Mining Frequent Trajectories of Moving Objects for Location Prediction

Mikołaj Morzy

Institute of Computing Science
Poznań University of Technology
Piotrowo 2, 60-965 Poznań, Poland
Mikolaj.Morzy@put.poznan.pl

Abstract. Advances in wireless and mobile technology flood us with amounts of moving object data that preclude all means of manual data processing. The volume of data gathered from position sensors of mobile phones, PDAs, or vehicles, defies human ability to analyze the stream of input data. On the other hand, vast amounts of gathered data hide interesting and valuable knowledge patterns describing the behavior of moving objects. Thus, new algorithms for mining moving object data are required to unearth this knowledge. An important function of the mobile objects management system is the prediction of the unknown location of an object. In this paper we introduce a data mining approach to the problem of predicting the location of a moving object. We mine the database of moving object locations to discover frequent trajectories and movement rules. Then, we match the trajectory of a moving object with the database of movement rules to build a probabilistic model of object location. Experimental evaluation of the proposal reveals prediction accuracy close to 80%. Our original contribution includes the elaboration on the location prediction model, the design of an efficient mining algorithm, introduction of movement rule matching strategies, and a thorough experimental evaluation of the proposed model.

1 Introduction

Moving objects are ubiquitous. Portable devices, personal digital assistants, mobile phones, laptop computers are quickly becoming affordable, aggressively entering the market. This trend is parallel to the widespread adoption of wireless communication standards, such as GPRS, Bluetooth, or Wi-Fi networks. Recent advances in positioning technology compel manufacturers to equip their devices with positioning sensors that utilize Global Positioning System (GPS) to provide accurate location of a device. Accurate positioning of mobile devices paves the way for the deployment of location-based services and applications. Examples of location-based services include location-aware information retrieval, emergency services, location-based billing, or tracking of moving objects. It is important to note that location-based services are not limited to mobile devices, such as mobile phones, PDAs or laptops; these services can be successfully deployed for other types of moving objects, e.g., vehicles or even humans. In order to fully exploit the possibilities offered by location-aware services, it is crucial to determine the current position of a moving object at any given point in time.

Typically, a moving object is equipped with a transmitting device that periodically signals its position to the serving wireless carrier. Between position disclosures the exact location of a moving object remains unknown and can be determined only approximately. Unfortunately, the periodicity of position acknowledgments can be interrupted by several factors. For instance, the failure can be caused by power supply shortage of a moving object. Positioning systems have known limitations that can result in communication breakdown. Signal congestions, signal losses due to natural phenomena, or the existence of urban canyons lead to temporal unavailability of a moving object positioning information. Whenever the location of a moving object is unknown, a robust method of possible location prediction of a moving object is required.

Predicting the location of a moving object can be a difficult task. Firstly, the sheer amount of data to be processed precludes using traditional prediction methods known from machine learning domain. The stream of data generated by positioning sensors of thousands of moving objects requires new, robust and reliable data mining processing methods. The location prediction mechanism must allow for fast scoring of possible moving object location. The method must work online and should not require expensive computations. Furthermore, the performance of the prediction method should not degrade significantly with the increase of the number of moving objects. We also require that, given the prediction accuracy is satisfactory and does not drop below a given threshold, the prediction method should favor prediction speed over prediction accuracy. We believe that this feature is crucial for the development of successful location-based services. The success of a location-based service depends on whether the service is delivered to a particular object at a particular location and on particular time. If objects move quickly and change their location often, then the speed of computation must allow to deliver the service while the object still occupies a relevant location. For instance, complex models of movement area topology and movement interactions between objects may produce accurate results, but their computational complexity is unfeasible in mobile environment. Similarly, prediction methods based on simulation strongly depend on numerous input parameters that affect the quality of the resulting movement model. The cost of computing the model can be prohibitively high and the model itself may not scale well with the number of moving objects.

Another important drawback of currently used prediction methods is the fact that most of these methods do not utilize historical data. The raw data collected from moving objects hide useful knowledge patterns that describe typical behavior of moving objects. In particular, trajectories frequently followed by moving objects can be mined to discover movement patterns. Movement patterns, represented in the form of human-readable movement rules can be used to describe and predict the movement of objects.

Data mining techniques have been long considered inappropriate and unsuitable for online location prediction due to long processing times and computational expensiveness of these techniques. In this paper we prove that this assumption is entirely incorrect and that data mining techniques can be successfully used for location prediction. We build a probabilistic model of an unknown position of a moving object based on historical data collected from other objects moving on the same area. We mine logs of historical position acknowledgments to discover frequent trajectories of objects representing popular movement routes, and then we transform frequent trajectories into movement rules. In order to predict the location of a moving object, for which only a part of its movement history is known, we score the movement history of the object against the database of movement rules to find possible locations of the object. For each possible location we compute the probability of prediction correctness based on the support and confidence of discovered movement rules. Our method is fast and reliable. Frequent trajectories and movement rules are discovered periodically in an offline manner. The scoring process is performed online. Our experiments show that the scoring process can be performed within milliseconds. The presented method is independent of the movement area topology and scales well with the number of moving objects. The idea of using movement rules for location prediction was first presented in [15]. The work presented in this paper continues and extends our previous initial findings in a number of ways. The original contribution of this paper includes:

– refinement of the frequent trajectory model,
– design of an efficient *Traj-PrefixSpan* algorithm for mining frequent trajectories,
– modification of the *FP-Tree* index structure for fast lookup of trajectories,
– experimental evaluation of the proposal.

The paper is organized as follows. Section 2 presents the related work on the subject. In section 3 we introduce notions and definitions used throughout the paper. The *Traj-PrefixSpan* algorithm and frequent trajectory matching methods are presented in section 4. Section 5 contains the results of the experimental evaluation of our proposal. The paper concludes in section 6 with a brief summary.

2 Related Work

Both spatial data mining and mobile computing domains attract significant research efforts. The first proposal for spatial data mining has been formulated in [11]. Since then, many algorithms for spatial data mining have been proposed [5]. Authors in [6] introduce a spatial index for mining spatial trends using relations of topology, distance, and direction. A comprehensive overview of current issues and problems in spatial and spatio-temporal databases can be found in [7], and recent advances in spatio-temporal indexing are presented in [14]. However, the problem of mining trajectories of moving objects in spatial databases remained almost unchallenged until recently. Examples of advances in this field include the idea of similar trajectory clustering [12] and the proposal to use periodic trajectory patterns for location prediction [13]. The aforementioned works extend basic frameworks of periodic sequential patterns [8] and frequent sequential patterns [1].

An interesting area of research proposed recently focuses on moving object databases [2]. In [17] authors consider the effect of data indeterminacy and fuzziness on moving objects analysis. According to the authors, an inherent uncertainty of moving objects data influences attribute values, relations, timestamps, and time intervals. Advances in mobile object databases can be best illustrated by the development of the Path-Finder system, a prototype moving object database capable of mining moving object data. The idea of using floating car data of an individual moving object to describe movement patterns of a set of objects is presented in [3].

Several proposals come from mobile computing domain. Most notably, tracking of moving objects resulted in many interesting methods for location prediction. Authors in [10] present a probabilistic model of possible moving object trajectories based on road network topology. The solution presented in [21] advocates to use time-series analysis along with simulation of traveling speed of moving objects to determine possible trajectory of an object. A modification of this approach consisting in using non-linear functions for movement modeling is presented in [19]. A movement model that employs recursive motion functions mimicking the behavior of objects with unknown motion patterns is introduced in [18]. Another complex model with accuracy guarantees is presented in [20]. Recently, [22] consider predicting location in presence of uncertain position signals from moving objects. The authors present a *min-max* property that forms the basis for their *TrajPattern* algorithm for mining movement sequences of moving objects.

3 Definitions

Given a database of moving object locations, where the movement of objects is constrained to a specified area A. Let $O = \{o_1, \ldots, o_i\}$ be the set of moving objects. Let p denote the position of a moving object w.r.t. a system of coordinates W, $p \in W$. The *path* $P = (p_1, \ldots, p_n)$ is an ordered n-tuple of consecutive positions of a moving object. Unfortunately, the domain of position coordinates is continuous and the granularity level of raw data is very low. Therefore, any pattern discovered from raw data cannot be generalized. To overcome this problem we choose to transform original paths of moving objects into trajectories expressed on a coarser level. The *net* divides the two-dimensional movement area A into a set of rectangular regions of fixed size. We refer to a single rectangular region as a *cell*. Each cell has four *edges*. Cells form a two-dimensional matrix covering the entire area A, so each cell is uniquely identified by discrete coordinates $\langle i, j\rangle$ describing the position of the cell in the matrix. A moving object always occupies a single cell at any given point in time. When moving, an object crosses edges between neighboring cells. Each edge can be traversed in two directions, vertical edges can be traversed eastwards and westwards, whereas horizontal edges can be traversed northwards and southwards.

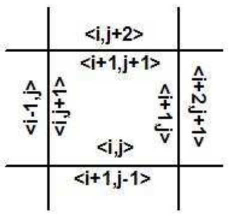

Fig. 1. Edge enumeration

Figure 1 presents the enumeration scheme of edges of the cell $\langle i, j \rangle$ used by our algorithm. Intuitively, the enumeration scheme preserves the locality of edges and allows for a fast lookup of all edges adjacent to a given edge. By an adjacent edge we mean an edge that can be traversed next after traversing a given edge. Each edge receives two sets of coordinates that are relative to its cell coordinates. The two sets of coordinates represent two possible directions of edge traversal. For instance, consider an object occupying the cell $\langle 2, 4 \rangle$. When the object moves northwards, it traverses an edge labeled $\langle 3, 5 \rangle$. The same edge, when traversed southwards, is identified as the edge $\langle 2, 6 \rangle$. The reason for this enumeration scheme is straightforward. An edge cannot have a single coordinate, because the set of possible adjacent edges depends on the direction of traversal. We have also considered other enumeration schemes, such as Hilbert curve or z-ordering. The main advantage of the presented edge enumeration scheme is the fact that any two neighboring edges differ by at most 2 on a dimension. In addition, any two edges within a single cell differ by at most 1 on any dimension.

Each path P_i of a moving object o_i can be unambiguously represented as a sequence of traversed edges. A *trajectory* of an object o_i is defined as an ordered tuple $R_i = (E_1, E_2, \ldots, E_n)$ of edges traversed by the path P_i. The *length* of a trajectory R_i, denoted $length(R_i)$, is the number of edges constituting the trajectory R_i. We refer to a trajectory of the length n as n-trajectory. We say that the trajectory $X = (X_1, X_2, \ldots, X_m)$ is contained in the trajectory $Y = (Y_1, Y_2, \ldots, Y_n)$, denoted $X \subseteq Y$, if there exist $i_1 < i_2 < \ldots < i_m$ such that $X_1 = Y_{i_1}$, $X_2 = Y_{i_2}$, …, $X_m = Y_{i_m}$. A trajectory X is *maximal* if it is not contained in any other trajectory. We say that a trajectory Y *supports* a trajectory X if $X \subseteq Y$. The *concatenation* Z of trajectories $X = (X_1, X_2, \ldots, X_m)$ and $Y = (Y_1, Y_2, \ldots, Y_n)$, denoted $Z = X \otimes Y$, is the trajectory $Z = (X_1, X_2, \ldots, X_m, Y_1, Y_2, \ldots, Y_n)$. Given a database of trajectories $D_T = \{R_1, \ldots, R_q\}$. The *support* of a trajectory R_i is the percentage of trajectories in D_T that support the trajectory R_i.

$$support(R_i) = \frac{|\{R_j \in D_T : R_i \subseteq R_j\}|}{|D_T|}$$

A trajectory R_i is *frequent* if its support exceeds user-defined threshold of minimum support, denoted *minsup*. Given a trajectory $R_i = (E_1, E_2, \ldots, E_n)$. The *tail* of the trajectory R_i, denoted $tail(R_i, m)$, is the trajectory $T_i = (E_1, E_2, \ldots, E_m)$. The *head* of the trajectory R_i, denoted $head(R_i, m)$, is the trajectory $H_i = (E_{m+1}, E_{m+2}, \ldots, E_n)$. Concatenation of the tail and head yields the original trajectory, i.e., $tail(R_i, m) \otimes head(R_i, m) = R_i$.

Frequent trajectories are transformed into movement rules. A movement rule is an expression of the form $T_i \Rightarrow H_i$ where T_i and H_i are frequent adjacent trajectories and $T_i \otimes H_i$ is a frequent trajectory. The trajectory T_i is called the tail of the rule, the trajectory H_i is called the head of the rule. Contrary to the popular formulation from association rule mining, we do not require the tail and the head of a rule to be disjunct. For instance, an object may traverse edges E_i, E_j, E_k and then make a U-turn to go back through edges E_k, E_j. Thus, the same edge may appear both in the tail and the head of a rule. However, this difference does not affect the definition of statistical measures applied to movement rules, namely, support and confidence.

The *support* of the movement rule $T_i \Rightarrow H_i$ is defined as the support of $T_i \otimes H_i$,

$$support\,(T_i \Rightarrow H_i) = \frac{|T_j \in D_T : T_j \supseteq (T_i \otimes H_i)|}{|D_T|}$$

The *confidence* of the movement rule $T_i \Rightarrow H_i$ is the conditional probability of H_i given T_i,

$$confidence\,(T_i \Rightarrow H_i) = P\,(H_i|T_i) = \frac{support\,(T_i \otimes H_i)}{support\,(T_i)}$$

4 Proposed Solution

Formally, the location prediction problem can be decomposed into two subproblems:

– discover movement rules with support and confidence greater than user-defined thresholds of *minsup* and *minconf*, respectively,
– match movement rules against the trajectory of a moving object for which the current location is to be determined.

In section 4.1 we present the *Traj-PrefixSpan* algorithm that aims at efficient discovery of frequent trajectories and movement rules. Section 4.2 describes the modified *FP-Tree* index structure. In section 4.3 we introduce three matching strategies for movement rules.

4.1 Traj-PrefixSpan Algorithm

The algorithm presented in this section is a modification of a well-known *PrefixSpan* algorithm [16]. The difference consists in the fact that, contrary to the original formulation, we do not allow multiple edges as elements of the sequence (each element of a sequence is always a single edge). In addition, each sequence is grown only using adjacent edges, and not arbitrary sequence elements. The following description presents an overview of the *PrefixSpan* algorithm, already augmented to handle trajectories.

Given a trajectory $X = (X_1, X_2, \ldots, X_n)$, the *prefix* of the trajectory X is a trajectory $Y = (Y_1, Y_2, \ldots, Y_m)$, $m \leq n$, such that $Y_i = X_i$ for $i = 1, 2, \ldots, m-1$. The *projection* of the trajectory X over prefix Y is a sub-trajectory X' of the trajectory X, such that Y is the prefix of X' and no trajectory X'' exists such that Y is the prefix of X'', X'' is the sub-trajectory of X, and $X'' \neq X'$.

Let $X' = (X_1, \ldots, X_n)$ be a projection of X over $Y = (Y_1, \ldots, Y_{m-1}, X_m)$. The trajectory $Z = (X_{m+1}, \ldots, X_n)$ is a postfix of X over the prefix Y, denoted $Z = X/Y$. In other words, for a given prefix Y and a given postfix Z, $X = Y \otimes Z$.

Let Y be a frequent trajectory in the database of trajectories D_T. An Y*-projected trajectory database*, denoted by $D_{T/Y}$, is the set of all postfixes of trajectories in D_T over the prefix Y. Let X be a trajectory with the prefix Y. The *support count* of X in Y-projected trajectory database, denoted by $support_{D_{T/Y}}(X)$, is the number of trajectories Z in $D_{T/Y}$, such that X is a sub-trajectory of $Y \otimes Z$.

```
procedure $TrajPrefixSpan(Y, l, D_{T/Y})$
 1: scan $D_{T/Y}$ to find edges $e$ such, that
        if $(l > 0)$ then $e$ is adjacent to the last edge of $Y$
        else $Y$ can be extended by $e$ to form a frequent trajectory
 2: foreach edge $e$ create $Y' = Y \otimes e$
 3: foreach $Y'$ build $D_{T/Y'}$
 4: run $TrajPrefixSpan(Y', l + 1, D_{T/Y'})$
end procedure
```

Fig. 2. Traj-PrefixSpan algorithm

Traj-PrefixSpan algorithm consists of three phases. In the first phase the algorithm performs a full scan of the trajectory database D_T to discover all frequent 1-trajectories. In the second phase each frequent 1-trajectory Y is used to create an Y-projected trajectory database. Every pattern contained in an Y-projected trajectory database must have the prefix Y. The third phase of the algorithm consists in recursive generation of further Y'-projected trajectory databases from frequent trajectories Y' found in projections. The pseudocode of the algorithm is presented in Figure 2. The initial call is $TrajPrefixSpan(<>, 0, D_T)$.

4.2 FP-Tree

The physical indexing structure used in our algorithm is a slightly modified *FP-Tree* [9]. The main change consists in storing sequences of elements (as opposed to sets of elements), and allowing a bi-directional traversal of the tree. *FP-Tree* is an undirected acyclic graph with a single root node and several labeled internal nodes. The root of the tree is labeled *null*, and internal nodes are labeled with edge numbers they represent. Each internal node of the tree has a label, a counter representing the support of a sequence from the root to the given node, and a pointer to the next node with the same label (or a *null* pointer if no such node exists). In addition, the index contains a header table with edges ordered by their support and pointers to the first occurrence of an edge within the *FP-Tree*. The tree is constructed during the execution of the *Traj-PrefixSpan* algorithm by pattern growth. Each frequent trajectory discovered by the *Traj-PrefixSpan* algorithm is inserted into *FP-Tree* index for fast lookup. After the frequent trajectory discovery process finishes, the *FP-Tree* contains all frequent trajectories discovered in the database. Generation of movement rules is a straightforward task. For each frequent n-trajectory $X = (X_1, X_2, \ldots, X_n)$, $(n-1)$ movement rules can be generated by splitting the trajectory in every possible place, $T_1 \Rightarrow H_1, T_2 \Rightarrow H_2, \ldots, T_{n-1} \Rightarrow H_{n-1}$.

4.3 Matching strategies

After frequent trajectories have been found and stored in the *FP-Tree*, they can be used to predict the unknown location of a moving object. For each moving object its known trajectory has to be compared with movement rules generated from frequent trajectories. In the next sections we introduce three matching strategies for scoring a partial trajectory of a moving object with the database of movement rules. In all examples let $X = (X_1, X_2, \ldots, X_m)$ be a partial trajectory of a moving object, for which we are seeking its most probable location. For a given partial trajectory X the set of all matched movement rules is denoted by L_X.

Whole Matcher The *Whole Matcher* strategy consists in finding all movement rules $T_i \Rightarrow H_i$ such, that $X = T_i$ (i.e., the tail of the rule entirely covers the partial trajectory X). The head H_i can be used as a prediction of a possible location of a moving object. The probability that a moving object follows H_i is given by $\mathit{confidence}\,(T_i \Rightarrow H_i)$. The Whole Matcher strategy yields accurate results, but disallows any deviations of matched rules from the partial trajectory X. Furthermore, in case of long partial trajectories, the Whole Matcher strategy may fail to find a matching movement rule.

Last Matcher The *Last Matcher* strategy discards all information from the partial trajectory X except for the last traversed edge X_m. The strategy finds all movement rules $T_i \Rightarrow H_i$ such, that $X_m = T_i$. The result of the strategy is the list of edges (movement rule heads H_i) ordered by descending values of $\mathit{confidence}\,(T_i \Rightarrow H_i)$. The Last Matcher strategy finds matching movement rules even for very short partial trajectories, but the predictions in L_X are less reliable, because they ignore the movement history of a moving object.

Longest Last Matcher The *Longest Last Matcher* strategy is a compromise between the two aforementioned strategies. For a given partial trajectory X it finds all movement rules $T_i \Rightarrow H_i$ such, that T_i covers a part of the partial trajectory X, i.e., there exists j, $1 \leq j < m$ such, that $T_i = \mathit{head}(X, j)$. The strategy outputs, as the result, the movement rule heads H_i weighted by the relative coverage of the partial trajectory X. For a given movement rule $T_i \Rightarrow H_i$ the strength of the prediction is defined as $\mathit{confidence}\,(T_i \Rightarrow H_i) * \frac{\mathit{length}(T_i)}{\mathit{length}(X)}$. Edges contained in L_X are ordered according to the descending value of the prediction strength.

5 Experiments

In this section we report on the results of the experimental evaluation of the proposed approach. All experiments were conducted on a PC equipped with AMD Athlon XP 2500+ CPU, 521 MB RAM, and a SATA hard drive running under Windows XP SP2 Home Edition. Algorithms and the front-end application were implemented in C# and run within Microsoft .NET 2.0 platform. Synthetic datasets were generated using Network-based Generator of Moving Objects by T.Brinkhoff [4]. Experiments were conducted using the map of Oldenburg. The number of moving objects varied from 1 000 to 10 000, the number of classes of moving objects was set to 10, and the number of time units in each experiment was 200. We set the maximum velocity of moving objects to 50, locations of objects were registered using *PositionReporter* method. All results reported in this section are averaged over 30 different instances of datasets. The experiments measure: the time of mining frequent trajectories, the number of discovered frequent trajectories, the time of matching a partial trajectory with the database of moving rules, and the quality of location prediction.

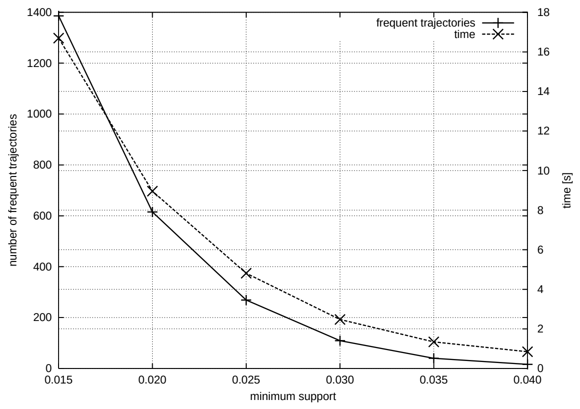

Fig. 3. Minimum support

Figure 3 shows the number of frequent trajectories (depicted on the left-hand side axis of ordinates) and the time of mining frequent trajectories (depicted on the right-hand side axis of ordinates) with respect to the varying value of the *minsup* threshold. Both measured values decrease with the increase of the *minsup* threshold. As can be clearly seen, the correlation between the number of frequent trajectories and the time it takes to mine them is evident. We are pleased to notice that even for low values of *minsup* threshold the algorithm requires less than 20 seconds to complete computations and the number of discovered frequent trajectories remains manageable.

Figure 4 presents the number of frequent trajectories (depicted on the left-hand side axis of ordinates) and the time of mining frequent trajectories (depicted on the right-hand side axis of ordinates) with respect to the varying number of moving objects for a set value of $minsup = 0.025$. Firstly, we notice that the time of mining frequent trajectories is linear w.r.t. the number of moving objects, which is a desirable property of our algorithm. Secondly, we observe a slight decrease in the number of discovered movement rules as the number of moving objects grows (a fivefold increase in the number of moving objects results in a 20% decrease of the number of discovered movement rules). This phenomenon is caused by the fact that a greater number of moving objects is spread more or less uniformly over the movement area, and the *minsup* threshold is expressed as the percentage of the number of all moving objects. Thus, less edges become frequent. For a smaller number of moving objects edges in the center of the city tend to attract more moving objects, and less restrictive *minsup* threshold makes more of these edges frequent, resulting in more movement rules.

Figure 5 shows the number of frequent trajectories (depicted on the left-hand side axis of ordinates) and the time of mining frequent trajectories (depicted on the right-

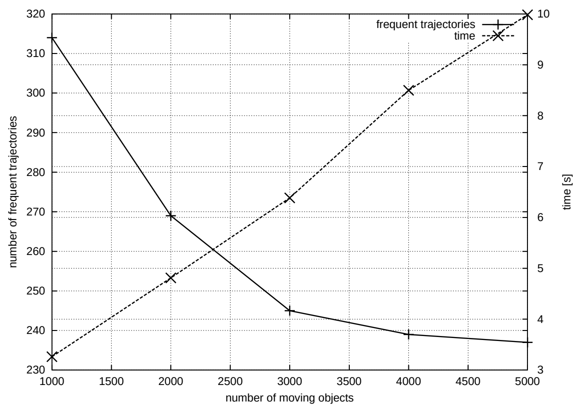

Fig. 4. Number of moving objects

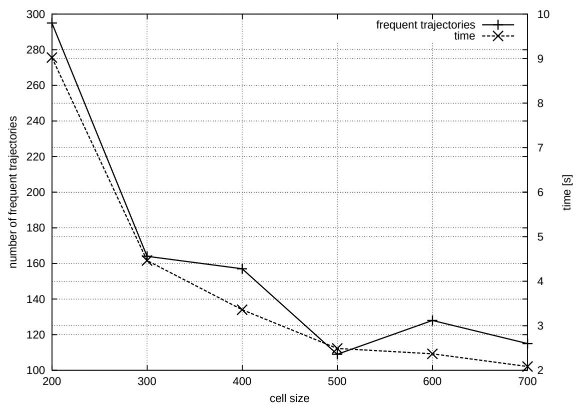

Fig. 5. Cell size

hand side axis of ordinates) with respect to the varying size of an edge cell. The size of a cell is expressed in artificial units. The time of mining steadily decreases with the growth of the cell size. This result is obvious, because larger cells result in less frequent trajectories. On the other hand, the decrease is not linear. For larger cell sizes the number of discovered frequent trajectories is indeed lower. However, discovered frequent trajectories have higher support and tend to be longer, contributing to the overall computation time. The interpretation of the second curve, the number of discovered frequent trajectories, is more tricky. One can notice atypical deviations for cell sizes of 400 and 600 units. These random effects are probably caused by accidental structural influence of larger and smaller cell sizes on areas of intensified traffic. The results presented in Figure 5 emphasize the importance of correct setting of the cell size parameter (e.g., the difference in the number of discovered frequent trajectories is 10 when changing the cell size from 300 to 400 units, and it grows to 40 when changing the cell size from 400 to 500 units). Unfortunately, our model does not permit to choose the optimal value of the cell size parameter other than experimentally.

The next two figures present the results of experiments evaluating the accuracy of prediction of the location using movement rules. These experiments were conducted as follows. First, a database of moving objects was generated using a set of fixed parameters. Then, 50 trajectories were randomly drawn from each database. Each test trajectory was then split into a tail and a head. The tail was used as a partial trajectory, for which future location of an object was to be predicted. Finally, the prediction returned from each matching strategy was compared to the known head of the test trajectory and the quality of prediction was computed. Let $X = (X_1, X_2, \ldots, X_m)$ be a randomly selected trajectory of a moving object, divided into $tail(X, k)$ and $head(X, k)$. The tail is used as a partial trajectory for matching. If the next traversed edge, which is X_{k+1} is not contained in the set of matching strategy answers L_X, then the quality of location prediction $Quality(X, L_X) = 0$. Otherwise, the quality of matching is computed as the probability of traversing X_{k+1} diminished by weighted incorrect predictions from L_X that had prediction strength greater than X_{k+1}, i.e.,

$$Quality(X, L_X) = P(X_{k+1}) * (1 - \sum_{j \leq k : X_j \in L_X} \frac{P(X_j) - P(X_{k+1})}{k+1} \qquad (1)$$

In the above formula we assume that L_X is ordered by the decreasing prediction strength, so stronger predictions have lower indices.

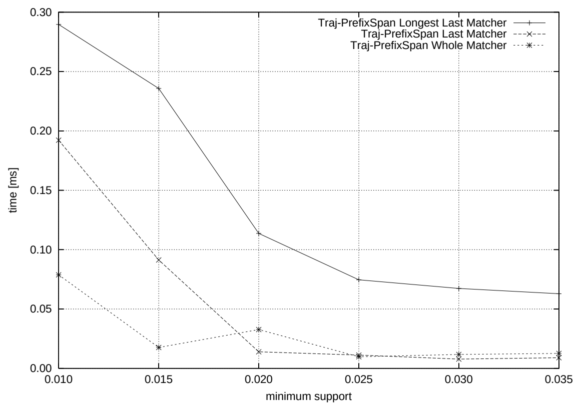

Fig. 6. Prediction time

Figure 6 presents the average time required to match a partial trajectory with the database of movement rules with respect to the varying *minsup* threshold (and, consequently, to the number of discovered movement rules). The *Whole Matcher* and *Last Matcher* strategies perform almost identically, because both strategies can fully utilize the *FP-Tree* index structure. The *Longest Last Matcher* strategy performs slower, because it must traverse a larger part of the *FP-Tree*. Nevertheless, in case of all strategies the matching time is very fast and never exceeds 0.3 ms. We are particularly satisfied with this result, because it supports our thesis that data mining methods can be employed for real-time location prediction.

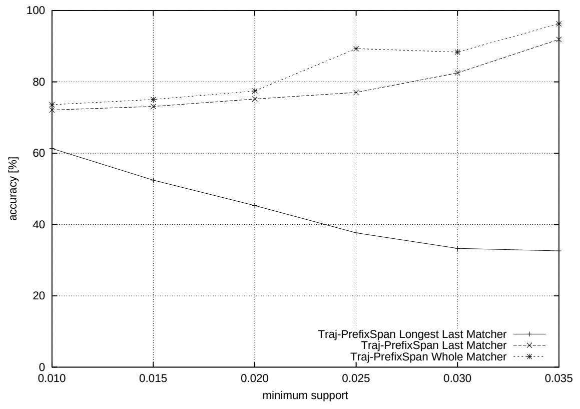

Fig. 7. Quality of prediction

Figure 7 depicts the average quality of prediction as computed by Equation 1. The prediction quality of the *Whole Matcher* and *Last Matcher* strategies reaches even 95% of accuracy for high *minsup* threshold values. For general settings of the *minsup* threshold the accuracy of both methods remains satisfactory between 75% and 85%. It is worth mentioning that the results depicted in the figure are computed according to our formula, which might be too penalizing for the *Longest Last Matcher* strategy, so the presented numbers are somehow biased towards simple matching strategies. The quality achieved by the *Longest Last Matcher* strategy varies from 35% to over 60%. Surprisingly, the quality of prediction increases with the decrease of the *minsup* threshold. This can be explained by the fact that low values of the *minsup* threshold produce more frequent trajectories and more often the correct prediction is placed high in the resulting set L_X. Nevertheless, from the experimental evaluation we conclude that the *Longest Last Matcher* strategy is inferior to the *Whole Matcher* and *Last Matcher* strategies under all conditions.

6 Conclusions

In this paper we have introduced a new data mining model aiming at the efficient prediction of unknown location of moving objects based on movement patterns discovered from raw data. The model represents frequent trajectories of moving objects as movement rules. Movement rules provide a simplification and generalization of a large set of moving objects by transforming original continuous domain of moving object positions into a discretized domain of edges of a superimposed grid. The main thesis of the paper, well proved by conducted experiments, is that data mining techniques can be successfully employed for real-time location prediction in mobile environments. Indeed, while most expensive and burdensome computations (e.g. the discovery of frequent trajectories) can be performed offline and periodically, the online matching of partial trajectories with the database of movement rules is executed very fast. The quality of location prediction is satisfying, but we aim at developing more efficient matching strategies for even better accuracy.

Our future work agenda includes:

- replacing uniform grid cells with differently sized areas that adaptively divide the area of movement based on the density and congestion of moving objects,
- developing new matching strategies,
- including temporal aspects in discovered movement rules,
- including spatial information in movement rules,
- providing more informed decisions to location-based services based on discovered movement rules.

6.1 Acknowledgments

The author wishes to express his gratitude to one of his students, Lukasz Rosikiewicz, who greatly contributed to the implementation of all experiments, whose results are presented above.

References

1. R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE'95, Taipei, Taiwan, March 6-10*, pages 3–14. IEEE Computer Society, 1995.
2. S. Brakatsoulas, D. Pfoser, and N. Tryfona. Modeling, storing and mining moving object databases. In *Proceedings of the 8th International Database Engineering and Applications Symposium IDEAS'2004*, pages 68–77, 2004.
3. S. Brakatsoulas, D. Pfoser, and N. Tryfona. Practical data management techniques for vehicle tracking data. In *Proceedings of the 21st International Conference on Data Engineering ICDE'2005*, pages 324–325, 2005.
4. T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.
5. M. Ester, A. Frommelt, H.-P. Kriegel, and J. Sander. Spatial data mining: Database primitives, algorithms and efficient dbms support. *Data Mininig and Knowledge Discovery*, 4(2/3):193–216, 2000.
6. M. Ester, H.-P. Kriegel, and J. Sander. Knowledge discovery in spatial databases. In *Proceedings of the 23rd German Conference on Artificial Intelligence, KI'99*, pages 61–74, 1999.
7. G. Gidofalvi and T. B. Pedersen. Spatio-temporal rule mining: issues and techniques. In *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery DaWaK'2005*, pages 275–284, 2005.
8. J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *ICDE'99, Sydney, Austrialia, 23-26 March*, pages 106–115. IEEE Computer Society, 1999.
9. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 2000. ACM Press.

10. H. A. Karimi and X. Liu. A predictive location model for location-based services. In *ACM GIS'03, New Orleans, Louisiana, USA, November 7-8*, pages 126–133. ACM, 2003.
11. K. Koperski and J. Han. Discovery of spatial association rules in geographic databases. In *SSD'95, Portland, Maine, August 6-9*, pages 47–66. Springer, 1995.
12. Y. Li, J. Han, and J. Yang. Clustering moving objects. In *ACM SIGKDD'04, Seattle, Washington, USA, August 22-25*, pages 617–622. ACM, 2004.
13. N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *ACM SIGKDD'04, Seattle, Washington, USA, August 22-25*, pages 236–245. ACM, 2004.
14. M. F. Mokbel, T. M. Ghanem, and W. G. Aref. Spatio-temporal access methods. *IEEE Data Engineering Bulletin*, 26(2):40–49, 2003.
15. M. Morzy. Prediction of moving object location based on frequent trajectories. In *Proceedings of the 21st International Symposium on Computer and Information Sciences, ISCIS'06, November 1-3, 2006, Istanbul, Turkey*, volume 4263, pages 583–592. Springer Verlag, 2006.
16. J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE'01, Heidelberg, Germany, April 2-6*, pages 215–224. IEEE Computer Society, 2001.
17. D. Pfoser and N. Tryfona. Capturing fuzziness and uncertainty of spatiotemporal objects. In *Proceedings of the 5th East European Conference on Advances in Databases and INformation Systems ADBIS'2001*, pages 112–126, 2001.
18. Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *ACM SIGMOD'04, Paris, France, June 13-18*, pages 611–622. ACM, 2004.
19. G. Trajcevski, O. Wolfson, B. Xu, and P. Nelson. Real-time traffic updates in moving objects databases. In *DEXA'02, Aix-en-Provence, France, 2-6 September*, pages 698–704. IEEE Computer Society, 2002.
20. O. Wolfson and H. Yin. Accuracy and resource concumption in tracking and location prediction. In *SSTD'03, Santorini, Greece*, pages 325–343. Springer, 2003.
21. B. Xu and O. Wolfson. Time-series prediction with applications to traffic and moving objects databases. In *MobiDE 2003, September 19, 2003, San Diego, California, USA*, pages 56–60. ACM, 2003.
22. J. Yang and M. Hu. Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects. In *EDBT'06, Munich, Germany, March 26-31*, pages 664–681. Springer, 2006.