Mining Complete User Moving Paths in a Mobile Environment

Shin-Mu Tseng W. C. Chan

Department of Computer Science and Information Engineering
National Cheng Kung University, Tainan, Taiwan, R.O.C.
tsengsm@mail.ncku.edu.tw

Abstract. In a mobile computing environment, analyzing the moving patterns of the mobile users can help understand the behavior of the users. Although a number of researches have been done on the analysis of moving patterns of users, they focused primarily on location tracking or discovering the cyclic moving patterns. However, in real applications, there exist more types of user moving patterns that have not been explored yet. In this paper, we propose a new data mining method that can discover *complete moving paths* of the users, which include the cyclic, non-cyclic and backward patterns in terms of user's movement records.

1 Introduction

With the rapid development of the wireless communication techniques, more and more users subscribe various kinds of mobile services in a mobile environment due to the advantages of easy accesses. When a user is roaming in a mobile environment, the information relevant to the movement of the user will be recorded in the system log. From the business standpoint, valuable information regarding the user's moving behavior may exist in the system log.

Therefore, it is an important topic to discover the moving patters of the mobile users from the system log, such that personalized services could be provided by utilizing the discovered moving behavior of users. In past studies, some methods had been proposed for mining the users' moving patterns. In [5], a probability-based approach was proposed to build a model for predicting the movement of users. Moreover, a data delivery strategy was developed based on the prediction model such that the latency for the mobile users to access the interested information could be minimized. However, the probability-based approach will encounter the efficiency problem when the input database becomes very large.

In recent years, data mining techniques have been developed rapidly and applied in a wide range of applications since they can efficiently discover valuable information hidden in large databases. Some well-known data mining techniques including association rules discovery [1,

4] and sequence patters discovery [2, 8, 10] have been applied successfully to various kinds of applications like basket analysis and navigation pattern analysis for web browsing. However, few studies have used data mining techniques to analyze the moving patterns of mobile users.

To our best knowledge, one of the few studies working on mining user moving patterns was done by Peng et al. [9]. An algorithm named LM was proposed to discover cyclic moving patterns of users, which mean the cyclic paths an user go through with the same starting and ending location. Although the LM method proposed in [9] was useful for finding users' cyclic moving patterns, there exist more types of patterns to be discovered from users' moving log in a mobile environment.

Consider a network as shown in Figure 1, in which each node represents a base station. Table 1 lists four example moving sequences of mobile users. For example, the moving sequence MS_1, <g, b, f, i, f, g>, means the sequence an user starts from g, moving through b, f, i, and backward to f and g consecutively. By using LM method [9] to analyze the moving sequences as listed in Table 1 (setting the threshold $\xi = 50\%$), some cyclic moving patterns can be found as shown in Table 2. Although this provides useful information regarding user's moving behavior, some problems arise. First, in addition to the cyclic moving patterns, there exist frequent moving paths that are the superset of the cyclic moving patterns. Second, although Peng et al. [9] pointed out the backward pattern problem as incurred in [2], this problem was not resolved in LM method. For example, the discovered moving patterns MP_1, {g, b} in Table 2, actually comes from the moving sequences MS_1 and MS_3 as in Table 1. However, it was not differentiated that the moving path g $\rightarrow$ b was a forward movement in MS_1 but a backward movement in MS_3.

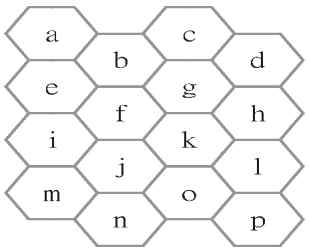

Figure 1. A network architecture.

Table 1. Example moving sequences.

MS_{id}	D_{MS}
MS_1	<g, b, f, i, f, g>
MS_2	<g, k, o, l, k,
MS_3	<b, f, i, f, g, b>
MS_4	<f, k, o, l, k, f>

Table 2. Discovered moving patterns by LM.

MPid	Moving Pat-
MP_1	{g, b}
MP_2	{k, o, l, k}
MP_3	{b, f, i, f, g}

In this paper, we propose a new data mining method named *CMP (Complete Moving Path)-mining*, which can resolve the problems mentioned above. CMP-mining uses a special data structure named CMS-tree to efficiently discover complete moving paths of users, which contain all of cyclic, non-cyclic and backward patterns. Through empirical evaluation, CMP-mining was also shown to have good performance in terms of efficiency and scalability.

The rest of the paper is organized as follows: In section 2, we describe the definition of the studied problem; our new data mining method is introduced in Section 3; empirical evaluation results for the proposed method is described in Section 4; a conclusion is made in Section 5.

2 Problem Descriptions

In this work, the architecture for the mobile system is based on the two level standard of IS-41/GSM [7], and the two-tier architecture including HLR (home location register) and VLR (visitor location register) is used for database management. For each mobile user, the relevant information and the current location are stored in the HLR databases via the registration operation. Whenever a registration operation is performed in reflecting the movement of a user, a moving record is recorded in the "moving log" with the form of (VLR_{old}, VRL_{new}), which indicates that the user has moved from the VLR_{old} to VRL_{new}. The old VLR is initialized as NULL for a new user task. Hence, the complete moving sequences of all users can be obtained from the moving log.

Under this architecture, the problem we want to resolve is to discover the *maximal complete moving path*, which is different from those discovered by MF algorithm [2] or MM algorithm [9]. Consider the example of user movement in Figure 2, in which the user starts from location a, and moves to b and c, then backward to b and a, finally moves to e as end stop. By using MF algorithm, the discovered moving sequence will be {abc, ae} since the backward movement will not be taken into considerations. If the MM algorithm is applied, the moving sequence discovered will be {abcba} since it focuses on finding the cyclic moving patterns. Being different from the previous approaches, we will discover the frequent complete moving path, which will be {abCBae}, where the upper case indicates a backward movement. This contains all of the non-cyclic, cyclic and backward moving patterns.

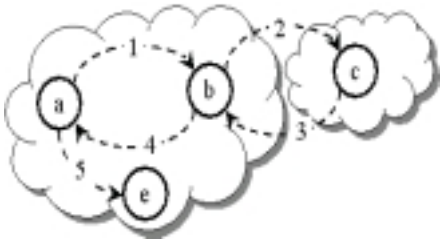

Figure 2. Example moving behavior.

3 Proposed Method

We propose a new method, namely *Complete Moving Path (CMP)*-mining, for discovering the complete moving paths of mobile users as described in Section 2. The CMP-mining method is based on the concept of WAP (Web Access Patterns)-mining [8] and augmented with several new ideas. The CMP-mining method consists of three steps: 1) Construction of Complete Moving Sequence (CMS)-tree: A special data structure named CMS-tree is constructed to record the moving sequences of all users by reading from the moving log, 2) Mining of complete moving paths: Discover the frequent complete moving paths based on the constructed CMS-tree and the specified minimum support, 3) Generation of the maximal complete moving paths: Collecting and integrating the frequent moving paths discovered in Step 2 and generate the maximal complete moving paths as the final output. It is obvious that

Step 3 is trivial once step 2 is done. For example, suppose four frequent complete moving paths namely {Ae, cg, Gh, cGh} were discovered in step 2, only {Ae, Gh, cGh} will be generated as the maximal complete moving paths since cg is the substring of cGh. Hence, we will describe only the details of step 1 and step 2 in the following sections.

3.1 Construction of CMS-tree

The main purpose of constructing CMS-tree is to aggregate the user's moving records in the moving log into the complete moving sequences such that the task of discovering frequent complete moving paths can be done efficiently. The segment range in the moving log to form a complete moving path is application dependent and can be determined by the system administrator. The main advantage of CMS-tree is that all information for mining the maximal complete moving paths will be stored into it after one database scan of the original moving log. There exist some basic properties in constructing the CMS-tree:

1. The moving log will be scanned once only to transform the needed information into the CMS-tree.
2. For a moving record <VLR_{old}, VRL_{new}>, the movement is considered as a backward if VLR_{old} appears later than VRL_{new} in the current moving sequence segment.
3. The backward information will be recorded in the parent node of the CMS-tree.
4. A header table is used to record the starting location of a moving sequence. When a node is inserted into the CMS-tree initially and linked to the header table, there is no need to consider the backward information of that node. This is because there will be no further information to be recorded back for the last node in a moving sequence.

We illustrate how CMS-tree is constructed by using a running example. Suppose a moving log is obtained as in Table 3 under the network architecture as in Figure 1. First of all, a root node is created for the CMS-tree and its identifier is set as NULL. Then, the moving log is scanned record by record. For each moving record <VLR_{old}, VRL_{new}>, a new task is initiated and the last task is terminated if the value for VLR_{old} is NULL (as the assumption in

Section 2). Each moving record is thus transformed into a path in the CMS-tree. Hence, a moving path <g, f, K, f, e> will be generated first in the CMS-tree, where the upper-case letter represents a backward information. Then, three moving paths <g, f, K, g, h>, <b, g, k, F, g> and <b, f, K, F, b> will be generated into the CMS-tree consecutively. Finally, a complete CMS-tree is constructed as shown in Figure 3. The algorithm for constructing CMS-tree is as shown in Figure 4.

Table 3. A moving log file.

(VLR_{old}, VRL_{new})	(VLR_{old}, VRL_{new})
(NULL, g)	(NULL, b)
(g, f)	(b, g)
(f, k)	(g, k)
(k, f)	(k, f)
(f, e)	(f, g)
(NULL, g)	(NULL, b)
(g, f)	(b, f)
(f, k)	(f, k)
(k, g)	(k, f)
(g, h)	(f, b)

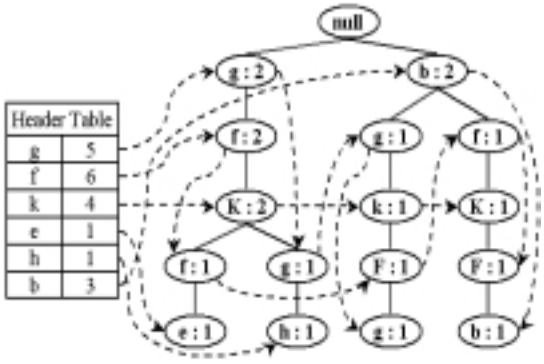

Figure 3. Constructed CMS-tree for Table 3.

Algorithm CMS-tree
Input: a moving logs (ML).
Output: a complete moving sequences tree, CMS-tree.
Method:

```
Procedure CMS-tree (ML)
{
  /*
  Y is used to keep the current maximal  moving sequence. Y and tempC to NULL for ini-
  tialization.
  */
  create a root node for CMS-tree;
  let ptr point to the root node of CMS-tree;
  for each moving pair (oi, ni) on ML {
    set A = oi and B = ni;
    if ( A == NULL ) {
      if ( tempC == NULL) {
        set tempC = B;
      }
      else {
        insert_tree ( tempC, ptr);
        let ptr point to the root node of
                CMS-tree;
        set Y = NULL and tmpC = B;
      }
    }
    else {
      set P = appearance ( A, Y);
      set Q = appearance ( B, Y);
      if ( P > Q ) { /* A backward to B */
        A = upper_case (A);
        append A to string Y;
        insert_tree ( A, ptr);
      }
      else {
        append A to string Y;
        insert_tree ( A, ptr);
      }
      tempC = B;
    }
  }
}
```

Procedure. appearance
Input: node p and string Y.
output: return L.
Method: call appearance (p, Y).

```
Procedure appearance ( p, Y)
{
  /*
  L is the index for the first position p in Y. If p doesn't appear in Y, then L = |Y| + 1
  */
 set L = 1;
 for each element yi in Y {
   if ( p == yi ) {
     return L;
   }
   add 1 to L;
 }
 return L;
}
```

Procedure. insert_tree
Input: node p and the pointer of one node in CMS-tree, prt.
Output: NULL.
Method:

```
Procedure insert_tree ( p, prt)
{
 if ( ptr has a child N and N.id == p.id ) {
   add 1 to N's count;
 }
 else {
   create a new node N with count 1;
   insert N into the header table (without care about backward);
 }
 let ptr point to N;
}
```

Figure 4. CMS-tree algorithm.

3.2 CMP-Mining

The algorithm of CMP-mining is as shown in Figure 5, which finds out the frequent complete moving paths from the constructed CMS-tree and the specified minimum support by using a recursive approach.

Algorithm CMP-mining
Input: CMS-tree constructed based on Algorithm 1 and a minimum support s
Output: the set of frequent complete moving path CMP
Method:

Procedure CMP-mining (Tree, α) {
for each a_i in the header of Tree {
If (a_i.support ≧ s) {
generate pattern $\beta = a_i \cup \alpha$ with support = a_i.support;
construct β's conditional pattern base and then β's conditional CMS-tree $Tree_\beta$;
if ($Tree_\beta \neq null$) {
call CMP-mining($Tree_\beta$, β);
}
else {
add α to CMP;
} } } }

Figure 5. CMP-mining algorithm.

To illustrate how CMP-mining algorithm works, suppose the minimum support is set as 50% and the CMS-tree has been constructed as in Figure 3. Since the counts of node f exceeds the minimum support, we may want to find out the complete moving paths with node f. This can be easily done by tracking the node link in the header table and the CMS-tree. Consequently, the following five moving sequences can be obtained: <g:2>, <g:0, f:1, K:1>, <b:1, g:1, k:1>, <b:1> and <b:0, f:1, K:1>, where the number following ":" represents the counts of that node. Then, a *conditional CMS-tree* is built by using the obtained moving sequences in a way similar to constructing CMS-tree. The only differences are as follows:

1. Since the final output must be complete moving paths, only the parent nodes of the nodes tracked in CMS-tree or conditional CMS-tree are qualified for insertion into the header table. In the above example, these nodes are <g>, <K>, <k>, <b> and <K>.
2. To differentiate the forward and backward movement, two different nodes will be created in the header table for the same location identifier if they contain forward and backward movement information, respectively.

Figure 6 shows the conditional CMS-tree based on node f. After the recursive execution of CMP-mining algorithm, give frequent complete moving path will be discovered, namely {g,

f}, {f, K, f}, {K, f}, {g, f, k} and {f, k}. Take the path {f, K, f} as example, it indicates that mobile users are likely to start a task from location f, moving through location k, and return to f. Table 4 also shows the final moving paths after executing the step of generating the maximal complete moving paths.

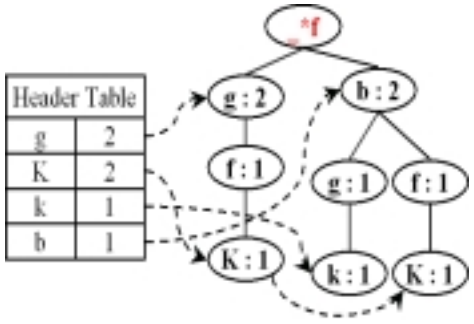

Figure 6. f-based conditional CMS-tree.

Table 4. Maximal complete moving paths.

Maximal complete moving paths
{f, K, f}
{K, f}
{g, f, k}

4 Empirical Results

Several experiments were conducted to evaluate the performance of the CMP-mining method. The main focus is on evaluating the scalability of the CMP-mining method through simulations.

4.1 Simulation Model

Table 5 shows the parameters used in the simulation model. For simplicity, we model the mobile computing system as a 4*4 mesh [11], where there exists a server (VLR) in each node. Since our focus is on discovering complete moving paths from moving log, it suffixes

to consider only the number of input moving sequences and ignore the modeling of number of mobile users. The number of moving sequences is modeled by parameter D, and the parameters *minlen* and *maxlen* control the minimal and maximal lengths of moving path for a mobile user. We use a probabilistic model to simulate the moving behavior of the mobile users. Initially a node server is chosen randomly as the starting location, then an user is assumed to move backward to the last location with probability P_{back}, which is in exponential distribution with unit mean. Moreover, the users are assumed to move forward to other locations with probability P_{other}, which is equal to (1- P_{back}) / (N-1).

Table 5. Parameters for the simulation model.

Parameter	**Meaning**
Network Architecture	4×4 mesh
minlen	minimal length of moving path
maxlen	maximal length of moving path
P_{back}	backward probability for user movement
P_{other}	forward probability for user movement
N	number of reachable server
D	number of moving sequences

Table 6 shows the base settings for the simulation parameters. We varied some major parameters in following experiments for evaluating the performance of the proposed algorithm under different system conditions. Section 4.2 describes the results by varying the data size; the effects of varying the value of minimum support on execution time and the number of discovered paths are given in Section 4.3 and Section 4.4, respectively.

Table 6. Base parameters.

Parameter	Default value
minlen	6
maxlen	10
P_{back}	Exponential distribution
P_{other}	（1－P_{back}）/（n-1）
n	4

4.2 Effect of Varying Data Size

In this experiment, we measure the execution time of CMP-mining under different data size by varying the number of logged moving paths from 200K to 1000K. The other parameters are set as in Table 6 and the minimum support value is set as 0.25%. Figure 7 shows the experimental results, with both the execution time for CMS-tree construction and CMP-mining. It is observed that our method executes scaleably under increased data size, in both part of CMS-tree construction and CMP-mining.

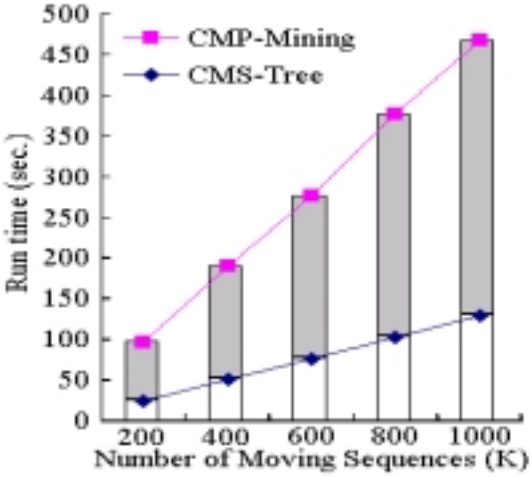

Figure 7. Effect of varying data size.

4.3 Effect of Varying Minimum Support

In this experiment, we measure the execution time of CMP-mining and the number of discovered paths under different settings of minimum support, which was varied from 0.01% to 0.3%. The other parameters are kept the same as in Table 6. Figure 8 shows the experimental results. It is observed that the execution time of CMP-mining increases when the support threshold becomes smaller. In particular, there exists a sharp rise when support threshold is changed from 0.1% to 0.02%. By examining the properties of discovered moving paths, we found that their lengths decrease under smaller support threshold (the average path length varies from 4 to 2 when the support threshold is decreased from 0.1% to 0.02%). Meanwhile, as shown in Figure 9, the number of discovered moving paths increases with

support threshold decreased. In overall, our method still shows scalability under different settings of support threshold.

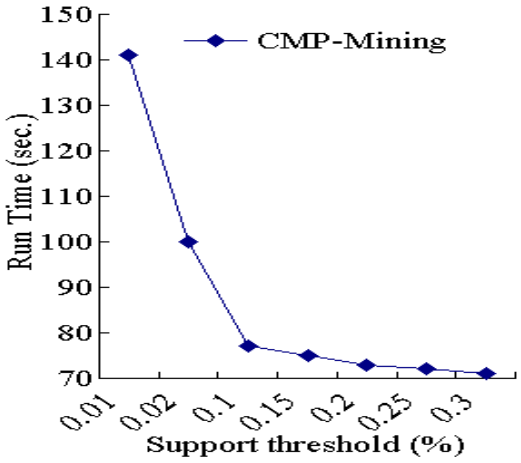

Figure 8. The run time under varied support threshold.

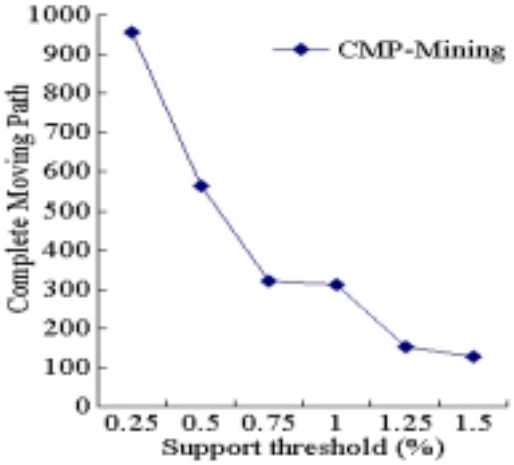

Figure 9. The number of moving paths under varied support threshold.

5 Conclusions and Future Work

A new method, namely *CMP-mining*, is proposed in this research for discovering the complete moving paths of users in a moving environment. By using a special data structure named CMS-tree, the moving sequences of all users can be recorded by reading the dataset

only once, and the frequent complete moving paths can be discovered efficiently based on the constructed CMS-tree. Through empirical evaluation, the proposed method was shown to deliver good scalability under different system conditions like varied data size and support threshold. In future work, we will explore two further issues: 1) The memory problem in constructing CMS-tree when the dataset is extremely large, 2) More detailed performance evaluation of CMP-mining under various conditions in a mobile system.

References

1. R. Agrawal and R.Srikant, "Fast Algorithm for Mining Association Rules in Large Database" In *Proceedings of International Conference on Very Large Data Bases,* pp.478-499, 1994.
2. Ming-Syan Chen, Jiawei Han, and Philip S. Yu, "Data mining : An Overview from a Database Perspective." *IEEE Transactions on Knowledge and Data Engineering*,Vol. 8, No.6, December 1996.
3. M. S. Chen, J. S. Park and P. S. Yu. "Efficient Data Mining for Path Traversal Patterns" *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 2, pp. 209-221, 1998.
4. J. Han, J. Pei, and Y. Yin, " Mining Frequent Patterns without Candidate Generation", In *Proceedings 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00)*, 2000.
5. C. Lee and C. C.Chen, "A Data Delivery Strategy in Ubiquitous Computing Systems" In *Proceedings of the 7th International Conference on Database Systems for Advanced Applications (DASFAA 2001)*,p.p. 210-217,2001.
6. Y.-B. Lin. "Modeling Techniques for Large-Scale PCS Networks." *IEEE Communication Magazine*, 35(2):102-107, February 1997.
7. ELA/TLA, "Cellular Radio Telecommunication Intersystem Operations", 1991.
8. J. Pei, J. Han, Mortazavi-Asl, B., and Zhu, H. "Mining Access Pattern efficiently from Web logs" In *Proceedings of the 2000 Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 396-407, 2000.
9. W. C. Peng and M. S. Chen, "Mining User Moving Patterns for Personal Data Allocation in Mobile computing System" in *Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing*.
10. M. K. Shan and H. F. Li, "Fast Discovery of Structural Navigational Patterns from Web User Traversals" In *Proceedings of SPIE on Knowledge Discovery and Data Mining,* vol. 4730, 2002.
11. N. Shivakumar, J. Jannink, and J. Widom. "Per-User Profile Replication in Mobile Environments: Algorithms, Analysis and Simulation Results," *ACM Journal of Mobile Networks and Applications*, pp. 129-140, 1997.
12 O. Wolfson, S. Jajodia, and Y. Huang. "An Adaptive Data Replication Algorithm," *ACM Transactions on Database Systems*, pp. 253-314, 1997.