EAI Endorsed Transactions
on Context-aware Systems and Applications Research Article ICST.ORG

Mobility Patterns Mining Algorithms with Fast Speed

Giang Minh Duc[1,*], Le Manh[2] and Do Hong Tuan[3]

[1] HCM City University of Technology, HCM City, Vietnam
[2] Van Hien University, HCM City, Vietnam
[3] HCM City University of Technology, HCM City, Vietnam

Abstract

In recent years, mobile networks and its applications are developing rapidly. Therefore, the issue to ensure quality of service (QoS) is a key issue for the service providers. The movement prediction of Mobile Users (MUs) is an important problem in cellular communication networks. The movement prediction applications of MUs include automatic bandwidth adjustment, smart handover, location-based services,… In this work, we propose two new algorithms named the Find_UMP_1 algorithm and the Find_UMP_2 algorithm for mining the next movements of the mobile users. In the Find_UMP_1 algorithm, we make to reduce the complexity of the traditional UMPMining algorithm. In the Find_UMP_2 algorithm, we perform to reduce the number of transactions of the User Actual Paths (UAPs) database. The results of our experiments show that our proposed algorithms outperform the traditional UMPMining algorithm in terms of the execution time. In addition, we also propose the UMP_Online algorithm in order to reduce the execution time as adding new data. The benefit of applying the UMP_Online algorithm is that the system can run online in real time. Therefore, we can perform the applications effectively.

Keywords: *mobility patterns, mobility rules, cellular communication networks, data mining, mobility prediction.*

Received on 14 August 2015, accepted on 27 September 2015, published on 05 November 2015

doi: 10.4108/eai.5-11-2015.150603

1. Introduction

Currently, due to the rapid development of mobile communication networks, many people use personal mobile devices to search for information on the Internet. Almost everyone has a mobile device as cell phone, personal digital assistant (PDA) or notebook. In addition, many people search for information while traveling all over the world. At about 6.8 billion mobile phones are used around the world in 2013 at the rate of 96, 97% of the world population [1]. Therefore, the propose problem is how to ensure quality of mobile services.

In cellular communication networks [2], a mobile user can move from one location to another which neighboring cell in the network. When MUs move like that, the location of mobile users will be constantly updated to Visitor Location Register (VLR) ([3], [4], [5]) of the system. VLR is an intermediate database in order to store temporary information about mobile users in the service area of Mobile Switching Center (MSC). The location information of MUs then is transferred to home location register (HLR). The

Corresponding author. Email: ducgm.bdg@vnpt.vn

HLR is a database which long-term storage of information of MUs. The movement history of MUs is extracted from the log files and it is stored in the HLR of the MSC. The historical data is used to predict the mobility of MUs.

Due to the properties of the cellular communication networks are mobility, disconnection, long time delay, hand-off, bandwidth continuously changing... so there were some recent researches, which applied the traditional User Mobility Pattern (UMP) Mining algorithm ([7], [8], [9], [10]) to overcome these problems. However, the UMP Mining algorithm has a long execution time, running offline. Therefore, the above applications are reduced effectiveness.

In particular, our main contributions can be summarized as follows:

- Our proposed algorithms make increased running speed of the traditional UMP Mining algorithm in two ways. (1) We perform to reduce the complexity of the traditional UMP mining algorithm. (2) We reduce a number of transactions when movements mining of MUs.
- We propose UMP_online algorithm to avoid scanning of full database again. This algorithm executes to mine the new dataset (new transactions are added to the database). Therefore, the mobile service providers (MSPs) can supply their applications more efficiently.
- The results of our experiments show that:

- Execution time of the first improvement (Find_UMP_1 algorithm) reduces more than 25% compared with the traditional UMPMining algorithm.
- Execution time of the second improvement (Find_UMP_2 algorithm) reduces more than 75% compared with the traditional UMPMining algorithm.
- The third improvement (UMP_Online algorithm) has an execution time down about 57.94% compared with the Find_UMP_2 algorithm.

The rest of our paper is organized as follows. In section 2, we present related work. In section 3, our proposed scheme is explained. Finally, we present the experimental results in Section 4 and conclude our work in section 5.

2. Related work

Problem mining sequential patterns mentioned in [6], [7], [8], [9]. The algorithm in [6] applied Apriori algorithm in grid computing and does not take into the topology of the network while creating the candidate patterns. In [14], Mira H. Gohil and S. V. Patel compared different methods of the next location prediction.

The UMPMining algorithm in [7] predicts the next location of mobile users using data mining techniques. In [7], Yavas et al presented an AprioriAll based sequential pattern mining algorithm to find the frequent sequences and to predict the next location of the user. They compared their algorithm's results with Mobility Prediction based on Transition Matrix (TM). In [10], Byungjin Jeong applied the UMPMining algorithm to perform the decision smart handover for the purpose of reducing the number of unnecessary handover in architecture Macro / Femto-cell networks.

In [11] and [12], the authors also applied the UMPMining algorithm for location-based services (LBSs) in the cellular communication networks. In [11], Abo-Zahhad et al. presented LBSs as emergency, safety, traffic management, and public information applications. In [12], Lu et al., presented to find segmenting time intervals where similar mobile characteristics exist.

The above works apply the traditional UMPMining algorithm to enhance quality of services. Our algorithms improve the traditional UMP algorithm to further enhance the quality of services.

3. Proposed scheme

Before giving our proposed algorithms, we present the traditional UMP Mining algorithm in [7] and calculate its complexity in subsection A as follows.

3.1 The traditional UMPMining algorithm:

Suppose that User Actual Paths (UAPs) have a form as follows: C = {c_1, c_2... c_n}. Each c_k denotes the ID number of the cell k_{th} in coverage area.

For example, we have the coverage map simulated as follows:

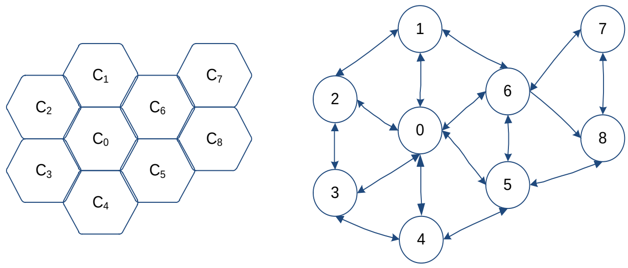

Figure 1. The simulation of the cellular network and graph G

The data of UAPs as follows:

Table 1. Paths of mobile users

UAP ID	UAPs
1	{5,6,0,4}
2	{3,4,5,0}
3	{1,2,3,4,0,5}
4	{3,2,0}

G is called a directed graph corresponds to cells in the mobile coverage area. Each cell of the G is a node as Fig. 1. If there are two cells that called A, B neighboring each other (a common border) in the mobile coverage area, they have a directed and unweighted edge from A to B and from B to A.

Definition 1:

Suppose that there are two UAPs, A = $\{a_1, a_2, ... a_n\}$ and B = $\{b_1, b_2 ... b_m\}$. B is a substring of A, if exist: $1 \le i_1 < ... < i_m \le n$, $b_k = a_{ik}$, $\forall k$, and $1 \le k \le m$.

In addition, B is a substring of A, if all cells of B exist in A (not need sequent in A).

For example, in Figure 1, suppose that A = $\{c_4, c_0, c_6, c_7, c_8, c_5\}$ and B = $\{c_6, c_8\}$ is the length-2 sequence of A. In addition, The UAP B is contained by the UAP A.

The UMPMining algorithm is a sequence pattern mining algorithm which applied in the movement predict of the cellular networks ([7], [8], [9], [10], [11], [12]).

UMPMining algorithm

Input: UAPs of database D, min_supp, graph G

Output: L (UMPs)

1. $C_1 \leftarrow$ the length-1 patterns
2. k = 1
3. L = $\varnothing$ // initially the set is empty
4. **while** $C_k \ne \varnothing$
5. **for each** (UAP $a \in D$) do
6. S = {s | $s \in C_k$ and s is subsequence of a}
7. **for each** $s \in S$ do
8. s.count = s.count + s.suppInc
9. **endfor**
10. **endfor**
11. L_k = {s | $s \in C_k$, s.count $\ge$ min_supp }
12. L = L $\cup$ L_k
13. $C_{k+1} \leftarrow$ Cand_Gen (L_k, G), $\forall c \in C_{k+1}$, c.count = 0
14. k = k + 1
15. **endwhile**
16. **return L**

At line 13, the Cand_Gen() function is written as follows:

Cand_Gen algorithm

Input: L_k; G

Output: Candidates (C_{k+1})

Candidates = $\varnothing$ //initially the candidates set is empty
for each L = $(l_1, l_2, …, l_k)$, $L \in L_k$ do
  N^+ = {v | $l_k \rightarrow v$ }
  for each $v \in N^+$ (l_k) do
    C' = $(l_1, l_2, …, l_k, v)$
    Candidates $\leftarrow$ *Candidates* $\cup$ *C'*
  endfor
endfor
return *Candidates*

For UMPMining algorithm, from line 5 to line 10 (finding support of C_n) is rewritten as follows:

Find_support_UMP(S_k)

Input: database D

Output: SP(S_k) (support of S_k)

1. **for each** (UAP $a \in D$) **do** //scan all database D
2. **for** ($i = 1$; $i \le |a|$; i++) **do** //|a|: length of sequence a
3. Find position $(s_1, s_2, …, s_k) \in S_k$ in sequence a
4. Find S_k.count
5. **endfor**
6. **endfor**
7. **return** SP(S_k)

The complexity of the Find_support_UMP function:

- For the loop at line 1: the complexity is O(m), where m = |D|
- For the second loop (line 2): the complexity is O(n), where n = | a |: the average length of string $a \in D$.
- Thus, the complexity of this algorithm is: $O(m \times n)$.

In order to reduce the complexity of the UMP Mining algorithm we perform steps as follows:

3.2. Find_UMP_1 algorithm

We map the UAPs database (D) to the M_{dd} Mobility Matrix (definition 6).

Steps as follows:

Definition 2: Data Mining Context

Let O be a non-empty limited set of transactions (UAP ID) and I be a non-empty limited set of cells, R be a two subject relation between O and I such that $o \in O$ and $i \in I$, $(o,i) \in R \Leftrightarrow$ transaction o contains cell i_{th}. The data mining context is the triple (O, I, R).

Definition 3: Data Mining Context Matrix

Give a *mobile user's paths* table includes two properties that are UAP_ID (code of a transaction) and UAP (path of a mobile user through the cells of the mobile coverage map). Call O is a set of transactions. I is a set of cells and R is a two subject relation between O and I, $R \subseteq O \times I$, where $(o, i) \in R$ if and only if transaction o is contained cell i_{th}.

Definition 4: Galois Connection

Give a data mining context (O, I, R), where two functions ρ and λ, they are defined as follows: ρ P(I) → P(O) and λ P(O) → P(I):

Give $S \subseteq I$, $\rho(S) = \{o \in O \mid \forall i \in S, (o,i) \in \mathbf{R}\}$

Give $X \subseteq O$, $\lambda(X) = \{i \in I \mid \forall o \in X, (o,i) \in \mathbf{R}\}$

Where P(X) is a set of subsets of X. A pair of function (ρ,λ) is defined in such that is called Galois Connection.

ρ(S) value denotes a set of transactions that have common all cells in S. λ(X) The value denotes a set of cells that have in all transactions of X.

Property 1: a pair of function (ρ, λ) has properties as follows:

1.1 Where $S_1, S_2 \in P(I)$, $S_1 \subseteq S_2 \Rightarrow \rho(S_2) \subseteq \rho(S_1)$

1.2 Where $X_1, X_2 \in P(O)$, $X_1 \subseteq X_2 \Rightarrow \lambda(X_2) \subseteq \lambda(X_1)$

1.3 $S \subseteq \lambda(\rho(S))$ and $X \subseteq \rho(\lambda(X))$

1.4 $\lambda(\rho(\lambda(X))) = \lambda(X)$ and $\rho(\lambda(\rho(S))) = \rho(S)$

Definition 5: the frequent set

Give a data mining context (O, I, R), and $S \subset I$, the frequency level of S is defined as the ratio of the number of transactions to all of the transactions. The frequent of S is called the support of S (SP(S)) and it is computed as follows:

$$SP(S) = |\rho(S)| / |O|$$

Where | . | is the length of the set.

Give $S \subset I$ and min_supp is a minimum support threshold, S is a support set by the min_supp threshold if and only if $SP(S) \geq$ min_supp.

FS (O, I, R, min_supp): is the set of the support subsets satisfy the min_supp threshold or FS (O, I, R, min_supp) = $\{S \in P(I) \mid SP(S) \geq min\_supp\}$

Clause 1:

Give $S \in$ FS(O, I, R, min_supp), if $T \subseteq S$, then $T \in$ FS(O, I, R, min_supp)

Demonstration: due to $T \subseteq S$, according to property (1.1) of the Galois Connection of a pair of function (ρ, λ), we have $\rho(S) \subseteq \rho(T)$, therefore $min\_supp \leq SP(S) \leq SP(T) \rightarrow T \in$ FS(O,I,R,min_supp).

Clause 2:

Give $T \notin$ FS(O, I, R, min_supp), if $T \subseteq S$, then $S \notin$ FS(O, I, R, min_supp).

Demonstration: due to $T \subseteq S$, according to property (1.1) of the Galois Connection of a pair of function (ρ, λ), we have $\rho(S) \subseteq \rho(T)$, therefore $SP(S) \leq SP(T) < min\_supp \rightarrow S \notin$ FS(O,I,R,min_supp).

Definition 6: M_{dd} mobility Matrix

The M_{dd} mobility matrix is similar to the binary matrix as definition 3, but it is added as follows: each $M[O_m, i_n]$ is a location of a mobile user traveling in mobile network (Table 2).

Column i_n: code of a cell in mobile network.

Row o_m: the actual paths of a mobile user.

We exchange data from the table 1 to table 2 as follows:

Table 2. Mobility matrix of mobile users

	i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7
o_1	3	0	0	0	4	1	2	0
o_2	4	0	0	1	2	3	0	0
o_3	5	1	2	3	4	6	0	0
o_4	3	0	2	1	0	0	0	0

For example, in table 2, mobile user 2 (UAP ID = 2) moves between the cells as follows:

Figure 2. Mobility of UAP ID = 2

Therefore, the path o_2 performs the following: o_2 = (4, 0, 0, 1, 2, 3, 0, 0)

The following is a new algorithm to find UMPs from the mobility matrix M_{dd}:

Find_UMP_1 algorithm

Input: min_supp, M_{dd}, G
Output: L

1. $L = \varnothing$ // initially the large patterns set is empty
2. $L_1 \leftarrow$ Find_L_1 //generate L_1 from Find_L_1 function
3. **for** ($k=2$; $L_{k-1} \neq \varnothing$; k++) do
4. $\quad L_k \leftarrow$ Find_$L_k(L_{k-1})$ //generate L_k from L_{k-1}
5. $\quad L = L \cup L_k$
6. **endfor**
7. **return** L

At line 2, we have Find_L_1() function as follows:

Find_L_1 algorithm

Input: (O, I, R), *min_supp*, M_{dd}, G
Output: L_1.

1. $L_1 = \varnothing$
2. **for each** ($i \in I$ and $j \in$ field of M_{dd}) //i: cell ID and it is also a column of M_{dd}
3. $\quad S=\{s \mid s \in M_{dd}$ and $s_{ij} \neq 0\}$
4. $\quad$ **for each** $s \in S$
5. $\quad\quad$ s.count = s.count + 1
6. $\quad$ **endfor**
7. **endfor**
8. $L = \{s \mid s \in C_1$, s.count $\geq$ min_supp$\}$
9. $L_1 = L_1 \cup L$
10. **return** L_1

At line 4 of Find_UMP_1 algorithm, we have a function finds L_k from L_{k-1} as follows:

Find_$L_k(L_{k-1})$ algorithm

Input: L_{k-1}, G, M_{dd}

Output: L_k

1. $L_k = \varnothing$
2. **for** (each $X \in L_{k-1}$) do
3. $\quad$ **for** (each $Y \in L_{k-1}$ and $X \neq Y$) do
4. $\quad\quad S = X \cup Y$
5. $\quad\quad S = \{s_1, s_2, \ldots, s_{k-1}, s_k\}$ //s_k: a set of cells be linked to s_{k-1} of G
6. $\quad\quad$ **if** ($|S| = k$ and SP(S) $\geq$ min_supp) then
7. $\quad\quad\quad L_k = L_k \cup \{S\}$
8. $\quad\quad$ **endif**
9. $\quad$ **endfor**
10. **endfor**
11. **return** L_k

At line 6 of Find_L_k(), we have a function finds the support of S_k as follows:

Find_support(S_k) algorithm

Input: S_k, M_{dd}
Output: $SP(S_k)$

1. **for each** $o \in M_{dd}$ do //scan all M_{dd}
2. $\quad$ Find location $(s_1, s_2, \ldots, s_k) \in S_k$ of $o \in M_{dd}$
3. $\quad$ Find S_k.count
4. **endfor**
5. **return** SP(S_k)

- The complexity of the Find_support() algorithm:

- For the loop at line 1: the complexity is O(m), where m = |O|: the total number of records of M_{dd}
- Thus, the complexity of the algorithm is: O(m).

The complexity of the Find_support algorithm is reduced n times (reduce of one loop) compared to UMPMining algorithm.

3.3. Find_UMP_2 algorithm

The Find_UMP_2 algorithm is similar to the Find_UMP_1 algorithm, they differ from the function to find the support, as follows:

- Decreasing the number of transactions:

According to the clause 2, we have:

$T \notin FS(O, I, R, min\_supp)$, if $T \subseteq S$, then $S \notin FS(O, I, R, min\_supp)$.

Find_Supp_2(S_k) algorithm

Input: M_{dd}, S_k, *min_supp*, G
Output: $SP(S_k)$

1. Dem_dong = 1
2. **if** $|S_k| = 2$ then
 //scan all rows of M_{dd} ($o_n \in M_{dd}$)
3. $\quad$ **for** ($i = 1$; $i \leq |O|$; i++) do
4. $\quad\quad$ **if** ($S_k \in O$ and $(s_1, s_2, \ldots, s_k)$ have in order of O) then
5. $\quad\quad\quad$ Store_Array $\leftarrow$ save variable i
6. $\quad\quad\quad$ Store_Array $\leftarrow$ count the number of rows
7. $\quad\quad$ **endif**
8. $\quad$ **endfor**
9. **else** //$|S_k| > 2$
10. $\quad S_{k-1} \leftarrow S_k$
11. $\quad |O_R| \leftarrow$ Store_Array //$|O_R|$: the number of rows contains S_{k-1}
12. $\quad$ **for** ($i = 1$; $i \leq |O_R|$; i++) //$|O_R| < |O|$
13. $\quad\quad$ **if** ($S_k \in O_R$ and $(s_1, s_2, \ldots, s_k)$ have in order of O_R) then
14. $\quad\quad\quad$ Store_Array $\leftarrow$ save variable i
15. $\quad\quad\quad$ Store_Array $\leftarrow$ count the number of rows
16. $\quad\quad$ **endif**
17. $\quad$ **endfor**
18. **endif**
19. SP(S_k) $\leftarrow$ Find support
20. **if** SP(S_k) $\geq$ min_supp then
21. $\quad$ Store_Array $\leftarrow$ Temp_Array
22. **endif**
23. **return** SP(S_k)

Remarks:

- When $|S_k| = 1$, the method calculating the support of the Find_support (S_k) and the Find_support_2 (S_k) algorithm is the same, so the execution time of the two algorithms are equal (as shown table 6).
- When $|S_k| = 2$, the method calculating the support of the Find_support_2 (S_k) algorithm is added line 4 ÷ 7, 20 ÷ 22 with the following meanings:

- If $SP(S_k) \geq$ min_supp, we save these rows to the Store_Array including the number of satisfied rows and S_k.supp.
  Our purpose reduces the number of the loop time as finding S_{k+1}. According to clause 2, we show that: if $S_k \notin FS(O, I, R, min\_supp)$ and $S_k \subseteq S_{k+1}$, then $S_{k+1} \notin FS(O, I, R, min\_supp)$. For example, if $S_k = \{3, 2\}$ and $SP(S_k) = 1 \leq min\_supp = 1.33$, then $S_{k+1} = \{3, 2, 1\} \leq$ min_supp.

- This algorithm was executed for an actual database as follows:
  Input data UAPs have the number of paths as: 56 198 (all rows of matrix M_{dd}: $|O| = 56\ 198$).
  The number of BTSs is 351 (The number of fields of matrix M_{dd}: $|I| = 351$).

- When $|S_k| \geq 2$:

- At line 10 ÷ 11: get the number of rows contained S_{k-1} be O_R to reduce the number of the loop.
- At line 12 ÷ 17: instead of scanning of full database, we just execute the O_R loop times.

3.4. UMP_Online algorithm

In this section, we develop the incremental algorithms to find the large sets from the mobile database. The proposed algorithm is named UMP_Online. In order to avoid scanning of full database again, this algorithm executes to mine the new dataset (new transactions are added to the database).

The purpose of this algorithm is to reduce the execution time of mining the MUs movements. Therefore, MSPs can supply their applications more efficiently.

Here is the UMP_Online algorithm:

UMP_Online algorithm

Input: New candidate sets have length-i: C_{inew}

Old candidate sets have length-i: C_i

Old large sets have length-i: L_i; min_supp

Output: New large set: L

```
for each (c ∈ Cinew)
  if c ∈ Ci then
    supptotal = s.supp + c.supp    //s ∈ Ci và s = c
    s.supp = supptotal
    if supptotal >= min_supp then
      if c ∈ Li then
        l.supp = supptotal //l ∈ Li; l = c = s
      else // c ∉ Li
        Li = Li ∪ c
        Find_Lk()
      endif
  else //  c ∉ Ci
      Ci = Ci ∪ c
      if (c.supp >= min_supp) then
        Li = Li ∪ c
        Find_Lk()
      endif
  endif
  endif
endfor
return L
```

The old result table is the candidate sets C_i and the large patterns L_i (C_i, L_i are found as running the Find_UMP_2 algorithm). This algorithm uses the previous results and takes update to the mobility patterns as follows:

- Finding the candidate patterns (C_{inew}) from the new data set.
- The support value is calculated for each sequence $c \in C_{inew}$, if the min_supp value is satisfied.
- Update the candidate patterns (if c.supp $\geq$ min_supp) to the old candidate patterns (C_i, L_i).
- Returning the new large set: L.

Due to the UMP_Online algorithm returns the result which be set of the large sets L, we should prove that the set

L finding enough all L_i. When Li gives rise (line 9 and 15), the algorithm calls the Find_L_k() function (line 10 and 16).

Theorem 1: the Find_L_k algorithm ensures to find enough all keys.

Using the inductive method to prove the Find_L_k algorithm that ensures to find all keys.

First, L_1 is true because $L_1 = \{S \in P(I) \mid SP(S) \geq minsupp \wedge |S| = 1\}$

Suppose that L_{k-1} is true, we should prove the Find_L_k algorithm creates L_k true. That is L_k contained all large sets S, so that | S | = k.

Indeed, due to set $X \in F_{k-1}$ and $Y \in F_{k-1}$, so $|X| = |Y| = k-1$. In addition to wanting $S = X \cup Y$ is a candidate, then $|S| = k$ (line 6 of the Find_L_k algorithm). According to clause 2, the set $S \in L_k$ must be the large sets → L_k candidates should be created from L_{k-1} (line 2, line 3 of the Find_L_k function).

3.5. Finding the mobility rules

According to the results from the data mining phase (UAPs → UMPs); the mobility patterns of mobile users (UMPs) were founded. In this section, we will find the mobility rules from UMPs.

Example: we have a form UMP is (3, 4, 5). The mobility rules as follows:

(3) → (4, 5)

(3, 4) → (5)

Suppose that we have the UMP $L = \{i_1, i_2, \ldots i_k\}$, where $k > 1$. All mobility rules are generated from the pattern as follows:

$\{i_1\} \text{ -> } \{i_2, \ldots, i_k\}$

$\{i_1, i_2\} \text{ -> } \{i_3, \ldots, i_k\}$

…

$\{ i_1, i_2, \ldots, i_{k-1}\} \text{ -> } \{ i_k \}$

Give the mobility rule R is: $(i_1, i_2, \ldots, i_{m-1}) \rightarrow (i_m, i_{m+1}, \ldots, i_k)$, the confidence value is calculated as follows:

$$Confidence(R) = \frac{(i_1, i_2, \ldots, i_k).count}{(i_1, i_2, \ldots, i_{m-1}).count} \times 100$$

By using UMPs, all mobility rules are generated and the confidence value is also calculated. Rules (if confidence ≥ min_conf) will be selected.

. Finding the mobility rules:

We have the rules generation algorithm as follows:

Gen_Rules algorithm

Input: Minimum confidence value: min_conf
User mobility patterns: UMPs
Output: Set of mobility rules: R

1. **for all** L $\in$ *UMPs*, L = $(i_1, i_2, \ldots, i_k)$, where k > 1 **do**
2. **for all** m from 1 to k - 1 **do**
3. //get all the mobility rules
4. *head* $= (i_1, i_2, \ldots, i_{m-1})$
5. *tail* $= (i_m, i_{m+1}, \ldots, i_k)$
6. *rule = head → tail*
7. //calculate the confidence value of the rule
8. *rule.conf* = (*L.count/head.count*) * 100
9. **if** *rule.conf* ≥ *min_conf* **then**
10. *R = R ∪ rule*
11. **end if**
12. **end for**
13. **end for**
14. **return *R***

At line 4, the head is the part of the rule before the arrow. At line 5, the tail is the part of the rule after the arrow (rule = head → tail).

After running the Gen_Rules algorithm, we have the results table from actual data as follows (min_conf = 5%):

Table 3. The rules result

The rules:

SN	Head	Tail	Confidence
0	1	7	11.8
1	1	12	18.8
2	1	17	36.6
3	3	5	10.5
4	3	17	7.3
5	3	26	25.3
6	4	9	28.5
7	4	23	6.6
8	5	3	13.9
9	5	16	12.1
10	7	1	19.4
11	7	17	8.9
…………………………			
917	53,61	88	9.6
918	53,66	52	75.8
919	54,63	56	6.5
920	54,63	59	5.4
921	54,63	79	36.1
922	54,63	88	8.9
923	54,65	69	12.5
924	54,79	63	64.7
925	56,63	54	64.6
926	56,63	88	50.2
927	56,64	54	99.3
928	57	99,85	6.5
…………………………			

. *The movement prediction of mobile users:*

From the above rules results (set of mobility rules R), we find out the set of predicted cells as follows:

Pre_Mov algorithm

Input: Current movement of the user: $P = (c_1, c_2, \ldots, c_i)$

Set of mobility rules: R.

Output: Set of predicted cells: Pre_Cells.

```
Pre_Cells = ∅        // assign set Pre_Cells = ∅
n = 1               // n: cardinal number of Rule_Array
for each r: (i1, i2,…, ij)→(ij+1,…, ik)∈ R do
  if (i1, i2, …, ij) ⊆ P and ij = ci then
    Pre_Rule ← r
    Rule_Array[n] ← (Pre_Rule, confidence value)
    n = n + 1
  endif
endfor
Sort (Rule_Array) // in descending order with
                  // respect to confidence value.
for i = 1 to n do
   Pre_Cells ← Right_cell  //get the first cell that is
     // on the right side of each rule in Rule_Array
    n = n + 1
endfor
return Pre_Cells
```

In this part, the next movement of user is predicted. Suppose that the movement of a user (up to now) is P = (64, 56, 63). Current this user is being cell 63 of the coverage region. The algorithm finds out the rules as follows: (56, 63) → (54) and (56, 63) → (88) (line 925 and 926 of Table 3). The set of predicted cells is {54, 88} (both cell 54 and cell 88 are selected). The cell 54 is selected first because it has the confidence value more than the confidence value of the cell 88 (64.6 and 50.2).

4. The experimental results

In this section, we investigate the performance of our Find_UMP_1, Find_UMP_2, UMP_Online algorithms and compare them with the performance of the traditional UMPMining algorithm in terms of the execution time.

Our experimental environments are given in Table 4. Training data set and Testing data set used form [13].

Training data set: the number of UAPs. Training data sets include three sets given in Table 5.

Table 4. Experimental environments

Name	Parameter
Processor	Intel Core i3-2330M, 2.20GHz
RAM	4 GB
Operating System	32-bit
Programming language	Microsoft Visual Studio 2005
Database management system	SQL Server 2005

Table 5. Training data sets

Name	Number of transactions of MUs
Data set 1	56198
Data set 2	68787
Data set 3	34895

These datasets are the actual database of MUs. The database is transformed from the User ID to the integer n (n = 1, 2, 3,...) and they cannot be decoded to protect customer information.

- The testing dataset is UAPs; it is used to evaluate the accuracy of the users' mobility prediction.

Testing data set contains 7207 transactions of MUs.

The number of BTS: 351.

4.1. Compare the execution time of the algorithms: UMPMining, Find_UMP_1 and Find_UMP_2

The data set is performed for comparison: data set 1.

In table 6, the execution time of the Find_UMP_1 algorithm is 410 seconds and the execution time of the UMPMining algorithm is 548 seconds (down 25.18%). While the execution time of the Find_UMP_2 algorithm is only 136 seconds (down 75.18% as compared to the UMPMining and 66.82% as compared to the Find_UMP_1 algorithm).

Table 6. The results of three algorithms

C_n	UMPMining		Find_UMP_1		Find_UMP_2	
	quantity C_n	Run time	quantity C_n	Run time	quantity C_n	Run time
C_1	351	32	351	1	351	1
C_2	1488	167	1488	129	1488	129
C_3	3340	341	3340	274	3340	5
C_4	79	8	79	6	79	1
Total	**5258**	**548**	**5258**	**410**	**5258**	**136**

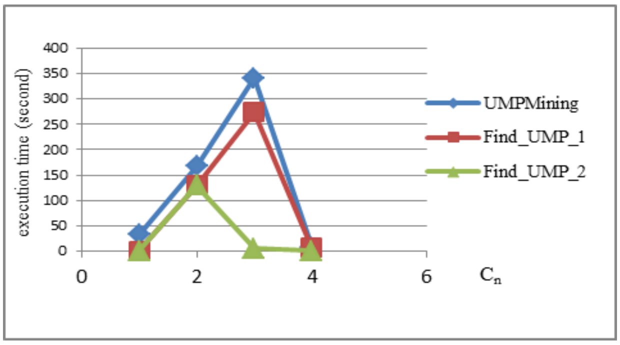

Figure 3. The execution time results of three algorithms

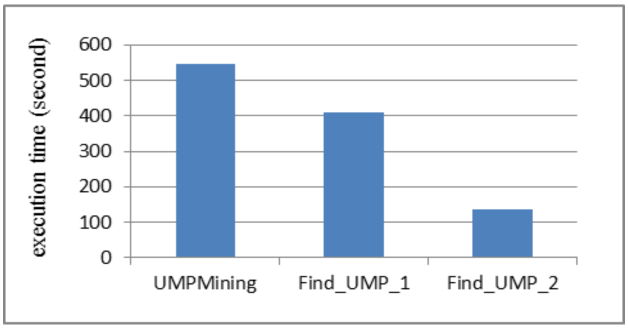

Figure 4. The execution time total of three algorithms

4.2. The experimental results when running the UMP_Online algorithm

- When not applying the algorithm UMP_Online:

Each update a new data set, we perform as follows:

- Database (total) = database (old) + database (new)
- Running the Find_UMP_2 algorithm for database (total).

- When applying the algorithm UMP_Online:

We perform as follows:

- Get the old results (Cn, Ln).
- Running the UMP_Online algorithm for the new database and update the result with the old results → new results.

To compare the results of the two methods above, we have the actual results as follows:

- Database (old): data set 1 (the number of records is 56 198).
- Database (new) data set 2 (the number of records is 68 787).
- Database (total): data set 1 + data set 2 (the number of records is 124 985)
- The execution time is 214 seconds.

When running UMP_Online algorithm, the execution time is 90 seconds (down 57.94%).

The same as above, we have:

- Database (old): data set 1 (DS1) + data set2 (DS2) (the number of records is 124 985)
- Database (new): data set 3 (the number of records is 34 895)
- Database (total): data set 1 (DS1) + data set 2 (DS2) + data set 3 (DS3) (the number of records is 159 880)
- The execution time of the Find_UMP_2 algorithm is 284 seconds.

The execution time of the UMP_Online algorithm is 95 seconds (down 66.54%).

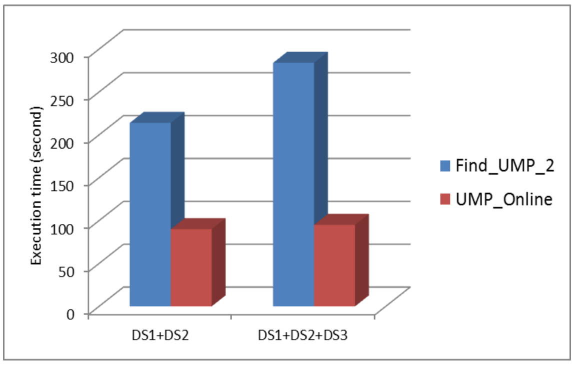

Figure 5. The execution time of two algorithms

4.3. The accuracy of the prediction:

- Recall: the number of correctly predicted cells / the total number of requests.
- Precision: the number of correctly predicted cells / the total number of predictions made.

- Changing of the recall values according to the min_supp values:

In Figure 6, if the min_supp value increases, then the recall value decreases. The reason is the increasing min_supp value will make the number of prediction rules decreased. Therefore, the number of correctly predictions is decreased.

Figure 6 compares the recall value changes of three data sets.

When the size of the training set increases, the recall values also increase (because the number of prediction rules increases).

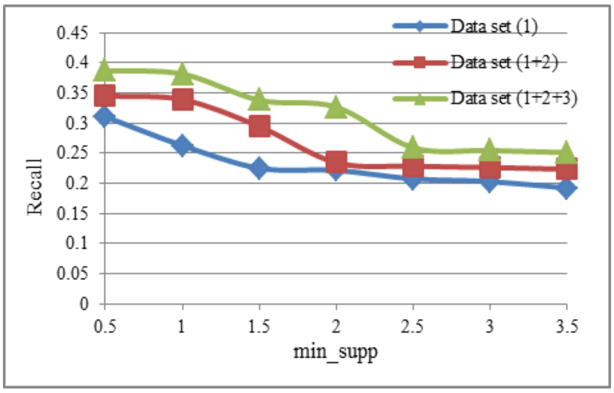

Figure 6. Changes of recall according to min_supp of three data sets

- The precision of the prediction rules when changing the min_conf value:

Testing data set: 7207 records.

When changing the minimum confidence value (min_conf), the precision value changes as Figure 7.

In Figure 7, when the min_conf value increases, the precision value also increases. Because of high min_conf values, only the rules that have high confidence values are used for prediction.

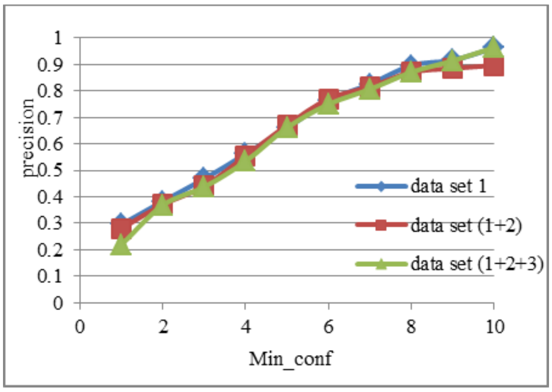

Figure 7. Precision of the prediction rules

5. CONCLUSION

The mobility prediction of Mobile Users is one of the important issues in mobile computing systems. Applications of the MUs mobility prediction are adjusting bandwidth of the networks, the location-based services, smart handover, ... However, these applications require the execution time of the UMPMining algorithm as quickly as possible. In this work, we proposed Find_UMP_1 algorithm and the Find_UMP_2 algorithm to solve the time problem. The results of our experiments shown that our proposed algorithms outperform the traditional UMPMining algorithm in terms of the execution time.

In addition, we also propose the UMP_Online algorithm in order to reduce the execution time as adding new data. The benefits of applying this algorithm are that the system can run online in real time. Therefore, MSPs can perform the above applications effectively.

References

[1] "List of Countries by Number of Mobile Phones in Use," Wikipedia, Gartner, 2010 .

[2] ETSI/GSM. Technical reports list. http://webapp.etsi.org/key/key.asp? full list=y.

[3] ETSI/GSM. Home location register/visitor location register – report 11.31–32.

[4] Alex Cabanes (IBM Systems & Technology Group): IBM BladeCenter - Home Location Register (HLR). June 2007.

[5] HRL Look Up – Service Manual (www.routomessaging.com).

[6] Cristian Aflori and Mitica Craus. "Grid implementation of Apriori algorithm. Advances in engineering software". *Volume 38, Issue 5*, 295-300, 2007.

[7] Gokhan Yavas, dimitrios Katsaros. Ozgur Ulssoy and Yannis manolopoulos. "A data mining approach for location prediction in mobile environments". Data and Knowledge Engineering, 54, 121-146, 2005.

[8] Mr. Mohammad Waseem, Mrs. R.R.Shelke, Location Pattern Mining of Users in Mobile Environment, International Journal of Electronics, Communication & Soft Computing Science and Engineering, 2013, ISSN: 2277-9477, Volume 2, Issue 9

[9] V. Chandra Shekhar Rao and P. Sammulal. Article: Survey on sequential pattern mining algorithms. International Journal of Computer Applications, 76(12):24–31, August 2013. Published by Foundation of Computer Science, New York, USA.

[10] Byungjin Jeong, Seungjae Shin, Ingook Jang, Nak Woon Sung, and Hyunsoo Yoon, "A Smart Handover Decision Algorithm Using Location Prediction for Hierarchical Macro/Femto-Cell Networks " in Vehicular Conference (VTC Fall), 2011 IEEE 74th, SanFrancisco, CA, Sept 2011, pp. 1-5.

[11] M. Abo-Zahhad, Sabah M. Ahmed, M. Mourad, " Services and Applications Based on Mobile User's Location Detection and Prediction", Int. J. Communications, Network and System Sciences, 2013, 6, 167-175.

[12] Lu, E. H.-C.; Tseng, V. S.; and Yu, P. S. 2011. "Mining cluster-based temporal mobile sequential patterns in location-based service environments". IEEE Trans. Knowl. Data Eng. 23(6):914–927.

[13] http://msdn.microsoft.com/en-us/ library/ bb895173.aspx (Training and Testing Data Sets – MSDN – Microsoft).

[14] Mira H. Gohil and S. V. Patel, "Mobile Location Prophecy: An analytical review". IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501 Vol.4, No5, October 2014