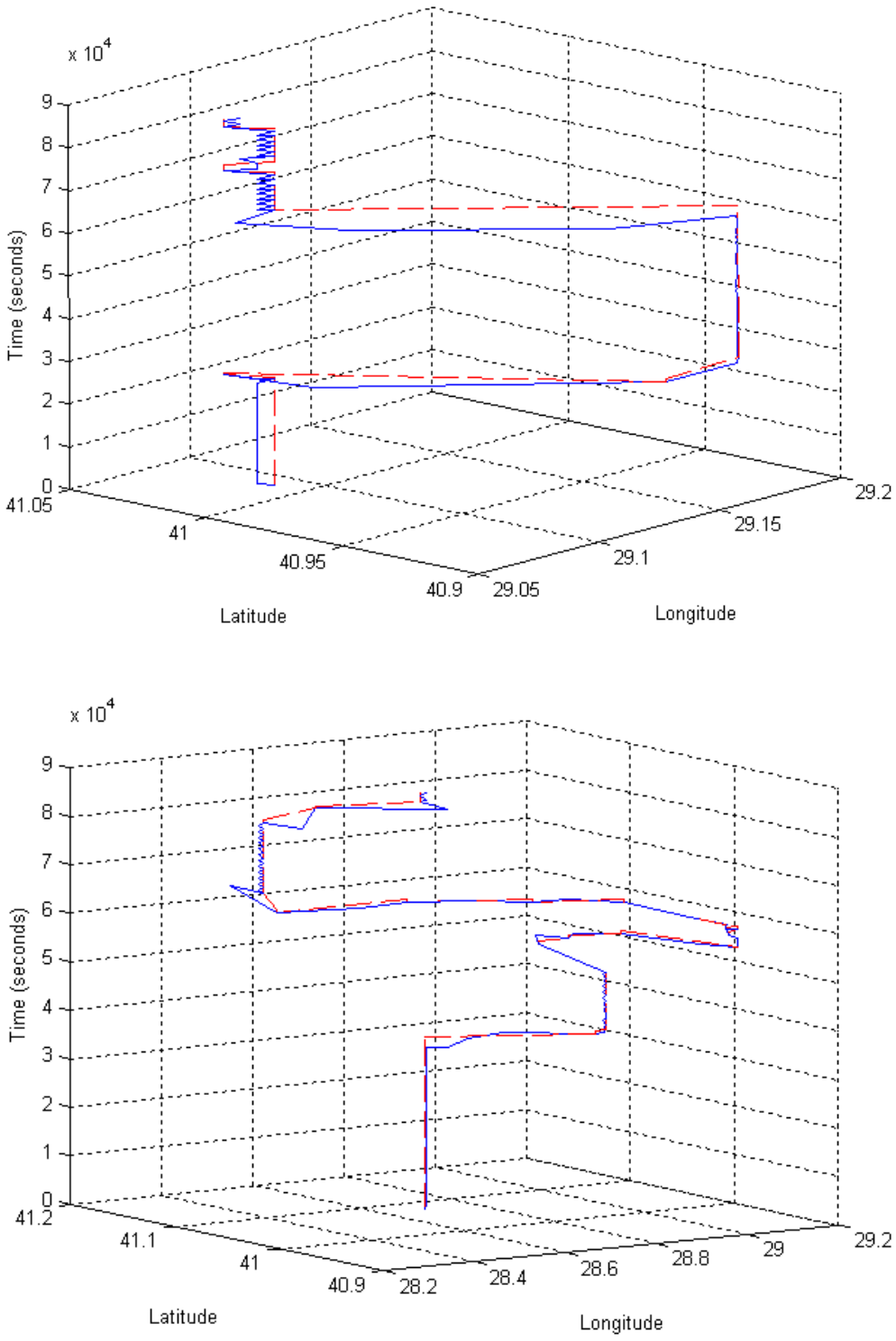


Figure 4.14. Comparison of the actual path with the predicted path

Table 4.3. Simulation results for each user in the real data

User ID	Error Count	Error Rate	Query Count from the Operator	Query Rate	Total Query Count	Avg. Error Distance (km)
1	97	16.14%	74	12.31%	601	0.71
3	38	6.30%	240	39.80%	603	0.37
4	76	14.90%	110	21.57%	510	0.57
5	22	3.67%	80	13.36%	599	0.61
6	54	9.14%	123	20.81%	591	0.92
10	58	9.72%	130	21.78%	597	1.72
12	11	1.89%	87	14.97%	581	0.30
13	35	5.79%	160	26.45%	605	1.13
Total	391	8.34%	1004	21.42%	4687	0.79

Table 4.4. Simulation results

	Real Data	Generated Data
Error Rate	8.34%	10.17%
Query Rate	21.42%	24.14%
Average Error Distance	0.79 km	1.29 km

Table 4.5. Fairness results of the real data

Min. Error Rate	1.89%	Min. Query Rate	14.97%
Max. Error Rate	16.14%	Max. Query Rate	39.80%
Std. Dev. for Error Rate	5.08%	Std. Dev. for Query Rate	8.89%

Table 4.6. Fairness results of the generated data

Min. Error Rate	0,50%	Min. Query Rate	6,00%
Max. Error Rate	32,89%	Max. Query Rate	59,87%
Std. Dev. for Error Rate	6,46%	Std. Dev. for Query Rate	14,98%

5. CONCLUSIONS

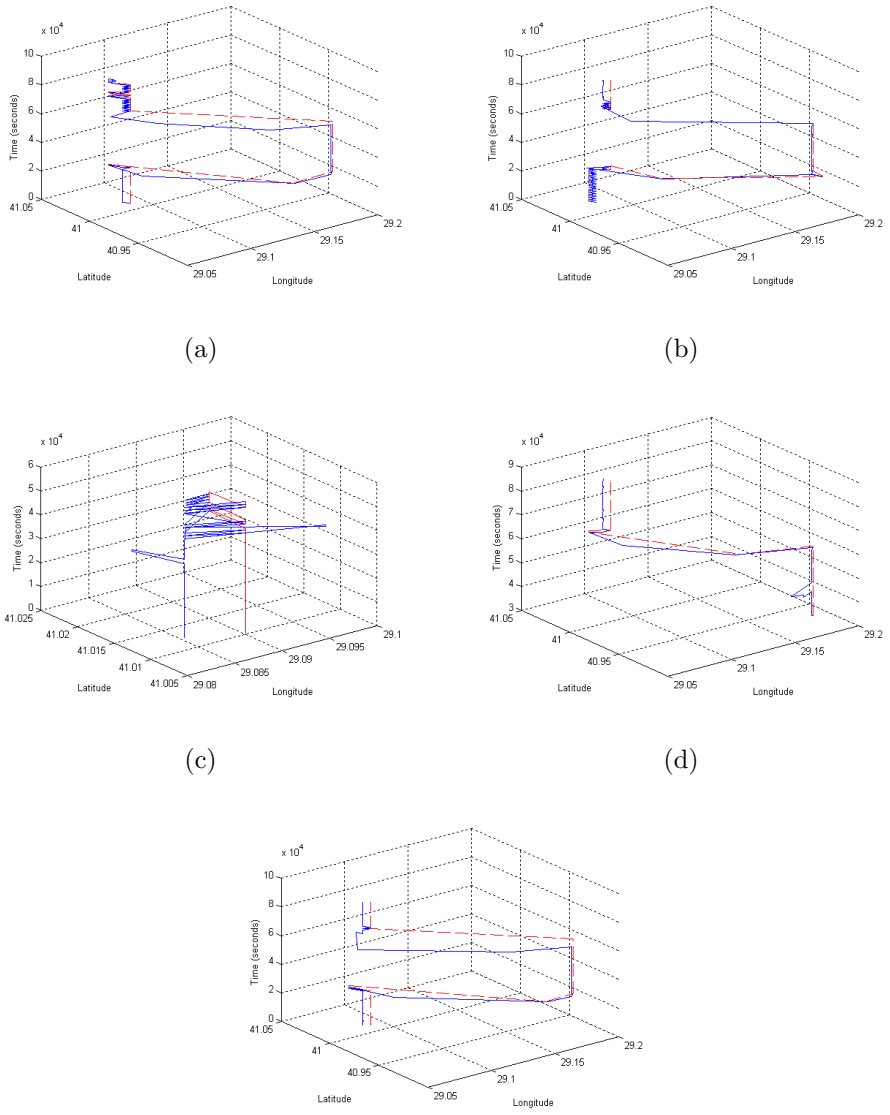
The goal is to decrease the number of location queries sent to GSM operators when tracking users. We have presented a method that reduces number of location queries and tracks users efficiently. The algorithm achieves this by mining movement patterns of users and predicting location by using these patterns. The technique used in this algorithm is to separate movement log data into two parts, point sequences that represent staying at a location and point sequences that represent traveling. Consequently, two different patterns, movement and stationary and two different predictions, point prediction and path prediction are introduced. Furthermore different point sequences representing the same path are merged in order to be comparable with each other.

The presented algorithm is evaluated with real data provided by a GSM operator and with artificially generated data. It is seen that the algorithm reduces the number of location queries significantly and predicts locations of users with good accuracy.

On the other hand the algorithm can be extended to be able to extract group behaviors. For example, in activities like sports contests, concert, shows, etc. querying locations of only a small number of users will give information about locations of the other users participating the activity. Moreover, a third type of pattern can be introduced that will represent transitions between stationary and movement patterns. For example, after staying at a location for some time, people usually follow predefined routes. Speeds of users can be calculated and the provided services can be differentiated according to speed information. If it is known for example that a user is moving very fast traffic information can be provided to this user. Alternatively if a user is residing at a location or moving slowly playing movies in cinemas around can be informed to this user. Finally, in order to reduce the amount of erroneus results caused by the deviation from the patterns locations of users can be queried from the GSM operator in specified time periods. This period can be differenet for different type of users. Users that deviate their patterns often can be queried frequently.

APPENDIX A: VISUAL COMPARISON OF PREDICTIONS

Prediction results of three arbitrarily chosen users are shown visually. Each figure shows actual path and predicted path for one day log data of a user. Because the test data consists of five days, five figures are drawn for each user. Solid lines show the actual path and the dashed lines show the predicted path.

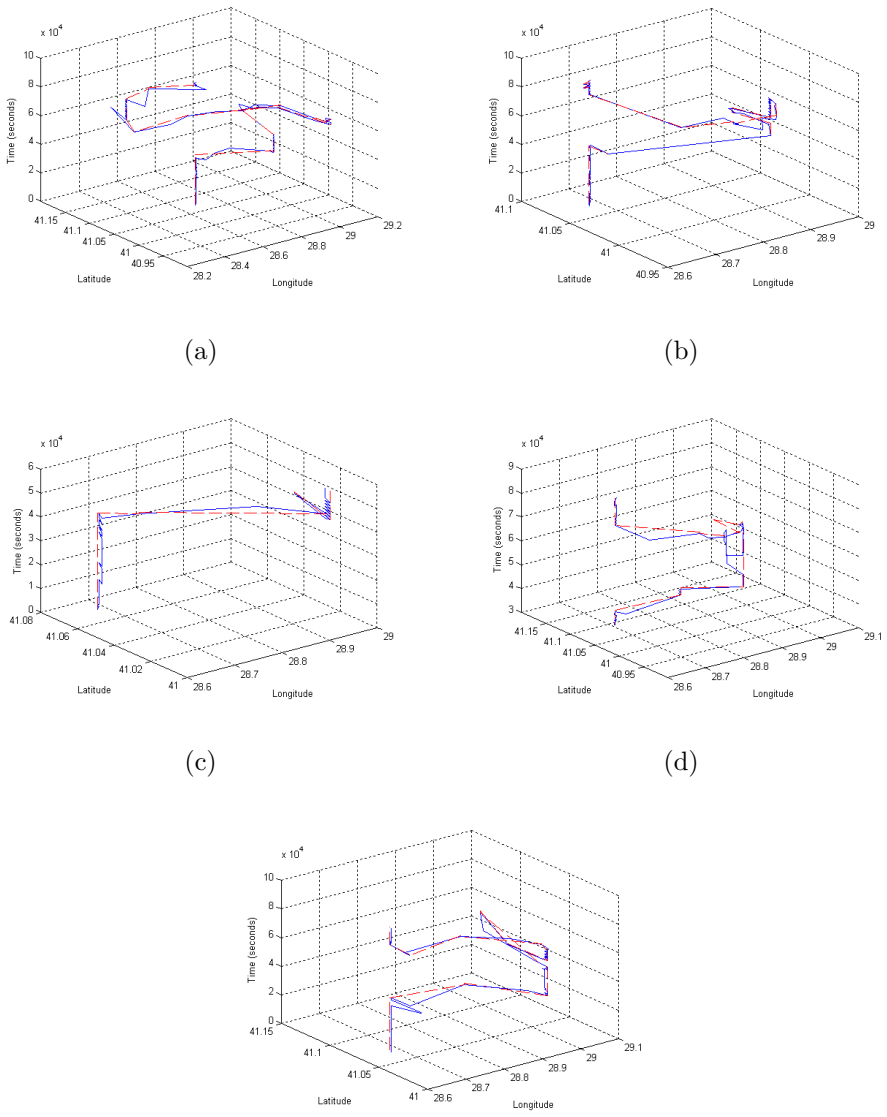


(a) (b)

(c) (d)

(e)

Figure A.1. Comparison of the predicted path with the actual path of User 5

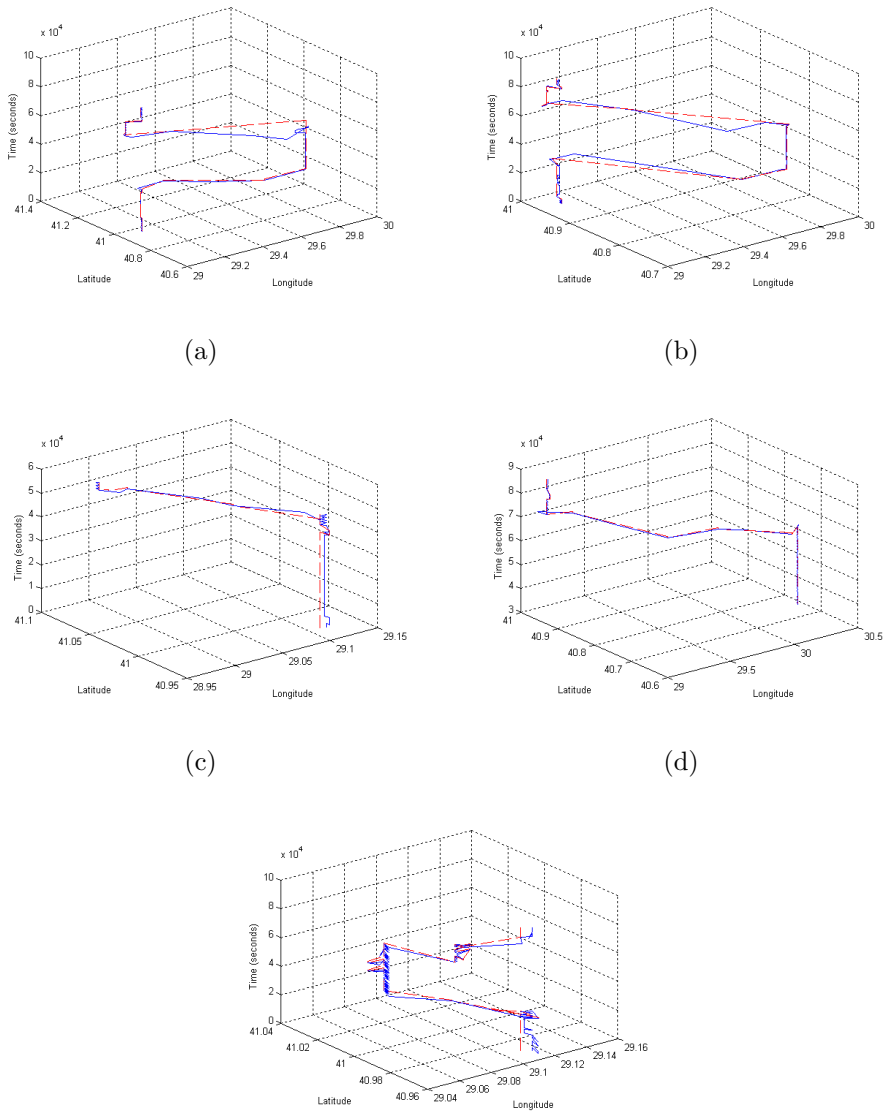


(a) (b)

(c) (d)

(e)

Figure A.2. Comparison of the predicted path with the actual path of User 6



(a) (b)

(c) (d)

(e)

Figure A.3. Comparison of the predicted path with the actual path of User 13

REFERENCES

1. Ashbrook, D. and T. Starner, "Using GPS to learn significant locations and predict movement across multiple users", *Journal Personal and Ubiquitous Computing*, Vol. 7, No. 5, pp. 275 - 286, October 2003

2. Wang, S., J. Min, B. K. Yi, "Location Based Services for Mobiles: Technologies and Standards", *IEEE International Conference on Communication (ICC)*, Beijing, China, June 2008

3. Schiller, J., Mobile Communications, 2nd Ed., Addison Wesley, 2003.

4. Varshavsky, A., "Are GSM Phones THE Solution for Localization?", *7th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA'06)*, pp. 34-42, 2006.

5. Spirito, M.A., A.G. Mattioli, "On the hyperbolic positioning of GSM mobile stations", *International Symposium on Signals, Systems, and Electronics*, pp. 173 177, 29 Sep-2 Oct 1998

6. Mangold, S., S. Kyriazakos, "Applying Pattern Recongition Techniques based on Hidden Markov Models for Vehicular Position Location in Cellular Networks," *Vehicular Technology Conference (VTC 1999 - Fall. IEEE VTS 50th)*, Vol. 2, pp. 780-784, 1999.

7. Wölfle, G., R. Hoppe, D. Zimmermann, and F.M. Landstorfer, "Enhanced Localization Technique within Urban and Indoor Environments based on Accurate and Fast Propagation Models", *European Wireless 2002*, Firence (Italy), Feb. 2002

8. Kunczier, H., H. Anegg, "Enhanced cell ID based terminal location for urban area location based applications", *IEEE Consumer Communications and Networking Conference, CCNC 2004*, Vol. 5, Issue 8, pp. 595 599, Jan. 2004.

9. Ferris, B., D. Hahnel, and D. Fox. ,"Gaussian processes for signal strength-based location estimation", *In Proc. of Robotics Science and Systems Conference*, 2006.

10. Schwaighofer, A., M. Grigoras, V. Tresp, and C. Hoffmann. "GPPS: A Gaussian process positioning system for cellular networks", *In Advances in Neural Information Processing Systems (NIPS)*, 2003.

11. Services Group (SerG) of the GSM Association, SE.23 Permanent Reference Document on Location Based Services, http://www.gsmworld.com/documents/lbs/se23.pdf

12. Ma, S., S. Tang, D. Yang, T. Wang and J. Han, "Combining clustering with moving sequential patterns mining: A novel and efficient technique", *Book: Advances in Knowledge Discovery and Data Mining*, Vol. 3056/2004, pp. 419 - 423, April 22, 2004

13. Hung, C. C., W. C. Peng and J. L. Huang, "Exploring regression for mining user moving patterns in a mobile computing system", *Proceedings of the 2005 Conference on High Performance Computing and Communications (HPCC-05)*, pp. 878-887, Capri-Sorrento Penisular, Italy, September 21-24, 2005.

14. Yavaş, G., D. Katsaros, Ö. Ulusoy and Y. Manolopoulos, "A data mining approach for location prediction in mobile environments", *Data & Knowledge Engineering 54*, pp. 121-146, 2005

15. Peng, W. C. and M. S. Chen, "Developing data allocation schemes by incremental mining of user moving patterns in a mobile computing system", *IEEE Transaction on Knowledge and Data Engineering*, Vol. 15, No. 1, pp. 70-85, January/February 2003

16. Liu, T., P. Bahl and I. Chlamtac, "Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks", *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 6, pp. 922-936, August 1998

17. Shivakumar, N., J. Jannink, and J. Widom, "Per-user Profile Replication in Mobile Environments: Algorithms Analysis and Simulation Result", *ACM/Baltzer Journal of Mobile Networks and Applications*, 2(2), pp. 129-140, 1997

18. Marmasse, N., C. Schmandt, "A User-Centered Location Model", *Personal and Ubiquitous Computing*, Vol. 6, pp. 318321, 2002

19. Samaan, N. and A. Karmouch, "A mobility prediction architecture based on contextual knowledge and spatial conceptual maps", *IEEE Transactions on Mobile Computing*, Vol. 4, No. 6, pp. 537-551, November/December 2005

20. Su, Z., Q. Yang, Y. Lu, H. Zhang, "An Evidence-Based Mobility Prediction Agent Architecture", *Proceedings of the First International Conference on Web Information Systems Engineering*, Vol. 1, pp. 214 221, 2000.

21. Chakraborty, G., "Efficient location management by movement prediction of the mobile host", *Proceedings of the 4th International Workshop on Distributed Computing, Mobile and Wireless Computing*, pp. 142 - 153, 2002

22. Bhattacharya, A. and Sajal K. Das, "LeZi-Update: An Information-Theoretic Approach to Track Mobile Users in PCS Networks", *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom99)*, pp. 112, 1999.

23. Bhattacharya, A. and Sajal K. Das, "LeZi-Update: An Information-Theoretic Framework for Personal Mobility Tracking in PCS Networks", *ACM/Kluwer Journal on Wireless Networks (Special Issue on selected Mobicom99 papers, Guest Eds: T. Imilienski and M. Steenstrup)*, Vol. 8, No. 23, pp. 121135, Mar-May 2002.

24. Quintero, A., "A User Pattern Learning Strategy for Managing Users Mobility in UMTS Networks", *IEEE Transactions on Mobile Computing*, Vol. 4, No. 6, pp. 552 - 566, November/December 2005.

25. Ghosh, R.K., S. Aggarwala, H. Mishra, A. Sharma and H. Mohanty, "Tracking of

Mobile Terminals Using Subscriber Mobility Pattern with Time-Bound Self Purging Indicators and Regional Route Maps", *Proceedings 7th international workshop Kharagpur (IWDC 2005)*, Vol. 3741, pp. 512-523, India, 2005.

26. Liang, B., Z.J. Haas, "Predictive Distance-Based Mobility Management for Multidimensional PCS Networks", *IEEE/ACM Transactions on Networking*, Vol. 11, No. 5, pp. 718-732, October 2003.

27. Cayirci, E. and I.F. Akyildiz, "User Mobility Pattern Scheme for Location Update and Paging in Wireless Systems", *IEEE Trans. Mobile Computing*, Vol. 1, No. 3, pp. 236-247, July-Sept. 2002.

28. Rissamen, J. and G.G. Langdon, "Universal modeling and coding", *IEEE Transactions on Information Theory*, 27(l), pp. 1223, January 1981.

29. Ghosh, R.K., S.K. Rayanchu, H. Mohanty, "Location management by movement prediction using mobility patterns and regional route maps", *In: Proceedings of IWDC 2003*, LNCS. Vol. 2981, pp. 153162, 2003.

30. Lee, K. C. K., H. V. Leong and A. Si., "Approximating object location for moving object database", *Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops (ICDCSW03)*, pp. 402- 407, 19-22 May 2003

31. Brakatsoulas, S., D. Pfoser, N. Tryfona, "Modeling, Storing and Mining Moving Object Databases", *International Database Engineering and Applications Symposium (IDEAS'04)*, pp. 68-77, 2004.

32. Wolfson, O., H. Yin, "Accuracy and Resource Consumption in Tracking and Location Prediction", *Proceedings of the 33rd international conference on Very large data bases Vienna*, pp. 1362-1365, Austria 2007.

33. Chan, J., B. Landfeldt, A. Seneviratne and P. Sookavatana, "Integrating Mobility Prediction and Resource Pre-allocation into a Home-Proxy Based Wireless Inter-

net Framework", *Proceedings IEEE International Conference on Networks, 2000. (ICON 2000)*, pp. 18 23, 2000.

34. Chan, J., A. Seneviratne, "A Practical User Mobility Prediction Algorithm for Supporting Adaptive QoS in Wireless Networks", *Proceedings IEEE International Conference on Networks (ICON '99)*, pp. 104-111, 1999.

35. Liu, G. and G. Maguire Jr., "A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communications", *ACM/Baltzer MONET*, 1(2), pp. 113-121, 1996.

36. Bharghavan, V. and M. Jayanth, "Profile-based Next-cell Prediction in Indoor Wireless LAN", *In Proc. IEEESICON97*, Apr. 1997, available at http://shiva.crhc.uiuc.edu/publications.html

37. Jayasuriya, A., J. Asenstorfer, "Mobility Prediction Model for Cellular Networks Based on Observed Traffic Patterns", *Proceedings IASTED International Conference, Wireless and Optical Communications (WOC 2002)*, pp. 386-391, Banff, Canada, July 2002.

38. Anagnostopoulos, T., C.B. Anagnostopoulos, S.Hadjiefthymiades, A. Kalousis, M. Kyriakakos, "Path Prediction through Data Mining", *IEEE International Conference on Pervasive Services*, pp. 128 135, 2007.

39. Kyriakakos, M., S.Hadjiefthymiades, N.Fragkiadakis, and L. Merakos, "Enhanced Path Prediction for Network Resource Management in Wireless LANs", *IEEE Wireless Communications Magazine*, Special issue on "The Evolution of Wireless LANs and PANs", Vol. 10, No. 6, pp. 62-69, 2003.

40. D. Levine, I. Akyildiz and M. Naghshineh, "A Resource Estimation and Call Admission Algorithm for Wireless Multimedia Networks Using the Shadow Cluster Concept", *IEEE/ACM Trans. on Networking*, 5(1), pp. 1-12, Feb. 1997.

41. Jannink, J., D. Lam, N. Shivakumar, J. Widom and D. Cox, "Efficient and Flexible Location Management Techniques for Wireless Communication System", *ACM/Baltzer Wireless Networks*, 3(5), pp. 361-374, 1997.

42. Liu, G., G. Maguire, "A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communications", *Mobile Networks and Applications*, Vol. 1, Issue 2, pp. 113 121, 1996.

43. Tseng, Vincent S., Kawuu W. Lin, "Efficient Mining and Prediction of User Behavior Patterns in Mobile Web Systems", *Information and Software Technology*, Vol. 48, Issue 6, pp. 357-369, June 2006.

44. Chellappa-Doss, R., A. Jennings, N. Shenoy, "User Mobility Prediction in Hybrid and Ad Hoc Wireless Networks," *Australian Telecommunications Networks and Applications Conference*, 2003.

45. Chellappa, R., A. Jennings, N. Shenoy, "A Sectorized Mobility Prediction Algorithm for Wireless Networks", *Proceedings of the 2003 International Conference on Information and Communication Technologies (ICT 2003)*, pp.86- 92 , Thailand, April 8-11 2003.

46. Su, W., S.J. Lee, M. Gerla, "Mobility Prediction in Wireless Networks", *21st Century Military Communications Conference Proceedings*, Vol. 1, pp. 491 495, 2000.

47. Capka, J., R. Boutaba, "Mobility Prediction in Wireless Networks Using Neural Networks", *IFIP/IEEE International conference on management of multimedia networks and services*, San Diego CA, Vol. 3271, No. 3, pp. 320-333, 2004.

48. Michaelis, S., C. Wietfeld, "Comparison of User Mobility Pattern Prediction Algorithms to increase Handover Trigger Accuracy", *IEEE 63rd Vehicular Technology Conference*, Vol. 2, pp. 952-956, Spring 2006.

49. Quinlan, J.R., "C4.5: Programs for machine learning (Morgan Kaufmann Series for Machine Learning), San Mateo 1993.

50. Bhattacharya, A., S. K. Das, "LeZi-Update: An Information- Theoretic Approach to Track Mobile Users in PCS Networks", *Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, pp. 1- 12, Seattle, 1999.

51. Witten, I. H., E. Frank, "Data Mining: Practical machine learning tools with Java implementations (The Morgan Kaufmann Series in Data Management Systems)", San Francisco, 2000.

52. Poon, T. W.T., E. Chan, "Traffic Management in Wireless ATM Network Using a Hierarchical Neural-Network Based Prediction Algorithm", *Proceedings of the international Conference on Computers and their Applications*, ICSA 2000, pp 393-395.

53. Kyriakakos, M., N. Frangiadakis , S. Hadjiefthymiades and L. Merakos, "RMPG: a Realistic Mobility Pattern Generator for the Performance Assestment of Mobility Functions", *Simulation Modelling Practice and Theory*, Vol. 2, Issue1, pp. 1-13, April 2004.

54. Greis, M., *The Network Simulator*, http://www.isi.edu/nsnam/ns/

55. Brinkhoff, T., "Generating Network-Based Moving Objects", *Proceedings of 12th International Conference on Scientific and Statistical Database Management*, Berlin, Germany, July 26-28, 2000, IEEE Computer Society Press

56. Brinkhoff, T., "A Framework for Generating Network-Based Moving Objects", GeoInformatica, Vol. 6, No. 2, pp. 153-180, 2002

57. Sinnott, R.W., "Virtues of the Haversine", *Sky and Telescope*, Vol. 68, No. 2, pp. 159, 1984