Spatiotemporal Pattern Mining Technique for Location-Based Service System

Thi Hong Nhan Vu, Jun Wook Lee, and Keun Ho Ryu

In this paper, we offer a new technique to discover frequent spatiotemporal patterns from a moving object database. Though the search space for spatiotemporal knowledge is extremely challenging, imposing spatial and timing constraints on moving sequences makes the computation feasible. The proposed technique includes two algorithms, AllMOP and MaxMOP, to find all frequent patterns and maximal patterns, respectively. In addition, to support the service provider in sending information to a user in a push-driven manner, we propose a rule-based location prediction technique to predict the future location of the user. The idea is to employ the algorithm AllMOP to discover the frequent movement patterns in the user's historical movements, from which frequent movement rules are generated. These rules are then used to estimate the future location of the user. The performance is assessed with respect to precision and recall. The proposed techniques could be quite efficiently applied in a location-based service (LBS) system in which diverse types of data are integrated to support a variety of LBSs.

Keywords: Spatiotemporal data mining, movement pattern, location prediction, location-based services.

Manuscript received Aug. 24, 2007; revised Dec.31, 2007.

This work was supported by the IT R&D program of MIC/IITA, Rep. of Korea [2006-S-022-01, Development of USN Middleware Platform Technology] and by a grant (#07KLSGC02) from Cutting-Edge Urban Development-Korean Land Spatialization Research Project funded by Ministry of Construction & Transportation of the Korean government.

Thi Hong Nhan Vu (email: nhanvth@yahoo.com) and Jun Wook Lee (phone: +82 42 860 5892, email: junux@etri.re.kr) are with IT Convergence Technology Research Laboratory, ETRI, Daejeon, Rep. of Korea.

Keun Ho Ryu (email: khryu@dblab.chungbuk.ac.kr) is with the School of Electrical and Computer Engineering, Chungbuk National University, Cheongju, Rep. of Korea.

I. Introduction

Convergence of location-aware devices, wireless communication, and geographic information system functionalities enables the deployment of new services such as location-based services (LBSs). Moving users of e-services involving location information disclose their positional information to the services, which in turn use this with other information to provide functionalities. Services accumulate data derived from the users' requests and integrate this with other user information in a multidimensional database [1], [2]. Our work concentrates on the development of data mining techniques for knowledge discovery and delivery in LBSs.

The data mining field dates back almost 20 years [3], [4]. However, the field of spatiotemporal data mining, which is extremely challenging due to the exponential explosion of the search space for knowledge, is still in its infancy [4], [5]. The algorithms proposed in [5] discover spatiotemporal periodic patterns from trajectories of equal length, which are then exploited in an index structure to support the execution of spatiotemporal queries. We are concerned with trajectories of random length and the problem of imprecise sampled points. In addition, DFS_MINE was introduced in [6] to discover spatiotemporal patterns for weather prediction. This paper studied the relationships between time-varying attributes for fixed locations, but it did not consider how the algorithm could be applied to moving object mining. In this case, it is necessary to seek the relationships between stable attributes of objects with varying-time locations. There have also been many studies on location prediction [7], [8]. The study which is the most relevant to ours is [8], in which the UMP algorithm was proposed. However, the new support value determining method of [8] prevents the pruning of unnecessary candidates.

Timestamps are not associated with rules, so when the user will enter a future location cannot be known. Moreover, the fixed cell in the network cannot be adapted to the thematic partition space considered here.

In this paper, towards the goal of reducing the error in the observations of a trajectory, the trajectory is reconstructed by re-sampling its positions. The proposed technique includes two algorithms, AllMOP and MaxMOP, to find all frequent movement patterns and maximal patterns, respectively. Though this task is extremely time-consuming, exploiting the spatiotemporal proximity feature of the problem domain makes the computation feasible. Because objects move in a thematically decomposed space, we take into account the concept of the graph and the transitive property of the similarity measurement of paths in a graph during the process of candidate generation, this helps to reduce the number of pattern candidates. Moreover, to control the density of the pattern regions and automatically adjust the shape and size of the regions, we employ a grid-based clustering technique. Efficiency is judged in comparison with the grid-based technique using the GSP algorithm [7] and DFS_MINE [6] with respect to the discovered knowledge, execution time, and memory requirement. The final goal is to support the information dissemination of the LBS system in a push-driven fashion. We propose a rule-based location prediction (RLP) algorithm to predict the future location of a moving user based on movement rules. Movement rules are generated from the movement patterns that are discovered from the user's historical movements using the algorithm AllMOP. The performance of this method is assessed with respect to precision and recall and compared to the previous method, UMP. The prediction of our technique is more precise than that of UMP. Using the proposed algorithm to predict future location and the algorithms to discover frequent movement patterns, the LBS system is able to efficiently provide users with traffic LBS and to send information to customers in a push-driven manner.

II. Problem Definition

Definition 1: Trajectory. The trajectory of a moving object with identifier o^j is defined as a finite sequence of points $\{(o^j, p_1, vt_1), (o^j, p_2, vt_2),\ldots, (o^j, p_n, vt_n)\}$ in the $X \times Y \times T$ space, where point p_i is represented by coordinates (x_i, y_i) at the sampled time vt_i for $1 \le i \le n$.

Assume that there are N distinct moving objects, DB is defined as the union of a time series of positions $DB = \bigcup_{j=1}^{N} D_j$, where D_j is a time series containing quadruples $(o^j, x^j_i, y^j_i, vt_i)$ for $1 \le j \le N$ and $1 \le i$.

The spatial organization of the reference plane $M \subseteq R$ is represented as a set of regions. The region is related to a specific thematic interpretation of space, so, M is represented as a finite set of regions $\{a_1,\cdots,a_n\}$ such that $\cup_{i=1}^{n} a_1 = M$ with $a_i \cap a_j = \phi$ and $i \neq j$. The moving possibility of an object from region to region is represented by a directed graph. After partitioning M, we get a hierarchical structure as introduced in [3]. However, in this study, we assume that a region of the lower level is fully contained in a region of the higher level.

Let T be the maximal timestamp among timestamps of the trajectories in the moving object database D. Let o^j_i denote the position of the moving object o^j, for $1 \le j \le N$ at timestamp vt_i for $1 \le i \le T$. The trajectory of an object can be defined by the sequence of points $o^j_1 o^j_2 \cdots o^j_K$ for $1 \le K \le T$.

Definition 2: Moving sequence. Given a minimal temporal interval τ and the trajectory's lifespan *max_span* [*start, end*], a moving sequence is a list of temporally ordered region labels $ms = \langle(a_1,t_1), (a_2,t_2),\cdots,(a_q,t_q)\rangle$, where a_i contains o^j_i, $t_i - t_{i-1} \ge \tau$, and $t_q - t_1 \le max\_span.end - max\_span.start$, for $q \le T$ and $1 \le i \le q$.

A sequence composed of k regions is denoted as a k-pattern. For example $\langle(R_1,t_1),(R_2,t_2),(R_2,t_3)\rangle$ is a 3-pattern.

Definition 3: Subsequence. For a moving sequence S_1, if region a_1 occurs before a_2, it is denoted as $a_1 < a_2$. S_1 is a subsequence of S_2 if there is a one-to-one temporally order preserving function f that maps regions of S_1 to those of S_2 such that for every $a_i \in S_1$: $a_i \cap f(a_i) \neq \varnothing$, if $a_i < a_j$ then $f(a_i) < f(a_j)$, and $t_{ai+1} - t_{ai} = t_{f(ai+1)} - t_{f(ai)}$.

For instance, in Fig. 3, sequence $\langle(R1_{23}, t_1), (R2_3, t_3)\rangle$ is a subsequence of $\langle(R1_3, t_1), (R0_1, t_2), (R2_3, t_3)\rangle$ since, sequentially, $R1_{23} \cap R1_3 \neq \phi$ at time t_1 and at time t_3 region $R2_3$ is the same.

Definition 4: Frequent pattern. A trajectory is said to comply with moving sequence *ms* if, for each region $a_i \in ms$ at vt_i, the point o^j_i of the trajectory is in a_i at the same time. The support of sequence *ms* can be defined as the number of trajectories in *DB* complying with it. If *ms* has *support*(*ms*) $\ge$ *min_sup*, where *min_sup* is a user-specified minimum support threshold, then *ms* is defined as a *frequent pattern*. A frequent pattern is *maximal* if it is not a subsequence of any other sequences.

To control the density of the pattern region a_i, a density-based partitioning method is exploited. Each region a_i of pattern *ms* is dense if the set of positions $A_i = \{o^j_i \mid o^j_i \in a_i\}$ forms a dense cluster. According to [9], a dense cluster is defined by two parameters r and *MinPts* points. We apply a modified version of the partitioning method in the consideration of a multi-level spatiotemporal grid. Progressing from finer to coarser, locally dense cells can be found, which later can be combined with dense nearby grid cells to form clusters. The size of cells at the lowest level is decided based on

the imprecision degree of the moving points, which will be presented later. In our case, *MinPts* is equal to the value *min_sup***N*. Therefore, if all regions in *ms* are dense, then *ms* is frequent.

Problem definition 1. Given the maximum speed v_{max} of a moving user, a moving object database *DB*, a reference plane $M \subseteq R^2$, directed graph *graph*, a time interval *max_span*[*start*, *end*] imposed on the lifespans of trajectories, and a minimum support *min_sup*, the problem is to discover frequent movement patterns in *DB*.

The second part of this study is concerned with movement rules discovered from the historical movements of the customer, which are used to predict the future location of a moving customer. The rule is defined as follows. A movement rule has the form $rul = A \rightarrow B$ in which A and B are two frequent movement patterns. The part of the rule before the arrow is the *antecedent*, and the part after the arrow is the *consequent* of the rule. As we describe later, the movement rule *rul* is generated from the frequent movement pattern $A \cup B$, which was just defined. The confidence of the rule *rul* is determined using the following formula:

$$confidence(rul) = \frac{support(A \cup B)}{support(A)} \quad (1)$$

in which the support of the pattern $A \cup B$ is the support of the rule. It is available after the frequent movement pattern discovering process, but the support of pattern A is not, and we have to determine it. It is notable through the process of discovering frequent patterns the sizes of pattern regions are reserved or get smaller when the length of the pattern increases. Therefore, the support of the antecedent A with length k is determined by accessing the set. All frequent k-patterns, F_k, are kept to find a pattern P_k whose regions coincide in temporal order or contain the regions of A. We then have its support, which is either the support of P_k or the number of objects in the overlapped region of some pair of regions at a specific time point belonging to P_k and A. Note that if just one region at time t in pattern A coincides with another region at time t in P_k, then the support of A is the same as that of P_k.

If the confidence value is at least a predefined threshold *min_conf*, then the rule is said to be frequent and confident.

Problem definition 2. Given the maximum speed v_{max} of a moving user, the historical trajectory of the user is a time series of positions denoted by $DB = \{(p_i, vt_i), 1 \le i \le n\}$ sampled at the time period Δt, a reference map $M \subseteq R^2$, the maximal timing constraint *max_gap,* minimum support *min_sup*, minimum confidence *min_conf*. The problem is to discover all frequent movement rules in *DB* satisfying both *min_sup* and *min_conf*.

III. Algorithms for Discovering Movement Patterns

We provide a function MINE_MOP to allow the adoption of the type of patterns we wish to obtain with the same input. Before processing, the moving object database *DB* is sorted according to object identifier o^i, and then it is sorted by timestamp *vt*. Next, the trajectories whose lifespans, denoted by $[vt_s, vt_e]$, are '*during*' the given time interval *max_span*[*start*, *end*] are extracted from *DB*. The temporal operator *during* was introduced in [10].

1. Trajectory Reconstructions

The sampling error across time was proved to be an error ellipse and a circle in the worst case [11], which is the case we take into account in this research. To make the operations more efficient, we operate on its minimum bounding rectangle, which is also the cell in the reference plane explained here. For a grid threshold r, and without considering time, reference plane M is decomposed into an $n_x \times n_y$ array of equal-sized cells. When time is considered, plane M is decomposed into uniform spatiotemporal units. A spatiotemporal unit is defined as the minimum spatial and temporal extent of interest. Each object stays in a cell for a certain time interval τ (or minimum interval *min_gap* here). Because the cell is used for storage, we have to decide the bounds of spatial and temporal extent.

For example, given a sampling rate of 12, the result of re-sampling the trajectories in Fig. 1 shows that cell d[2,3] in Fig. 1(a) is frequent for cell size r and re-sampling rate $\rho = 6$. However, for smaller cell size $r/2$ and re-sampling rate $\rho = 4$, the points in that location are distributed into two smaller cells d[5,6] and d[5,7] shown in Fig. 1(b), and both are infrequent.

Actually, the object's maximum velocity v_{max} and the chosen re-sampling ρ affect this choice. Re-sampling rate and cell size must be selected so that the object's movement produces at least one hit in each cell it visits. As a rule of thumb, parameter r must be chosen such that $(v_{max}/\rho) \ll (r/\sqrt{2})$. In addition, temporal extent τ is determined *a priori* and may change

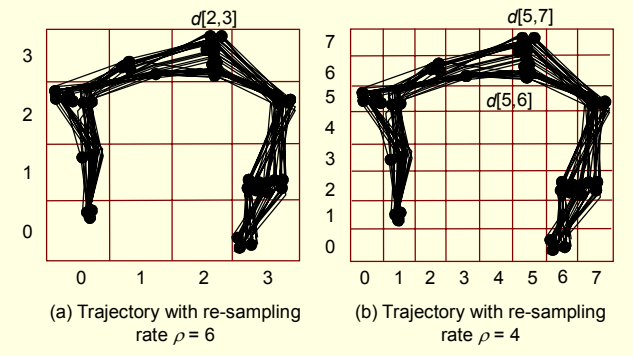


Fig. 1. Spatiotemporal extent vs. frequent cells.

depending on the application. As a rule of thumb, it should be chosen such that $1 \ll \rho\tau$, as $\rho\tau$ is a measure for hit number expectation per cell [12].

The object space having the origin (x_0,y_0) is represented as a regular grid and stored in an array $D[1{:}n_x,1{:}n_y]$. If $[S_x, S_y]$ is the two-dimensional size of the search space, each cell has size $[S_x/n_x, S_y/n_y]$. For point $P(x,y)$, the identifier of the cell Dij, to which P belongs, is determined by $i=(x-x_0)/(S_x/n_x)+1$ and $j=(y-y_0)/(S_y/n_y)+1$. Every trajectory is converted into a set of cell labels Dij, each of which is associated with a timestamp.

2. Trajectory Generalization

We assume multiple hits in a cell will count just once for a trajectory, so we have to generalize the trajectories. For a movement, we eliminate all consecutive points lying in the same cells and keep only the first point.

Assume that after projecting the points of trajectories in *DS* into cells, we obtain the result presented in Fig. 2(a). We remove the second points in the cells D20 and D12. The final results are shown in Fig. 2(b).

oid	*vt*	*x*	*y*	location
o^1	2006/1/30/ 7:00	10	0	D10
	2006/1/30/ 7:30	20	5	D20
	2006/1/30/ 8:00	25	7	
	2006/1/30/ 8:30	35	9	D30
	2006/1/30/ 9:00	33	13	D31
	2006/1/30/ 9:30	28	13	D21
o^2	2006/1/30/ 7:00	18	12	D11
	2006/1/30/ 7:30	16	22	D12
	2006/1/30/ 8:00	16	29	
	2006/1/30/ 8:30	18	35	D13

(a) Moving points stored in pages Dij

oid	*vt*	*x*	*y*	location
o^1	2006/1/30/ 7:00	10	0	D10
	2006/1/30/ 7:30	20	5	D20
	2006/1/30/ 8:30	35	9	D30
	2006/1/30/ 9:00	33	13	D31
	2006/1/30/ 9:30	28	13	D21
	2006/1/30/ 7:00	18	12	D11
o^2	2006/1/30/ 7:30	16	22	D12
	2006/1/30/ 8:30	18	35	D13

(b) Generalized trajectories

Fig. 2. Result of generalized trajectories in the database.

3. Dataset Transformation

Physically, the data structure of each cell in the moving sequence is constructed in the form of (Dij, o^j, vt_i), in which Dij contains a pointer, which indicates the page $D[i,j]$, where the position of object o^j at time vt_i is stored. Ultimately, the moving object dataset *DS* is converted into a set *MS* of moving sequences, each with distinct identifiers o^j.

4. Algorithm AllMOP

Discovering all frequent patterns is accomplished by the algorithm AllMOP, shown in Fig. 6. It takes the set of moving sequences *MS* from the previous step as input. The algorithm makes multiple passes to find all frequent patterns.

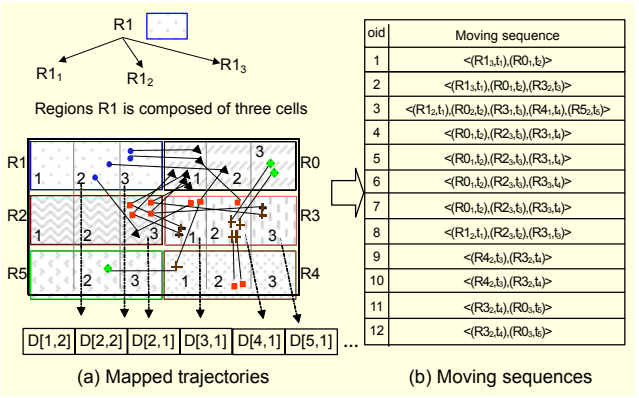
A. Finding Frequent 1-Patterns

It is impossible to apply techniques such as GSP or DFS_MINE directly because the shape and size of a region in one pattern is discovered and adjusted automatically at each pass. Therefore, we have to use the following procedure. First, a dataset of moving objects is decomposed into groups of moving points, each $A_i = \{o^j_i | o^j_i \in a_i,\}$ for one timestamp vt_i $(1 \le i \le T)$. Frequent 1-patterns are dense regions (or clusters) that are discovered from the sets A_i. To find them, for each timestamp vt_i, we scan the set *MS* to count cells and get frequent ones. Next, consecutive dense cells belonging to the same region a_i are merged into larger regions, which might be merged continuously to form clusters. They are then maintained in a set F_1, called frequent 1-patterns. The points lying in the sparse cells are assigned to the found clusters by applying range queries with diameter $(r/\sqrt{2})$. The points that do not belong to any cluster are called outliers. They are excluded from the cells as soon as they are found. The empty cells are discarded simultaneously.

Figure 3(a) shows a set of moving object trajectories in a two-dimensional space after the step of trajectory generalization. We assume that the maximal timestamp T is 5, and the support threshold, *min_sup*, is 2.

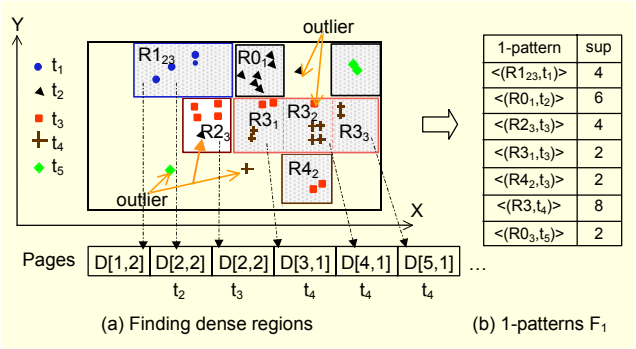
In this case, the spatial organization consists of five regions denoted by Rj $(0 \le j \le 5)$. The numbers that are marked in each region Rj index the cells belonging to that region. That is, the cell indexed by number 1 in region R1 is denoted as $R1_1$).

Moving points are projected into cells, which have pointers that indicate pages, one for each cell. In the figure, the pointers are represented by the dashed lines. As seen in Fig. 3(b), the starting point of object 1 at time t_1 logically lies in the cell $R1_3$ and is stored in the element $D[2,2]$. Its next position at time t_2 is in the cell $R0_1$ and is stored in $D[3,2]$. Next, dense regions are found. Figure 4(a) shows the groups of moving points obtained after decomposing the trajectories of Fig. 3(a). Consider the



oid	Moving sequence
1	<(R1₃,t₁),(R0₁,t₂)>
2	<(R1₃,t₁),(R0₁,t₂),(R3₂,t₅)>
3	<(R1₂,t₁),(R0₂,t₂),(R3₁,t₃),(R4₁,t₄),(R5₂,t₅)>
4	<(R0₁,t₂),(R2₃,t₃),(R3₁,t₄)>
5	<(R0₁,t₂),(R2₃,t₃),(R3₁,t₄)>
6	<(R0₁,t₂),(R2₃,t₃),(R3₁,t₄)>
7	<(R0₁,t₂),(R2₃,t₃),(R3₁,t₄)>
8	<(R1₂,t₁),(R2₃,t₂),(R3₁,t₅)>
9	<(R4₂,t₃),(R3₂,t₄)>
10	<(R4₂,t₃),(R3₂,t₄)>
11	<(R3₂,t₄),(R0₃,t₅)>
12	<(R3₂,t₄),(R0₃,t₅)>

Fig. 3. Moving trajectories used in the example.



1-pattern	sup
<($R1_{23}$,t_1)>	4
<($R0_1$,t_2)>	6
<($R2_3$,t_3)>	4
<($R3_1$,t_3)>	2
<($R4_2$,t_3)>	2
<(R3,t_4)>	8
<($R0_3$,t_5)>	2

Fig. 4. Example of finding frequent 1-patterns.

example of finding dense regions at time t_4. The count of cell $R4_1$ is just one less than *min_sup*, so it is discarded, and the only point in it is an outlier. We determine three dense cells, $R3_1$, $R3_2$, and $R3_3$, which refer to three different pages, namely, *D*[3,1], *D*[4,1], and *D*[5,1], as indicated by the pointers shown in Fig. 2(a). Since all of them are neighboring cells and are contained in the same region, R3, they are merged to form one cluster, R3. However, R3 still points to those three pages corresponding to the cells comprising it. Frequent 1-patterns F_1 are shown in Fig. 4(b).

B. Finding Frequent k-Patterns ($k \geq 2$)

A candidate 2-pattern is created by merging a pair of frequent 1-patterns in the consideration of the time constraint. Let (a_i,vt_i) and (a_j,vt_j) be two 1-patterns in F_1. A candidate 2-pattern, for example $\langle(a_i,vt_i), (a_j,vt_j)\rangle$, is created if $vt_i > vt_j$ and the regions a_i and a_j are neighbors because an object can only move into its neighboring regions.

For $k > 2$, candidate patterns are generated as follows. Given a set of frequent (k-1)-patterns, F_{k-1}, the candidates for the next pass are created by making a candidate for the next pass is created by merging a pair of (k-1)-patterns in F_{k-1}. Pattern $s_1 = \langle(a_1,vt_1), (a_2,vt_2),\ldots, (a_{k-1},vt_{k-1})\rangle$ joins with pattern $s_2 = \langle(b_2,vt'_2), (b_3,vt'_3),\ldots, (b_k,vt'_k)\rangle$. The candidate *cand* is produced by the merging of s_1 and s_2 if the following conditions are satisfied: after the first region of s_1 and the last region of s_2 are dropped, $vt_i = vt'_i$ and $a_i \cap b_i \neq \phi$ (for $2 \leq i \leq k$-1).

To facilitate fast and effective candidate generation, MBRs of the pattern regions are exploited. If all intersections of those pairs are non-empty, the created candidate pattern will be in the form of $cand = \langle(a_1,vt_1), (c_2,vt_2),\cdots, (c_{k-1},vt_{k-1}), (b_k,vt'_k)\rangle$, in which $c_2 = \text{MBR}(a_2) \cap \text{MBR}(b_2),\cdots, c_{k-1} = \text{MBR}(a_{k-1}) \cap \text{MBR}(b_{k-1})$. Then, we join the points in all regions of *cand* with criteria $R_i.o^j = R_j.o^j$ (or $cand = R_i \bowtie_{Ri.o^j = Rj.o^j \;\&\; Ri.vti = Rj.vtj} R_j$). The support value of a candidate pattern is the number of objects o^j that comply with the candidate patterns.

Next, the regions of the candidate need to be validated because they may no longer be dense after the join operation. If the support value is at least *min_sup*, then the regions of a candidate will be re-clustered. For a new cluster, a pattern will be created and the sizes of the pattern's regions are adjusted as well. For instance, consider the candidate pattern $\langle(R1_{23},t_1), (R0_1,t_2)\rangle$ shown in Fig. 5(a). After joining the points of $R1_{23}$ with those of $R0_1$, the size of cluster $R0_1$ is still maintained, but the size of cluster $R1_{23}$ is reduced to $R1_3$. Second, consider the candidate pattern $\langle(R2_3,t_3), (R3,t_4)\rangle$ depicted in Fig. 5(b). After joining the points of $R2_3$ with those of R3, the region $R2_3$ is still the same, but the remaining points of R3 are grouped into two clusters $R3_1$ and $R3_3$. Therefore, two new patterns are created, namely, $\langle(R2_3,t_3), (R3_1,t_4)\rangle$ and $\langle(R2_3,t_3), (R3_2,t_4)\rangle$.

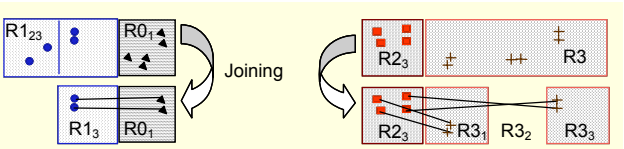


Fig. 5. Validating candidate patterns.

Finally we consider the candidate pruning method. A candidate *k*-pattern will be discarded if it has an infrequent subpattern. To tackle this problem, we keep a list of all minimal infrequent patterns in the memory *MinInfreqList* as introduced in [6]. Initially, *MinInfreqList* contains all infrequent 2-patterns. When a new candidate pattern *cand* is generated, we check whether it is a superpattern of any pattern in this list. If it is, it

Algorithm AllMOP()
For every 1-pattern $(R_i, t_i) \in F_1$ //*generating 2-patterns*
  For $(R_j, t_j) \in F_1$ && $i \neq j$ //*assume* $vt_i > vt_j$
    if ($\exists$ an edge in *graph* connecting R_i to R_j)) then
      $C_2 \leftarrow C_2 \cup \langle(R_i, t_i), (R_j, t_j)\rangle$
For all candidate 2-patterns $\langle(R_i, t_i),(R_j, t_j)\rangle \in C_2$
  $cand = R_i \bowtie_{Ri.o^j = Rj.o^j} R_j$;
  set_new_regions ← Clustering points in *cand*;
  If $|set\_new\_regions| \neq \phi$ then for each new region R_i'
    $F_2 \leftarrow F_2 \cup \langle(R_i', t_i), (R_j', t_j)\rangle$
  else Insert(*MinInfreqList*, *cand*);
For ($k = 3$; $F_{k-1} \neq \phi$; k++)
  For each pair P_i & $P_j \in F_{k-1}$
    cand ←joining the pair (P_i, P_j) after checking MBR-intersection
    if (*cand* $\neq \phi$) then $C_k \leftarrow C_k \cup cand$;
      Prune candidates by looking for a subpattern in *MinInfreqList*;
    If such a pattern exists, then update *MinInfreqList*;
  // *determining support the remaining candidates*
  For each $cand \in C_k$
    $cand = P_i \bowtie_{Pi.o^j = Pj.o^j \;\&\; Pi.vti = Pj.vtj} P_j$;
  if (|objects o^j complying with *cand*| $\geq$ *min_sup*)
    set_new_regions ← reclustering points in the regions of *cand*;
  If $|set\_new\_regions| \neq \phi$, then for each new region R_i'
    $F_k \leftarrow F_k \cup new\_pattern$;
  else Insert (*MinInfreqList*, *cand*);
FMOP← set of all frequent sequences in F_k;
return *FMOP*;

Fig. 6. Algorithm for finding all frequent patterns.

will be discarded at once, without making a temporal join on the sets of points of the candidate pattern regions.

5. Algorithm MaxMOP

The algorithm MaxMOP mines maximal patterns, which could save a lot of space and maintain all of the necessary information. In terms of the data structure, in addition to *MinInfreqList*, which is used for candidate pruning as previously mentioned, we need two other lists: *MaxFreqList* to store the maximal frequent patterns in the memory, and *CandList* to contain the patterns to prepare for the generation of new patterns. Both *MaxFreqList* and *CandList* are initialized with frequent 2-patterns; however, *MinInfreqList* is initialized with infrequent 2-patterns. *MaxFreqList* is updated with a new candidate pattern, *cand*, if all of the following three conditions hold: *cand* is not in the *MaxFreqList*, *cand* is found to be frequent, and *cand* is not a subpattern of any pattern in this list. If the three conditions hold, *cand* is inserted into *MaxFreqList*. After insertion, we remove all subpatterns of *cand* from the list. The result of this algorithm is kept in *MaxFreqList*.

IV. Predicting Future Location of Moving User

The purpose of the algorithm is to predict the location of a moving customer based on movement rules. In addition to re-sampling the sampled points, it is necessary to transform the time series of locations into moving sequences. A moving sequence is created if the time difference between two consecutive points is greater than *max_gap*. Next, the AllMOP algorithm takes the set *MS* of moving sequences as input, and a directed graph, *min_sup*, is called to discover all frequent movement patterns.

Assume a movement pattern $P = <(l_1, vt_1), (l_2, vt_2),\cdots,(l_k, vt_k)>$, where $k>1$. All possible rules or the candidate rules that can be derived from such a pattern are the following: $<(l_1, vt_1)> \rightarrow <(l_2, vt_2),\cdots,(l_k, vt_k)>$; $<(l_1, vt_1), (l_2, vt_2)> \rightarrow <(l_3, vt_3),\cdots,(l_k, vt_k)>$; $\cdots;<(l_1, vt_1), (l_2, vt_2),\cdots,(l_{k-1}, vt_{k-1})> \rightarrow <(l_k, vt_k)>$. The confidence values of the movement rules are computed using (1). Any rule whose confidence is higher than a predefined confidence threshold *min_conf* is selected. In this way, we discover all frequent movement rules from a set of movement patterns.

With the movement rules discovered, the future location of a moving object can be estimated. Assume a user has followed the path $P = <(l_1, vt_1), (l_2, vt_2),\cdots,(l_{j-1}, vt_{j-1})>$ up to now. The pseudo-code of the algorithm RLP is shown in Fig. 7. The algorithm searches for the matching rules. Their antecedents preserve the time order when mapping them to P and have non-empty overlapping with the regions of P in that order. The last location in the antecedent overlaps l_{j-1} as well. The first location of the consequent of such a rule and the value that is the sum of the confidence and the support values of the rule are stored in an array. This array is then sorted in descending order according to the sum of support and confidence values. The location and the timestamp in the first tuple is the next location the user will enter at that time.

```
Algorithm RLP()
PredLoc ← φ, t ← 1;
For each rule r = <(a1, vt1),…(ai-1, vti-1)> → < (ai, vti),…(ak, vtk) > ∈ Rules
    If (when mapping P to r the time order is preserved and the
        regions of {l1, l2,…,lj-1} successively have nonempty
        overlapping with the regions of {a1, a2…,ai-1} & (lj-1 overlaps ai-1)
            MatchingRule ← MatchingRule ∪ r;
            NextLocArray [t++]←(<ai, vti>, r.support +
                r.confidence);
    NextLocArray ← sort(NextLocArray);
    PredLoc ← NextLocArray[1];
return PredLoc;
```

Fig. 7. Algorithm for estimating future location.

V. Implementation and Performance Analysis

In this section, we present experiments that we carried out to evaluate the performance of the proposed techniques.

1. Evaluation of AllMOP and MaxMOP

We validated the efficiency of the proposed algorithms under diverse settings of parameters and datasets and compared them with grid-based technique using the GSP and DFS_MINE algorithms.

A. Synthetic Dataset Generation

To evaluate the efficiency of the proposed algorithms we built a dataset generator as follows.

Prior to producing trajectories, the regions on the map M for the maximal movement patterns were decided. Then, we determined the directed graph for this map, a bidirected graph in this case. Assume that P is a generated pattern. A set of trajectories with lifespan in timestamp (vt_t, $1 \le t \le T$) is created on the space. For each region of P, we assigned a set of points at a specific timestamp vt_t on those trajectories. It is important that the distance between two points on the trajectories are set such that it is proportional to the minimum temporal extent τ, sampling rate Δt in the pattern, as well as object's velocity. Moreover, we guarantee that the positions of the different objects arriving at the same location and at the same time will not coincide. This means that they are far enough apart, that is, $\alpha = r / (min\_sup * N)$. For an object's movement, it is necessary to determine if it has a random trajectory or if it complies with some frequent pattern. If its movement has a

random trajectory, the moving directions of its points are changed and the distance between the two points is also distorted. Otherwise, a maximal pattern P is selected and the moving object's trajectory is produced in the following way. Assume that the object's current point is p_{i-1}, and the object moves to the next point p_i. If the time difference between the two pattern regions corresponding to p_i and p_{i-1} is a maximum of ρ, then p_i of the trajectory is generated in its corresponding region and within α range. If not, p_i is generated randomly but such that the movement directs the next pattern region. A random trajectory is determined by a random direction with respect to the previous location and a random distance.

A map with the size of [1000, 1000] was partitioned into regular regions 250 m long and 125 m wide, that is, the map consists of 32 regions, denoted by a_i. We assumed that all of the objects in the map would travel at the maximum velocity v_{max} of 11 m/s, and the sampling rate, Δt, was 8. Throughout the experiment, we chose the same values for the re-sampling rate, ρ, and the temporal extent, τ. In accordance with such a setting of space, if the space is a 16×16 grid and τ is 2, then each region a_i is composed of 8 cells Dij with size r =62.5. The moving objects were generated such that each object would hit four cells of the region a_i and when we aggregated them using the higher resolution r = 125 and τ = 4 (in this case, the region a_i consisted of two cells with this size) there were two pairs of consecutive points falling into two cells with that larger size. On average, 70% of the object movements of the generated dataset had the maximal length of T. On average, 30% of those trajectories follow the same path. The remaining trajectories of the dataset (30%) are called random trajectories. The generated dataset, D500_T40_L10, includes 500 trajectories. The maximal timestamp, T, is 40, and the maximal pattern length, L, is 10.

B. Performance of AllMOP and MaxMOP

In all tests presented here, we applied the cell size r = 125 and the temporal extent τ = 4. The results show that for constant dataset size, the execution time of AllMOP gradually increases with the change of trajectory length (see Fig. 9). We obtained the similar results for various database sizes with the same maximal length L. This is because the number of frequent patterns increases as the *min_sup* value decreases, which leads to longer execution time. We can affirm that AllMOP is scalable because the number of objects increases from 500 to 3,000 (a factor of six) with a fixed maximal length, while the required time increases by a factor of less than six. Our other tests also returned the same results. Figure 9 shows a comparison of our two algorithms, AllMOP and MaxMOP. We used a fixed maximal timestamp T = 96 and *min_sup* = 7% for the datasets. Even though MaxMOP is little more time-consuming than AllMOP, it maintains only maximal patterns while AllMOP causes memory overflow. The cost of MaxMOP for mining maximal patterns is higher than that of AllMOP because MaxMOP includes an additional operation for pattern updating. However, its additional cost is not prohibitive.

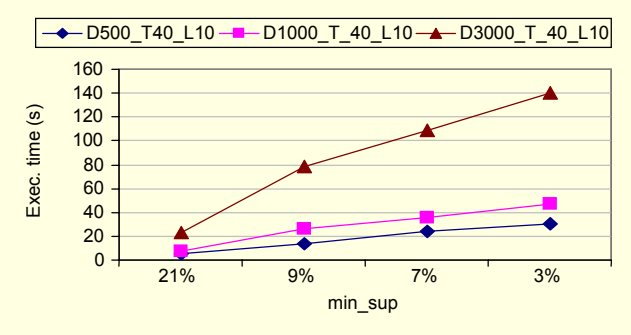


Fig. 8. Execution time vs. data size.

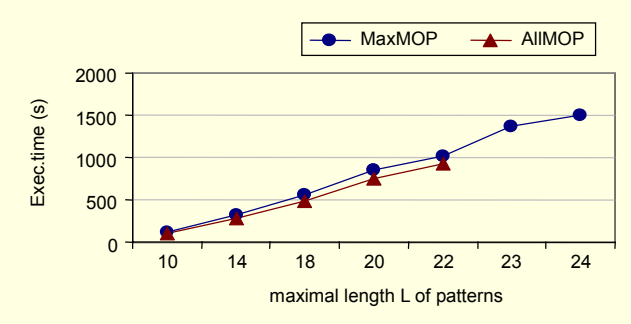


Fig. 9. AllMOP vs. MaxMOP with 3000 trajectories.

C. Comparison with GSP and DFS_MINE

The following experiments were performed on the dataset D3000_T40_L10 to assess the effectiveness of our techniques compared with the grid-based technique using GSP and DFS_MINE.

D. AllMOP vs. GSP

We used the dataset D3000_T40_L10 and considered both small cell size and large cell size.

For a small cell size (r = 62.5), even though GSP is very fast, moving points are split into different cells, so frequent cells are missed, which leads to longer frequent patterns being missed. As seen in Fig. 10(a), for the *min_sup* less than 9%, long patterns are missed. The cost of AllMOP is higher than that of GSP. The time required to scan the dataset of GSP to count the supports of candidate patterns, which are reduced after each step, is shorter than the time required to join and adjust the size of pattern regions with AllMOP (Fig. 10(a)).

For a larger cell size (r = 125), GSP discovers all actual

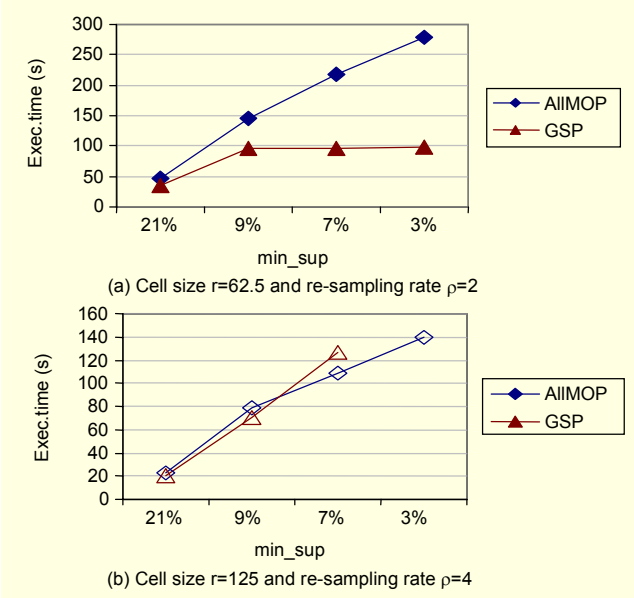


Fig. 10. AllMOP vs. GSP with running time as function of *min_sup*.

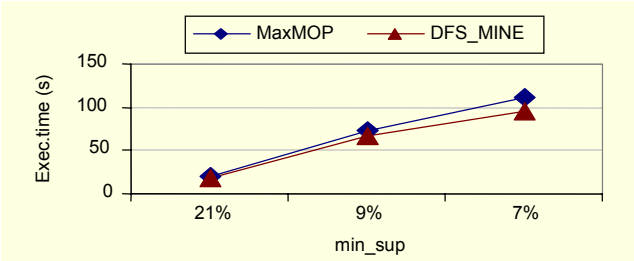


Fig. 11. MaxMOP vs. DFS_MINE with running time as function of min_sup.

frequent cells but overestimates them. This causes memory overflow with *min_sup* = 7% due to the storage of all subpatterns of a frequent pattern (Fig. 10(b)). With this cell size, for an actual pattern, GSP produces more than one maximal pattern including all subpatterns, whereas our clustering method compresses them into just one pattern, thus, saving lots of memory. Moreover, thanks to the efficient pruning method, the processing cost is significantly reduced. In contrast, the redundant candidates of GSP lead to high cost. For low minimum support values, the cost of GSP is less than that of AllMOP because clustering incurs a high cost, but the reverse is true for higher values because many more patterns are generated by GSP than AllMOP.

E. MaxMOP vs. DFS_MINE

This experiment was carried on the dataset D3000_T40_L10, and the cell size *r* was 125. Figure 11 shows that MaxMOP incurs a slightly higher cost than DFS_MINE due to clustering

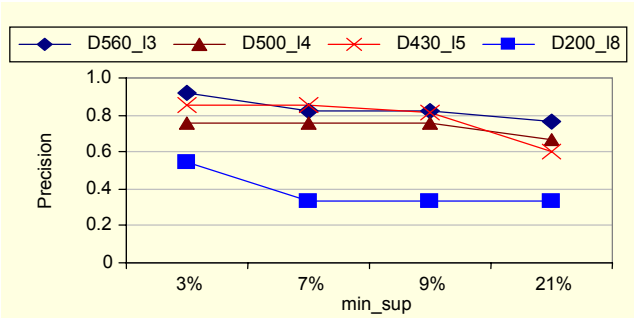


Fig. 12. Precision as function of *min_sup*.

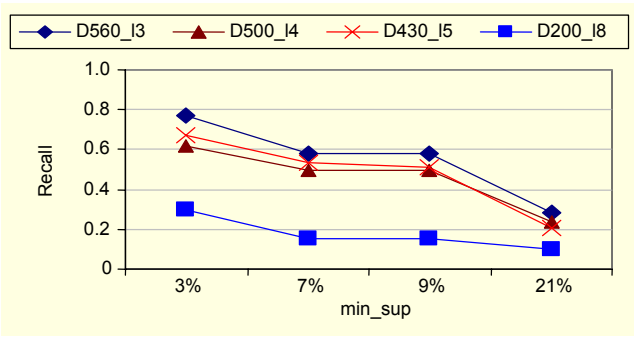


Fig. 13. Recall as function of *min_sup*.

and adjustment of pattern region size. Nevertheless, MaxMOP inherits an advantage of DFS_MINE, namely, the pruning method. Moreover, its temporal operation in candidate pattern generation plus the consideration of the direction of movement helps to eliminate many redundant candidates generated when DFS_MINE is used. However, without the clustering operation, the large number of actual patterns generated by DFS_MINE leads incurs a higher cost. Our test also demonstrated that MaxMOP efficiently compresses the patterns according to thematic regions as a result of the clustering operation.

2. Evaluation of the RLP Algorithm

The datasets used here were generated in the same way as the trajectories introduced in the previous section, but with a small modification. We distinguished the trajectories of the user by changing the weekday, but we kept the day timestamp of the trajectories in all datasets that we previously used. Two types of datasets were distinguished: the training set and the test set.

The following experiments were conducted to optimize the parameters of our methods, which are *min_sup* and *min_conf*. The chosen training set in this case is D3000_T40_L10. The test set is denoted as D200_l8, which means this set has 200 movements with the length of 8. We set the cell size *r* to 125 and the temporal extent τ to 4 in the next two experiments as

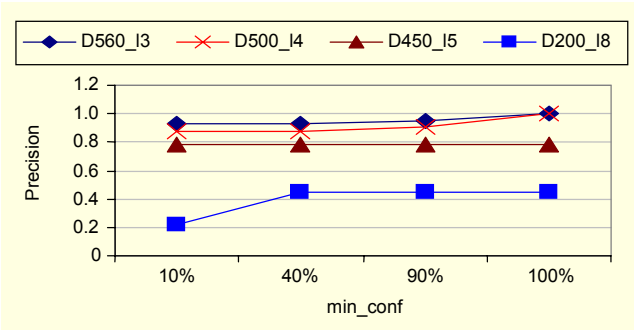


Fig. 14. Precision as a function of *min_conf*.

default values if there is no indication of specific values.

A. Effect of Minimum Support Values

We fixed the minimum confidence value at 70%. Figures 12 and 13 show that increasing the support threshold results in decreased precision and decreased recall. When the *min_sup* value increases, the number of frequent patterns decreases; thus, the number of movement rules in decreases, and this negatively affects the relevant and correct prediction. Additionally, a decrease in the number of movement rules leads to an increase in the number of false negative locations, and this brings about a decrease in recall. It would be the most appropriate to adopt a *min_sup* of 3% since both precision and recall are highest for this value.

Since precision and recall have inverse variation, we considered only precision in the next tests.

B. Effect of Minimum Confidence Values

We used the same test sets that were used to evaluate the effect of *min_sup* on the recall and precision of the RLP algorithm to investigate how the minimum confidence values *min_conf* affect them by fixing *min_sup* at 9%. As Fig. 14 shows, precision increases and recall decreases as the minimum confidence value increases because only the rules with high confidence values are used for prediction.

C. Effect of Changing Granularities

In this test, the movement rules used in the experiments were obtained from the dataset D3000_T40_L10 and the future location prediction was carried out on the test set D500_I4. As previously described, the map has 32 regions, and the space is partitioned into regular cells. Cell sizes were adjusted according to the temporal extent τ by the following setting. Given the sampling rate $\Delta t = 8$, if the temporal extents selected are $\Delta t/2 = 4$ and $\Delta t/4 = 2$, then the cell sizes are changed from $r = 125$ to $r/2 = 62.5$, that is, the space becomes the 8×8 grid and the 16×16 grid, respectively. Therefore, each region consist

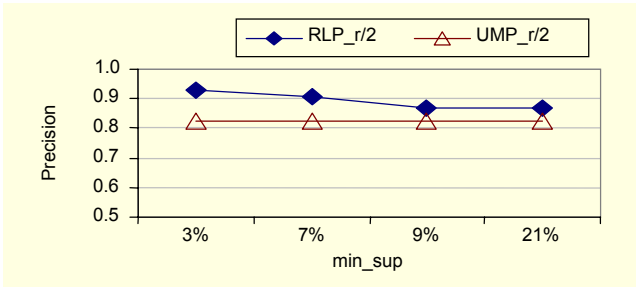


Fig. 15. RLP vs. UMP with precision as functions of *min_sup* for cell size *r*/2.

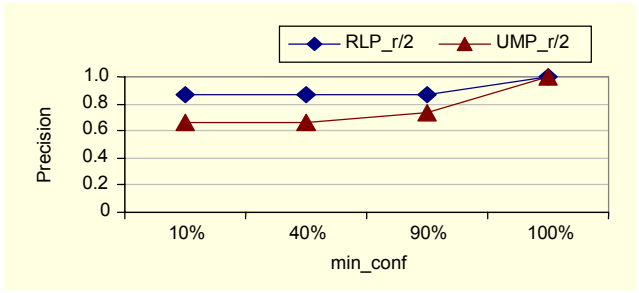


Fig. 16. RLP vs. UMP with precision as a functions of *min_conf* for cell size *r*/2.

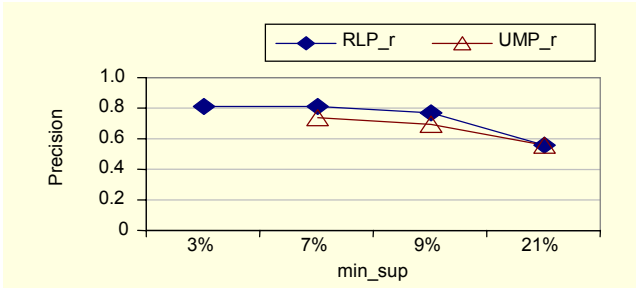


Fig. 17. RLP vs. UMP with precision as a function of *min_sup* for cell size *r*.

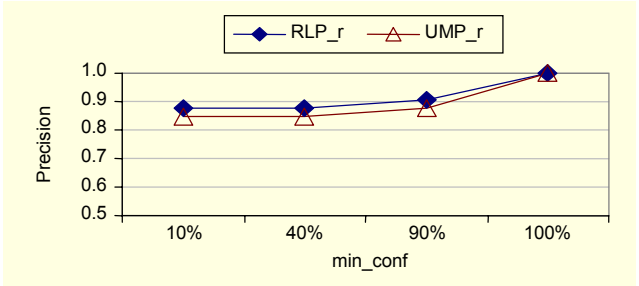


Fig. 18. RLP vs. UMP with precision as a function of *min_conf* for cell size *r*.

of 2 cells in the case of 8×8 grid and 4 cells in the case of 16×16 grid. UMP was tested on the grid defined in this research and in the same experimental setting for our method.

As seen in Figs. 15 and 16, the curves that represent the precision and the recall of UMP are much lower than those of

RLP. Because UMP is a grid-based method, with small cell sizes it misses a lot of frequent cells. When frequent cells are missed, longer frequent patterns are lost, and this in turn results in the loss of movement rules. The decrease in the number of rules reduces precision. Moreover, the loss of rules extends unpredictability to many locations, so recall is also reduced. Thus, these effects on precision and prediction are a function of the *min_conf* as shown in Fig. 16.

Figures 17 and 18 show a comparison of the precision and recall of RLP and UMP for the larger cell size *r*. The prediction ability of UMP improves, but is still lower than that of RLP, when the number of movement rules increases. Despite higher quality, UMP has a disadvantage, that is, the memory overflows when *min_sup* is rather small, less than 7% in our case (see Fig. 17).

VI. Conclusion

This paper proposed a new technique including two algorithms, AllMOP and MaxMOP, for mining frequent movement patterns from a massive amount of raw data referred in space and time. Movement patterns discovered from trajectories in traffic may allow inducement of traffic flow information to help users travel efficiently. Additionally, to support service providers in sending information to users in a push-driven manner, the RLP technique for estimating users' future locations was proposed based on the users' past movements. Our proposed algorithms were implemented, and their performance was studied with various parameters and datasets and was compared with the previous methods. We compared the frequent movement pattern mining technique with a grid-based technique using the GSP and DFS_MINE algorithms with respect to the capability of knowledge discovery, execution time, and memory requirement. The results indicated that our methods are quite good in terms of running time and the compression of discovered knowledge due to the operation of trajectory reconstruction and the use of clustering. The efficiency of the RLP location prediction technique was evaluated with respect to precision and recall. Compared with the previous grid-based UMP algorithm, our technique is better. The results showed that the performance of all algorithms is affected by the re-sampling rate and the distance threshold used in clustering; therefore, we also suggested suitable values.

References

[1] S. Jensen, A. Kligys, T.B. Pedersen, and I. Timko, "Multidimensional Data Modeling for Location-Based Services," *Proc. VLDB*, vol. 13, no. 1, 2004, pp.1-21.

[2] K.W. Min, K.W. Nam, and J.W. Kim, "Multilevel Location Trigger in Distributed Mobile Environments for Location-Based Services," *ETRI J.*, vol. 29, no. 1, 2007, pp. 107-109.

[3] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. VLDB*, 1994, pp. 487-499.

[4] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proc. EDBT*, 1996, pp. 3-17.

[5] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D.W. Cheung, "Mining, Indexing, and Querying Historical Spatiotemporal Data," *Proc. SIGKDD*, 2004, pp. 236-245.

[6] I. Tsoukatos and D. Gunopulos, "Efficient Mining of Spatiotemporal Patterns," *Proc. SSTD*, LNCS, vol. 2121, 2001, pp. 425-442.

[9] D. Katsaros, A. Nanopoulos, M. Karakaya, G. Yavas, O. Ulusoy, and Y. Manolopoulos, "Clustering Mobile Trajectories for Resource Allocation in Mobile Environments," *Proc. Intelligent Data Analysis Conference*, vol. 2810, 2003, pp. 319-329.

[10] G. Yava, D. Katsaros. O.Ulusoy, and Y. Manolopoulus, "A Data Mining Approach for Location Prediction in Mobile Environments," *Data and Knowledge Engineering,* vol. 54, no. 2, Aug. 2005, pp. 121-146.

[11] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. ACM Knowledge Discovery and Data Mining*, 1996, pp. 226-231.

[12] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communication of ACM*, vol. 26, 1983, pp. 832-843.

[13] D. Pfoser and C.S. Jensen, "Capturing the Uncertainty of Moving-Object Representations," *Proc. Advances in Spatial Databases, 6th International. Symposium SSD*, 1999, pp. 111-132.

[14] N. Meratnia and R.D. By, "Aggregation and Comparison of Trajectories," *Proc. GIS*, ACM, 2002, pp. 49-54.

Thi Hong Nhan Vu received the MS and PhD degrees from the School of Electrical and Computer Engineering, Chungbuk National University, Rep. of Korea, in 2004 and 2007, respectively. She is currently with the Electronics and Telecommunications Research Institute, Rep. of Korea. Her major research interests include spatiotemporal data management systems, location-based services, data mining, machine learning, context aware applications in ubiquitous computing environments, and the development of ubiquitous sensor network platforms.

Jun Wook Lee received the MS and PhD degrees from the School of Electrical and Computer Engineering, Chungbuk National University, Rep. of Korea, in 1997 and 2003, respectively. He is currently with the Electronics and Telecommunications Research Institute, Rep. of Korea. His major research interests include sensor data mining, spatiotemporal data mining, location-based services, context awareness in ubiquitous computing environments, and the development of USN middleware platforms.

Keun Ho Ryu received the PhD degree from Yonsei University, Rep. of Korea, in 1988. He is a professor at Chungbuk National University, Rep. of Korea. He worked at the University of Arizona as a post doctoral research scientist and at the Electronics and Telecommunications Research Institute. He has served on numerous program committees including the IEEE International Conference on Advanced Information Networking and Applications (AINA), the IEEE International Symposium on Mining the Asian Web (MAW), the International Conference on Web Engineering (ICWE), the International Conference on Web-Age Information Management (WAIM), the Asia Pacific Web Conference (APWeb), and the International Conference on Web Information Systems (WISE). He was a demonstration co-chair of the Very Large Data Base Conference (VLDB). His research interests include temporal databases, spatiotemporal databases, temporal GIS, ubiquitous computing and stream data processing, knowledge base information retrieval, database security, data mining, and bioinformatics. He has been a member of the IEEE and a member of the ACM since 1983.