Developing Data Allocation Schemes by Incremental Mining of User Moving Patterns in a Mobile Computing System

Wen-Chih Peng and Ming-Syan Chen, *Senior Member*, *IEEE*

Abstract—In this paper, we present a new data mining algorithm which involves incremental mining for user moving patterns in a mobile computing environment and exploit the mining results to develop data allocation schemes so as to improve the overall performance of a mobile system. First, we propose an algorithm to capture the frequent user moving patterns from a set of log data in a mobile environment. The algorithm proposed is enhanced with the incremental mining capability and is able to discover new moving patterns efficiently without compromising the quality of results obtained. Then, in light of mining results of user moving patterns and the properties of data objects, we develop data allocation schemes that can utilize the knowledge of user moving patterns for proper allocation of both personal and shared data. By employing the data allocation schemes, the occurrences of costly remote accesses can be minimized and the performance of a mobile computing system is thus improved. For personal data allocation, two data allocation schemes, which explore different levels of mining results, are devised: one utilizes the set level of moving patterns and the other utilizes the path level of moving patterns. As can be seen later, the former is useful for the allocation of read-intensive data objects, whereas the latter is good for the allocation of update-intensive data objects. The data allocation schemes for shared data, which are able to achieve local optimization and global optimization, are also developed. Performance of these data allocation schemes is comparatively analyzed. It is shown by our simulation results that the knowledge obtained from the user moving patterns is very important in devising effective data allocation schemes which can lead to significant performance improvement in a mobile computing system.

Index Terms—Data mining, mobile computing, user moving patterns, data allocation scheme, mobile database.

1 Introduction

Due to recent technology advances, an increasing number of users are accessing various information systems via wireless communication. Such information systems as stock trading, banking, and wireless conferencing, are being provided by information services, and application providers [13], [19], and mobile users are able to access such information via wireless communication from anywhere at any time [4], [9], [29].

For cost-performance reasons, a mobile computing system is usually of a distributed server architecture [13], [19], in which a service area, referring to the converge area where the server can provide services to mobile users, contains one or many cells where a cell refers to a communication area covered by a base station. In general, mobile users tend to submit transactions to servers nearby for execution so as to minimize the communication overhead incurred [13], [19] [28]. The properties of data objects accessed by mobile users can usually be divided into two types: the read-intensive type (or read type) and the update-intensive type (or update type). Data objects are assumed to be stored at servers to facilitate coherency control and also for memory saving at mobile units [31], [34]. Since the architecture of a mobile computing system is distributed in nature, data replication is helpful because it is able to improve the execution performance of servers and facilitate the location lookup of mobile users [17], [28], [31], [34]. The replication scheme of a data object involves how many replicas of that object to be created and to which servers those replicas are allocated. Clearly, though avoiding many costly remote accesses, the approach of data replication increases the cost of data storage and update. Thus, it has been recognized as an important issue to strike a compromise between access efficiency and storage cost when a data allocation scheme is devised.

It is noted that various data allocation schemes have been extensively studied in the literature [31], [34]. However, the data allocation schemes for traditional distributed databases are mostly designed in static manners and the user moving patterns, which are particularly relevant to a mobile computing system where users travel between service areas frequently, were not fully explored. As mentioned above, the server is expected to take over the transactions submitted by mobile users and static data allocation schemes may suffer severe performance problems in a mobile computing system. An example scenario is given in Fig. 1, where without loss of generality, we assume that the network topology of a mobile computing system is a four by four mesh topology. Suppose that the data are replicated statically at sites A, F, K, and P under the data allocation schemes for traditional distributed databases, and the mobile user U_1 is found to frequently travel in service areas

- *The authors are with the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan ROC.*
E-mail: wcpeng@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw.

Manuscript received 12 Jan. 2000; revised 18 Feb. 2001; accepted 26 Nov. 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 111213.

1041-4347/03/$17.00 © 2003 IEEE Published by the IEEE Computer Society

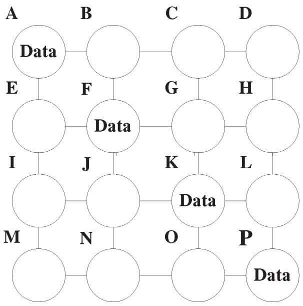


Fig. 1. The network architecture of mobile computing systems.

of A, B, and C (i.e., {A, B, C} is called the moving pattern of mobile user U_1). It can be seen that the advantage of having replicas on F, K, and P cannot be fully taken by mobile user U_1, and the extra cost of maintaining those replicas is not justified by the moving pattern of user U_1. In order to improve the system performance, efficient data allocation schemes based on moving patterns of mobile users are very important in a mobile computing environment. This is the very problem we shall address in this paper. More justification on the problem studied can be found in Section 2.3. Consequently, we shall explore in this paper the approach of mining user moving patterns in a mobile computing environment and utilize the mining results to develop data allocation schemes. Note that the data allocation schemes devised not only utilize the mining results but also consider the properties of data objects for improving the overall performance of a mobile system.

Specifically, for the development of data allocation schemes, we shall first devise an algorithm to capture the frequent user moving patterns from a set of log data (referred to as *movement log*) in a mobile environment. It is worth mentioning that to fully explore the temporal locality, which refers to the feature that consecutive movements of a mobile user are likely to fall into similar sites [11], the mining algorithm devised is enhanced with an incremental mining capability in the sense that it is able to focus on the recent moving patterns within an adjustable window size (referred to as *retrospective factor*) when deriving user moving patterns. As such, by avoiding generating user moving patterns for every single movement of individual mobile users, the frequent user moving patterns can be obtained much more efficiently without compromising the quality of results obtained.

Then, in light of mining results of user moving patterns, we devise data allocation schemes that can utilize the knowledge of user moving patterns for proper allocation of both personal data (referring to those data only accessible by each individual data owner) and shared data (referring to those data to be accessed by a group of users). By employing the data allocation schemes devised, the occurrences of costly remote accesses can be minimized and the performance of a mobile computing system is thus improved. For personal data allocation, two data allocation schemes, which explore different levels of mining results and properties of data objects, are devised: One utilizes the *set level* of moving patterns and the other utilizes the *path level* of moving patterns, where the set level refers to the set of moving patterns and the path level refers to the ordered sites for prefetching determined from the moving patterns. As can be seen later, the former is useful for the allocation of read data objects, whereas the latter is good for the allocation of update data objects. The data allocation schemes for shared data, which are able to achieve local optimization and global optimization, are also developed. Performance of these data allocation schemes is comparatively analyzed and sensitivity analysis on several design parameters, including the retrospective factor, is conducted. It is shown by our simulation results that the knowledge obtained from the user moving patterns is very important in devising effective data allocation schemes which can lead to significant performance improvement in a mobile computing system.

Various data mining capabilities have been explored in the literature [5]. One of the most important data mining problems is mining association rules in transaction databases [1], [2] [7], [21], [27]. Also, mining classification rules is an approach to develop rules that can efficiently classify data items based on certain features [25], [30]. Mining sequential patterns is the study to explore the knowledge of ordered data [3], [14]. Though dealing with various mining capabilities, these prior results were not applicable to mining user moving patterns in a mobile environment. In [6], an algorithm for mining Web user traversal patterns was proposed. With the assumption that the backward reference is made for ease of traveling but not for data access in a Web environment, the mining algorithms and results in [6] were mainly developed subject to that assumption, which is, however, not applicable to mining user moving patterns in a mobile environment where all user movements have to be taken into consideration to truly reflect the user moving patterns. More importantly, the mining algorithm in [6] neither deals with data allocation nor has the capability of incremental mining which is particularly important in a mobile environment. The very difference between these two environments calls for the design of a new algorithm for mining user moving patterns in a mobile environment. More justifications for the problem studied can be found in Section 2.

In addition, a significant amount of research efforts has been elaborated upon issues of data allocation in distributed systems [10], [13], [16], [31], [34]. We mention in passing that the authors of [34] proposed a data distribution scheme that is based on the read/write patterns of the data objects. Given some user calling patterns, the authors of [31] proposed an algorithm that employed the concept of minimum-cost maximum-flow to compute the set of sites where user profiles should be replicated. Without fully exploiting user moving patterns, the attention of the studies in [31], [34] was mainly paid to the distribution of location data for mobile users, but not to the personal and shared data allocation that are explored in this paper.

The contributions of this paper are twofold. We not only devise a new data mining algorithm for incremental mining of user moving patterns in a mobile computing system, but also in light of the mining results obtained and the properties of data objects, develop data allocation schemes to improve the overall performance of a mobile computing

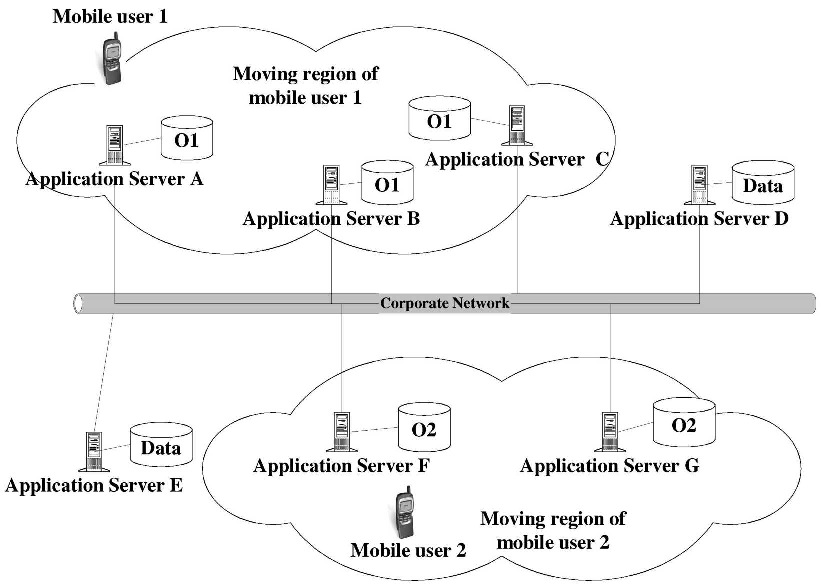


Fig. 2. An example of emerging wireless application services.

system. With incremental mining capability, frequent moving patterns of good quality can be obtained efficiently. Moreover, by exploring different levels of knowledge on user moving patterns, different data allocation schemes, which take the properties of data objects into consideration, are devised and their performance is comparatively analyzed. To the best of our knowledge, prior work neither fully explored the data mining capability for user moving patterns in a mobile computing system nor utilized such mining results for personal and shared data allocation, let alone devising schemes to exploit different levels of knowledge mined and conducting the corresponding performance analysis. These features distinguish this paper from others. With the rapid advances in wireless technologies, the mobile computing systems are becoming widely available nowadays. The fast increase in mobile applications justifies the timeliness and importance of this study.

This paper is organized as follows: Preliminaries are given in Section 2. Algorithms for mining user moving patterns in a mobile system are devised in Section 3.1, data allocation schemes based on user moving patterns for personal data allocation are developed in Section 3.2, and those for shared data allocation are developed in Section 3.3. Simulation results are presented and analyzed in Section 4. This paper concludes with Section 5.

2 Preliminary

To facilitate the presentation of this paper, some preliminaries are given in this section. To justify the problem studied, some real applications of wireless data access are described in Section 2.1. A location management method and the generation of movement log used for mining user moving patterns in a mobile computing system are described in Section 2.2. The usefulness of moving patterns for the design of efficient location management is described in Section 2.3.

2.1 Applications of Wireless Data Access

Various wireless data networking technologies, including IS-136 [32], CDMA2000 [18], and Wireless Application Protocol (WAP) [33], have been developed recently. Among others, WAP provides an open standard for connecting Internet content and advanced value-added services to mobile phones. For example, an emerging WAP calendar application [33] allows a mobile user to access his/her own calendar data through a mobile phone. Such calendar data of an individual mobile user is an example of personal data considered in Section 3.2 of this paper. On the other hand, corporate applications, such as sales force automation in which salepersons can instantly obtain the latest pricing and competitive information for their customers, access many data commonly used by many mobile users. These commonly used data are examples of shared data considered in Section 3.3 of this paper.

As pointed out earlier, for cost performance reasons, an application system which provides wireless data access services is of a distributed server architecture and the use of data replication is helpful in that the amount of remote access can thus be reduced, which in turn conserves the energy of mobile units. Consider an illustrative example in Fig. 2, where application servers provide calendar services. Suppose that O_1 and O_2 are the data objects which contain the calendar information, respectively, for mobile user 1 and mobile user 2. If data object O_1 is replicated at sites A, B, and C, where mobile user 1 frequently travels, then the amount of remote access and the communication overhead by mobile user 1 can be reduced since O_1 can be obtained locally. Clearly, how to select proper sites for data allocation is a very important issue and will benefit from effective methods for discovering user moving patterns which will be dealt with in this paper.

2.2 Generation of Movement Log

Note that in a mobile environment each mobile user is associated with a home location database which maintains

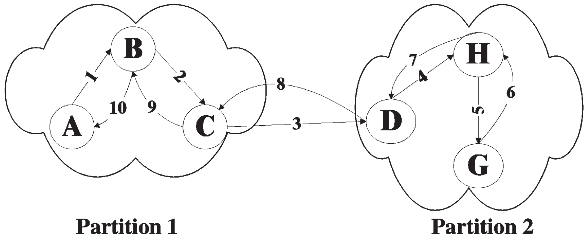


Fig. 3. An example of moving patterns in a mobile computing system.

an up-to-date location data for the mobile user. The location management procedure for a mobile computing system considered in this paper is similar to the one in IS-41/GSM [12] [22], which is a two level standard and uses a two-tier system of home location register (to be referred to as HLR) and visitor location register (to be referred to as VLR) databases. Each mobile user is associated with an HLR, whose database maintains recent mobile users' records and their current locations. When a mobile user moves out the area maintained by its HLR, a copy of the mobile user's record is created in its local VLR. In addition, the record in the HLR is updated to reflect the movement of that user. This procedure is referred to as registration.

In order to capture user moving patterns, a movement log is needed. A movement log contains a pair of (old VLR, new VLR) in the database when registration occurs. In the beginning of a new path, the old VLR is null. For each mobile user, we can obtain a moving sequence $\{(O_1, N_1), (O_2, N_2), ...(O_n, N_n)\}$ from the movement log. Generally speaking, each node in the network topology of a mobile computing system can be viewed as a VLR and each link is viewed as the connection between VLRs. With such a mobile system considered, we propose algorithms for incremental mining of user moving patterns and utilize the mining results to conduct proper data allocation in a mobile computing system.

Consider an illustrative example in Fig. 3,[1] where the number next to each link represents the sequence of movements of a mobile user. Thus, the movement log contains the moving path for that mobile user. As pointed out before, with the assumption that backward travels are made for ease of traveling and, hence, need not be considered, algorithm MF in [6] will terminate the discovery of maximal moving sequences when any backward reference occurs. The set of maximal moving sequences for this moving path output by algorithm MF is $\{ABCDHG\}$.[2] However, such a fragmented moving sequence is of little interest in a mobile environment where one would naturally like to know the complete traveling information, i.e., $\{ABCDHGHDCBA\}$ in this case, showing the very difference between these two environments. Note that by taking into consideration the backward travels, nodes may appear more than once in the same maximal moving sequence and it is necessary to extract these sequences, and take their occurrences into account when the corresponding user moving patterns are evaluated. Clearly, to deal with these problems, as well as to solve such important issues as incremental mining and data allocation which are particularly relevant to a mobile environment, it is essential to develop a new algorithm for mining user moving patterns in a mobile computing system.

2.3 Mining Moving Patterns for Location Management

It is worth mentioning that, in addition to providing many benefits in several database applications, mining user moving patterns is particularly important in a mobile computing system where the corresponding registration and hand-off overhead are costly. Here, we explain this advantage of mining user moving patterns for location management. As described before, each mobile user is mapped to an HLR, whose database always maintains most recent mobile user's record which includes current location. When the mobile user moves out of the area maintained by his/her HLR, the mobile user needs to inform HLR about the movement and HLR updates the database to reflect the mobile user's new location. However, the massive updates incurred will degrade the overall system performance. Several previous research efforts have been elaborated upon this update problem of HLR [15], [24]. The authors in [15] proposed a location management scheme in which mobile users do not have to inform the HLR for every single movement if the movement of the mobile user is within a bounded partition, where a bounded partition is defined according to user moving patterns. Consider the scenario in Fig. 3, for example. Assume, partition 1 and partition 2 are user moving patterns of the mobile user. If each movement of the mobile user needs to inform HLR, there will be 10 update operations in HLR. In contrast, the location management scheme in [15] will incur an update only if the mobile user moves out of a partition (instead of "moves out of a site"), in which case, there are only three update operations in HLR incurred for the scenario in Fig. 3 (i.e., 1 (initial update at site A in partition 1) + 1 (when the mobile user moves out of partition 1 in the 3th movement)+1 (when the mobile user moves out of partition 2 in the 8th movement) = 3), thus reducing the overall overhead. However, it is important to note that the methods to discover user moving patterns for the determination of partitions, which are, in our opinion, very important in the solution procedure, were absent in [15]. Clearly, user moving patterns are beneficial on developing efficient location management and querying strategy in a mobile computing system [15], [26], thereby justifying another motivation of this study.

3 Data Allocation Schemes Based on Moving Patterns

In this section, we develop a solution procedure, which is composed of a sequence of algorithms in the corresponding steps, to mine user moving patterns and improve data allocation in a mobile computing system. Specifically, with the user movement log given, we devise algorithms for incremental mining of moving patterns in Section 3.1. We

[1] Fig. 3 was borrowed from [15] for illustrative purposes.

[2] The example scenario described here and in Table 1 is to illustrate the difference between algorithm MF in [6] and algorithm MM in this paper. In order not to distract reader from the main theme of this paper for data allocation, readers interested in the details of algorithm MF are referred to [6].

develop an algorithm for determining maximal moving sequences (to be referred to as algorithm MM) in Section 3.1.1. Then, according to the maximal moving sequences determined, we devise an algorithm to identify large moving sequences (to be referred to as algorithm LM) in Section 3.1.2 in an incremental manner. The scenario for incremental mining of moving patterns is illustrated in Section 3.1.3.

In light of the user moving patterns determined, we develop data allocation schemes for personal data allocation in Section 3.2 and for shared data allocation in Section 3.3. Explicitly, a fix data allocation scheme (to be referred to as scheme DF) is presented in Section 3.2.1 first. By considering two levels of mining results, two data allocation schemes for personal data allocation are devised: one utilizes the set level of moving patterns (to be referred to as scheme DS), which favors the read data object, in Section 3.2.2 and the other utilizes the path level of moving patterns (to be referred to as scheme DP), which favors the update data objects, in Section 3.2.3. Note that the set level refers to the set of moving patterns and the path level refers to the ordered sites for prefetching determined from the moving patterns. Data allocation schemes based on moving patterns for shared data allocation (to be referred to as algorithm SD-local and SD-global) are devised in Section 3.3.

3.1 Incremental Mining for Moving Patterns in a Mobile Environment

Once the movement log is generated, we shall convert the log data into multiple subsequences, each of which represents a *maximal moving sequence*. After maximal moving sequences are obtained, we then map the problem of finding frequent moving patterns into the one of finding frequent occurring consecutive subsequences among maximal moving sequences. A sequence of k movements is called a *large k-moving sequence* if there are a sufficient number of maximal moving sequences containing this k-moving sequence. Such a threshold number is called a support in this paper. Note that after large moving sequences are determined, moving patterns can then be obtained in a straightforward manner. A moving pattern is a large moving sequence that is not contained in any other moving patterns. For example, suppose that {AB, BC, AE, CG, GH} is the set of large 2-moving sequences and {ABC, CGH} is the set of large 3-moving sequences. Then, the resulting user moving patterns are {AE, ABC, CGH}. User moving patterns are associated with the areas that users frequently travel in a mobile computing system. The overall procedure for mining moving patterns is outlined as follows:

Procedure for incremental mining of moving patterns.

- **Step 1 (Data collection phase)**. Employing algorithm MM (to be described in Section 3.1.1) to determine maximal moving sequences from a set of log data and also the occurrence count of moving pairs.
- **Step 2 (Incremental mining phase).** Employing algorithm LM (to be described in Section 3.1.2) to determine large moving sequences for every w maximal moving sequence obtained in Step 1, where w is the retrospective factor which is an adjustable window size for the recent maximal moving sequences to be considered.
- **Step 3 (Pattern generation phase).** Determine user moving patterns from large moving sequences obtained in Step 2, where user moving patterns are those frequent occurring consecutive subsequences among maximal moving sequences.

Note that in the data collection phase, the occurrence counts of moving pairs are updated online during registration procedure. For purposes of efficiency, algorithm LM is executed to obtain new moving patterns in an incremental manner for every w maximal moving sequence generated, where the unit of w is the number of maximal moving sequences. The selection of w will be determined empirically in Section 4 later. As users travel, their moving patterns can be discovered incrementally to reflect the user moving behaviors.

3.1.1 Finding Maximal Moving Sequences

As pointed out earlier, a moving pair, (old VLR, new VLR), is generated in a movement log for each registration procedure. Given a moving sequence $\{(O_1, N_1), (O_2, N_2), ...(O_n, N_n)\}$ of a user, we shall map it into multiple subsequences, each of which represents a maximal moving sequence. First, we can obtain a moving sequence $\{(O_1, N_1), (O_2, N_2), ...(O_n, N_n)\}$ for each mobile user from the movement log, where pairs of (O_i, N_i) are sorted by time. Then, algorithm MM (standing for maximal moving sequence), whose algorithmic form is given below, is applied to moving sequences of each mobile user to determine the maximal moving sequences of that user and update the occurrence count of moving pairs during registration procedure.

In algorithm MM, Y is used to keep the current maximal moving sequence and F is a flag to indicate if a node is revisited. Let D_F denote the database to store all the resulting maximal moving sequences. Also, S is the home location site of a mobile user. According to the roundtrip model considered [20], [31], the selection of S is either VLR or HLR whose geography area contains the homes of mobile users. In order to capture the complete traveling sequence, algorithm MM outputs a maximal moving sequence to D_F until the S is reached. In line 1 of algorithm MM, some parameters are initialized. Then, moving sequences are scanned in line 2. A maximal moving sequence is output and a new maximal moving sequence will be explored (from line 14 to line 18 of algorithm MM) if MM finds that N_i in the moving pair (O_i, N_i) is the same as the starting site S. Otherwise, N_i is appended into Y (in line 12 of algorithm MM) and the occurrence count of (O_i, N_i) is updated online in the database (in line 14 of algorithm MM). An example execution scenario by algorithm MM for the input in Fig. 3 is given in Table 1. As mentioned earlier, algorithm MM is different from algorithm MF in that unlike the latter, the former neither outputs a maximal moving sequence when backward travels are encountered (such as in the 6th move) nor updates the occurrence count of (O_i, N_i) online. Instead, algorithm MM will keep tracing the user movements until the starting point is reached and update the occurrence count of (O_i, N_i) online so as to reduce the overhead of database scan for generating large moving pairs.

TABLE 1
An Example Execution by Algorithm MM

Move	String Y by algorithm MF	Maximal moving sequences output by algorithm MM
1	AB	AB
2	ABC	ABC
3	ABCD	ABCD
4	ABCDH	ABCDH
5	ABCDHG	ABCDHG
6	-	ABCDHGH
7	-	ABCDHGHD
8	-	ABCDHGHDC
9	-	ABCDHGHDCB
10	-	ABCDHGHDCBA

```
Algorithm MM/* Algorithm MM for finding maximal
               moving sequences */
Input: A moving sequence $\{(O_1, N_1), (O_2, N_2), ...(O_n, N_n)\}$
     of a mobile user.
Output: Maximal moving sequences of the mobile user.
begin
Set $i$ to 1 and string $Y$ to null , where $Y$ is used to keep the
     current maximal moving sequence and S is the
     starting point.
while (not end of movings)
    begin
    Set $A = O_i$ and $B = N_i$;
      if ($A == S$ )
         begin
            Set Y=S;
            Append $B$ to $Y$;
         end
      else
         begin
            Append $B$ to string $Y$;
              Update the occurrence count of (A,B)
                 in database $D_F$;
            if ($B == S$ )
              begin
                 Output string $Y$ to database $D_F$;
                 Set $Y$ to null;
              end
         end
      i++;
    end
end
```

3.1.2 Finding Large Moving Sequences

With the maximal moving sequences obtained, we next determine the large moving sequences. A large moving sequence can be determined from all maximal moving sequences of each individual user based on its occurrences in those maximal moving sequences. Use *intrasequence count* to mean the number of occurrences of a moving sequence within a maximal moving sequence and *intersequence set* of a moving sequence to mean the set of maximal moving sequences which contain that moving sequence. The count of a large moving sequence is the sum of intrasequence counts from its intersequence set. For the example in Table 2, the intrasequence count of GB in {ABCGBCGBA} is 2 and that in {ABGBA} is 1. Also, the intersequence set of GB is {{ABCGBCGBA}, {ABGBA}}. Hence, the count of GB is the sum of intrasequence counts in its intersequence set (i.e., 2 (i.e., intrasequence count in ABCGBCGBA) + 1 (i.e., intrasequence count in ABGBA) = 3). To cope with this problem, we develop algorithm LM (standing for large moving sequence) for the determination of large moving sequences. Let L_k represent the set of all large k-moving sequences and C_k be a set of candidate k-moving sequences.

```
Algorithm LM/* Algorithm for finding large moving
              sequences */
Input: A set of $w$ maximal moving sequences
     of a mobile user.
Output: Large moving sequences of the mobile user.
begin
Determining $L_2$= {large 2-moving sequence}
                from moving pairs in $C_2$;
for $(k = 3; L_{k-1} \neq 0; k++)$
   begin
      $C_k = L_{k-1} * L_{k-1}$;  /* Generating $C_k$ from
                                $L_{k-1} * L_{k-1}$ */
      for $w$ maximal moving sequence $S$
         begin/* Calculating the intrasequence count
                    of $C_k$ within S */
         intrasequence = subsequence$(C_k, S)$;
         if (intrasequence$> 0$)
            Including S into intersequence set;
          /* sum of occurrence counts in a
                intersequence set */
          for all candidate $c \in$ intersequence
              $c$.count $= c$.count $+ c$. intrasequence;
         end
      $L_k = \{c \in C_k | c.\text{count} \geq \text{support}\}$;
   end
end
```

As pointed out in [27], the initial candidate set generation, especially for L_2, is the key issue to improve the performance of data mining. Since occurrence counts of moving pairs, i.e., C_2, were updated online in the data

TABLE 2
An Example for Counting the Occurrences of 2-Moving Sequences

C_2	Intrasequence counts		Total count from
	ABCGBCGBA	ABGBA	the intersequence set
AB	1	1	2
BC	2	-	2
CG	2	-	2
BG	-	1	1
GB	2	1	3

collection phase, L_2 can be determined by proper trimming on C_2 efficiently (line 1 of algorithm LM), showing the advantage of having online update in algorithm MM. Also, note that C_k can be simply generated from $L_{k-1} * L_{k-1}$ (line 4 of algorithm LM). For example, with the set of L_2 being {AB, BK}, we have a C_3 as {ABK}. As explained above, the occurrence count of each k-moving sequence is the sum of intrasequence counts (from line 5 to line 9 in algorithm LM) in its intersequence set (i.e., line 10 and line 11 in algorithm LM). Note that this step is very different from that in mining the path traversal patterns [6] where there are no loops in a moving sequence (i.e., the corresponding intrasequence count is always zero). The occurrences of each k-moving sequence in C_k are determined for the identification of L_k. After the summation of the occurrence counts in the intersequence set from line 10 to line 11 in algorithm LM, those k-moving sequences with counts exceeding the support are qualified as L_k (line 13 of algorithm LM). Notice that those large k-moving sequences are obtained from w maximal moving sequences of that mobile user, showing the incremental mining capability of algorithm LM. For illustrative purposes, with the maximal moving sequences of a mobile user being

$$\{ABCGBCGBA, ABGBA\},$$

Table 2 shows the corresponding counts of C_2.

3.1.3 An Illustrative Example for Incremental Mining of Moving Patterns

Consider an illustrative example for two consecutive runs of incremental mining of moving patterns in Fig. 4, where the network topology of a mobile computing system is the one in Fig. 1. Fig. 4a shows the run i where the retrospective factor w is set to 10. In the data collection phase, algorithm MM determines maximal moving sequences and counts the occurrences of moving pairs. Using the w*i maximal moving sequences generated by MM, LM determines, in the incremental mining phase, large moving sequences. In our illustrative example, we set the value of the support to be 4 and assume that the value of i is 1. It can be verified that the set of L_2 is {{NJ}, {NM}, {MN}} and the set of L_3 is {NMN}. In the pattern generation phase, moving patterns (i.e., $\{NJ, NMN\}$ in Fig. 4a) are determined from large moving sequences obtained in the incremental mining phase (i.e., L_2 and L_3 in this illustrative example). Following the same procedure, moving patterns (i.e., $\{NJ, ON, POPO\}$ in Fig. 4b) are generated incrementally as the mobile user travels. By having the advantage of online updating the occurrence counts of C_2 in algorithm MM, algorithm LM devised can be effectively applied to explore all the moving paths and generate user moving patterns. Note that due to the temporal locality of the user moving behavior, the identities of mining patterns discovered may vary as the time window advances. It can be verified that the procedure of mining user moving patterns is of polynomial time complexity. Specifically, with n moving pairs in a moving sequence, the complexity of algorithm MM is $O(n)$. Similarly to many other algorithms in the large itemset generation, the complexity of algorithm LM is in proportion to the number of L_K. As pointed out in [27], the initial candidate set generation, especially for L_2, is the key issue for the performance improvement of mining process. It is worth mentioning that since occurrence counts of moving pairs, i.e., C_2, were updated online in the data collection phase, L_2 can be efficiently determined by proper trimming on C_2, showing an effective feature of the proposed procedure of mining moving patterns.

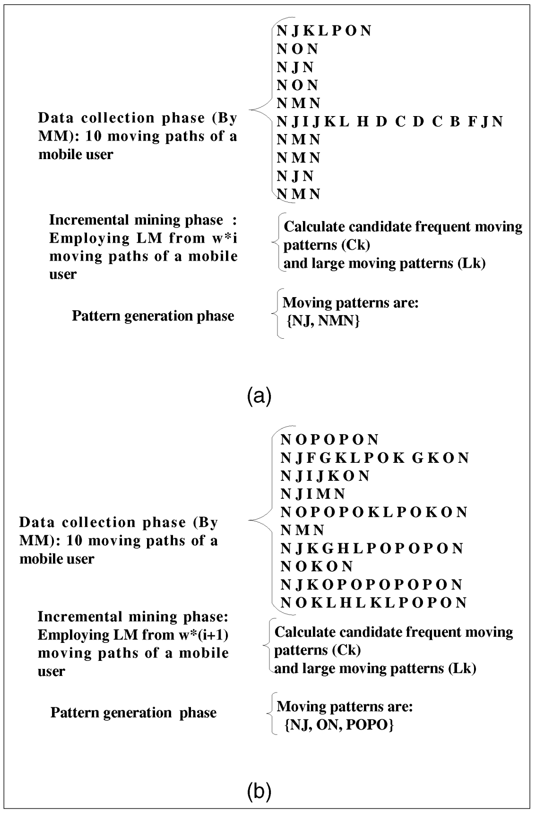

Data collection phase (By MM): 10 moving paths of a mobile user
N J K L P O N
N O N
N J N
N O N
N M N
N J I J K L H D C D C B F J N
N M N
N M N
N J N
N M N
Incremental mining phase : Employing LM from w*i moving paths of a mobile user
Calculate candidate frequent moving patterns (Ck) and large moving patterns (Lk)
Pattern generation phase
Moving patterns are: {NJ, NMN}
(a)
Data collection phase (By MM): 10 moving paths of a mobile user
N O P O P O N
N J F G K L P O K G K O N
N J I J K O N
N J I M N
N O P O P O K L P O K O N
N M N
N J K G H L P O P O P O N
N O K O N
N J K O P O P O P O P O N
N O K L H L K L P O P O N
Incremental mining phase: Employing LM from w*(i+1) moving paths of a mobile user
Calculate candidate frequent moving patterns (Ck) and large moving patterns (Lk)
Pattern generation phase
Moving patterns are: {NJ, ON, POPO}
(b)

Fig. 4. Two consecutive runs of incremental mining moving patterns. (a) Run i where $w = 100$. (b) Run $i + 1$ where $w = 10$.

As mentioned before, after large moving sequences are determined, moving patterns can be obtained in a straightforward manner. Moving patterns of a mobile user correspond to frequent movements of this mobile user in a mobile computing system. Thus, as will be explored in Section 3.2 below, in light of mining results obtained from algorithms MM and LM, we can utilize the knowledge of user moving patterns to devise effective data allocation schemes so as to improve the overall performance of a mobile computing system. Since the knowledge of user moving patterns is discovered incrementally, the data allocation schemes are thus able to dynamically change data replicated servers to fully exploit the moving patterns of mobile users. Therefore, if the mobile user changes his/her moving behavior, the data allocation schemes devised will be able to adaptively determine the new replication plan by using the incremental mining algorithm for moving patterns devised.

3.2 Schemes for Personal Data Allocation

As mentioned above, the occurrences of remote accesses can be reduced if data allocation schemes are devised in light of user moving patterns. In this section, we propose efficient data allocation schemes based on user moving patterns for personal data allocation, where personal data (e.g., personal calendar data described in Section 2.1) are those data only accessible by each data owner. We first describe in Section 3.2.1 the data allocation scheme in a fix pattern. Then, we devise two data allocation schemes based on two levels of user moving patterns, explicitly the set level and the path level, in Section 3.2.2 and in Section 3.2.3, respectively.

3.2.1 DF (Data Allocation Scheme in a Fixed Pattern)

In the scheme which allocates data in a fix pattern (referred to as scheme DF), the replication sites are determined when the database is created. Explicitly, the number of replicated sites and the sites at which the personal data can be replicated are predetermined. Though being adopted in some traditional distributed database systems due to its ease of implementation [34], scheme DF is not suitable for mobile computing environments where mobile users move frequently. Consider the example mobile computing system in Fig. 1 and assume that the replicated servers under scheme DF are {AFKP}. As explained before, if mobile user U_1 is found to frequently travel in service areas of A, B, and C, then the advantage of having replicas on F, K, and P is not exploited by user U_1. In our experimental studies in Section 4, scheme DF will be implemented and evaluated for comparison purposes.

3.2.2 DS (Data Allocation Scheme Based on the Set of Moving Patterns)

As described before, by exploring different levels of mining results, two data allocation schemes are developed: one utilizes the set level of moving patterns and the other utilizes the path level of moving patterns. In this section, the data allocation scheme based on the set of moving patterns (referred to as scheme DS) is described. Scheme DS takes advantage of moving patterns so as to reduce the number of remote accesses. Consider the example profile in Table 3 where the network architecture is the one in Fig. 1. The set of replicated servers under scheme DS is the set of servers determined from moving patterns of mobile users. For example, the set of the replicated servers for U_1 is the union of moving patterns of U_1 (i.e., {AE} ∪ {ABC} = {ABCE}) and that of U_2 is {BCGF}. In addition, as will be shown in Section 4, scheme DS is able to not only increase the hit ratio of local data access but also balance the workloads of servers since more replicated servers are employed under scheme DS than under scheme DF.

However, according to scheme DS, more replicated sites are employed as the number of sites appearing in user moving patterns increases, which, as will be seen in Section 4, though reducing the occurrences of remote access significantly, may result in too many sites replicated, thus compromising the overall performance improvement achieved. In order to eliminate the cost of maintaining replicated sites, we shall take the properties of data objects into consideration. Clearly, for a read data object, data replication is preferable in order to increase the likelihood of local reads. As a result, scheme DS favors those read-intensive data objects. On the other hand, for an update data object, scheme DS is not appropriate since scheme DS results in too many sites replicated, increasing the costs of update and communication. In view of this, we shall investigate the approach of utilizing the mining results in the path level, instead of the set level as in scheme DS. This will allow us of employing the technique of prefetches properly and ensuring the number of replicated sites not to exceed a predetermined limit (Such a limit is called prefetch size). Using the path level knowledge, as a mobile user moves, the personal data can be prefetched to the candidate sites predicted from user moving patterns and the advantage of mining user moving patterns can be fully exploited. This is the very motivation that the design of scheme DP is based on.

TABLE 3
An Example Profile for Illustrating Data Distribution Schemes

User ID	Moving patterns	Number of movements
U_1	AE, ABC	1500
U_2	BCGF	350
U_3	BCD	300
U_4	CGK	200

3.2.3 DP (Data Allocation Scheme Based on the Path of Moving Patterns)

Assume that a mobile user U_i, who is currently at site A, has a moving path ABCD. Clearly, before U_i moves into site B, fetching the data required to site B will be able to reduce the response time of later access. This technique is in general referred to as prefetching. In this section, a data allocation scheme based on the paths of moving patterns (referred to as scheme DP) is employed to determine the candidate sites for proper prefetching. An algorithmic form of scheme DP is given below. With the example input in Table 3, the scenario for local access hit of U_1 under DP is shown in Fig. 5, where it can be seen that the candidate sites of U_1 for prefetches are those candidate sites for prefetching predicted from moving patterns of U_1. For example, when U_1 is in the site A at first, scheme DP searches user moving patterns of U_1 to identify and then eliminate those patterns which do not contain the current site. In scheme DP, P_f is used to ensure the number of replicated sites not to exceed a predetermined limit denoted by P_{fetch}. If the number of user moving patterns are included then, P_f, is less than P_{fetch}, scheme DP will use the current site A as the key to search user moving patterns of U_1 for finding the candidate user moving patterns. Those candidate user moving patterns are used in the path level to prefetch data into the sites to which the mobile user is likely to move. As such, it can be verified that {AE} and {ABC} are the candidate user moving patterns found from line 3 to line 13 in scheme DP. After obtaining the candidate user moving patterns, scheme DP will determine the candidate sites for prefetching (i.e., the sites which U_1 may move from site A), which are sites B and E in

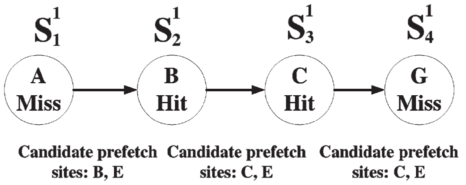

S_1^1
S_2^1
S_3^1
S_4^1
A Miss
B Hit
C Hit
G Miss
Candidate prefetch sites: B, E
Candidate prefetch sites: C, E
Candidate prefetch sites: C, E

Fig. 5. An illustrative example for local data access hits by U_1 under scheme DP.

this case (in line 14 of scheme DP). The candidate prefetch sites are determined accordingly.

Scheme DP:

```
Input: The current site, user moving patterns and $\mathrm{P}_{fetch}$.
Output: The candidate sites for prefetching.
begin
1. Initial the set of prefetch sites $P_f$, where $|P_f|$
        is the number of prefetch sites included thus far;
2. Search the current site in user moving patterns;
3. while $(\mathrm{P}_{fetch} > |P_f|)/ * \mathrm{P}_{fetch}$ is the predetermined
        prefetch size */
4.     begin
5.         Search user moving patterns of the
               corresponding user by using current
               site as the key;
6.         if (found)
7.             begin
8.                 Include the site next to the current
                       site to $P_f$;
9.                 Select this user moving pattern as
                       a candidate user moving pattern
                       used for prefetch;
10.            end
11.        else
12.                if all moving patterns are scanned,
                       go to line 14.
13.    end
14. return $P_f$
end
```

For illustrative purposes, different scenarios under data allocation schemes DF, DS and DP are shown in Table 4. In order to evaluate the efficiency of data allocation schemes, the local hit ratio can be formulated as $\sum_{j=1}^{k} N_{S_j^i} * P_{S_j^i}$, where S_j^i represents the jth moving hop of user U_i, k is the length of a moving path S_j^i, and $N_{S_j^i}$ is the probability for user U_i to appear at site S_j^i. $P_{S_j^i}$ is equal to 1, if data can be found at site S_j^i and $P_{S_j^i}$ is 0, otherwise. Consider the example scenario in Table 3 and Table 4, where the replicated servers under DS for U_1 are {ABCE} and the example moving path of U_1 is {ABCG}. The length of this moving path is 4. Also, S_1^1 is A at first. Since the data can be found in the set of replicated servers under DS, $P_{s_1^1}$ is set to 1. Due to the movement of U_1, the next site of U_1 is B, which is denoted as S_2^1. Since the data can also be found in the set of replicated servers under DS, $P_{s_2^1}$ is set to 1. It can be verified that the local hit ratio of U_1 under scheme DS can be formulated as $N_A + N_B + N_C$. Also, it can be obtained that the local hit ratio of U_1 under scheme DF is N_A since A is the intersection of {ABCG} and {AFKP}. With the execution scenario shown in Fig. 5, the local hit ratio of U_1 under scheme DP can be expressed as $N_B + N_C$ because B and C are the sites for successful prefetching whereas F is not. As can be seen, the candidate prefetch sites of DP vary when the corresponding user moves. Following the same procedure, the local hit ratios for the moving paths by all other mobile users under schemes DF, DS, and DP can be obtained.

As can be seen from the experimental resutlts in Section 4, scheme DP performs better than scheme DS in that it can achieve the same high local data access hit ratios as DS while incurring a much smaller cost for maintaining replicated sites than DS, showing the very advantage of employing the path level knowledge to do proper prefetches and, thus, suitable for update data objects.

3.3 Scheme for Shared Data Allocation

Shared data refers to those data that are used by many mobile users. Example shared data include public information, cooperative information, etc. By properly determining the set of replicated servers used by a group of mobile users, data allocation for shared data is able to increase the local data access ratio in the sense of both local and global optimization. Local optimization refers to the optimization that the likelihood of local data access by an *individual* mobile user is maximized whereas global optimization refers to the optimization that the likelihood of local data access by *all* mobile users is maximized.

With the user moving patterns obtained, we can develop shared data allocation algorithm (to be referred to as algorithm SD) to determine the set of replicated servers. Algorithm SD is greedy in nature and its performance will be evaluated in Section 4 experimentally. Recall that moving patterns of mobile users may contain different large k-moving sequences, L_k. We first convert these L'_ks into L'_2s and the allocation of shared data will be made in accordance with the occurrences of these L'_2s. An algorithmic form of SD is given below. Consider for example the moving scenario in Table 3, where U_1 is the frequent moving mobile user (with 1,500 movements) and U_2, U_3, and U_4 are nonfrequent moving users (with 350, 300, and 200 movements, respectively). An example profile for the counting in algorithm SD is given in Table 5, where the user occurrence count of L_2 is the number of mobile users whose moving patterns contain that L_2 and the total movement occurrence count of L_2 is the sum of all the movement occurrence counts of that L_2 from all mobile users. For example, since {AB} can only be found in the moving patterns of U_1, the user occurrence count of {AB} is one. Also, since U_1, U_2, and U_3 contain {BC} in their moving patterns, the user occurrence count of {BC} is 3. Let $n_{BC}(U_i)$ denote the occurrence count of {BC} in moving paths of mobile user U_i. The movement occurrence count of {BC} is thus the sum of $n_{BC}(U_1), N_{BC}(U_2)$ and $n_{BC}(U_3)$. This methodology can in fact be used to achieve both the *local optimization* (referred to as SD-local) and the *global optimization* (referred to as SD-global) in that one can use the total *user occurrences* of an L_2 for counting to achieve local optimization whereas using the total *movement occurrences* of that L_2 for counting to achieve global optimization.

TABLE 4
The Scenarios under Different Data Allocation Schemes for Personal Data Allocation

User with an example moving path	Replicated servers under DF	Local data hit of DF
U_1 with ABCG	AFKP	N_A
U_2 with BCGF	AFKP	N_F
U_3 with BCDH	AFKP	0
U_4 with CGH	AFKP	0

(a)

User with an example moving path	Replicated servers under DS	Local data hit of DS
U_1 with ABCG	ABCE	N_A+N_B +N_C
U_2 with BCGF	BCGF	N_B+ N_C+N_G+N_F
U_3 with BCDH	BCD	N_B+N_C+N_D
U_4 with CGH	CGK	N_C+N_G

(b)

User with an example moving path	Replicated servers under DP	Local data hit of DP
U_1 with ABCG	AE, ABC	N_B +N_C
U_2 with BCGF	BCGF	N_C+N_G+N_F
U_3 with BCDH	BCD	N_C+N_D
U_4 with CGH	CGK	N_G

(c)

(a) The scenario under DF. (b) The scenario under DS. (c) The scenario under DP.

```
Algorithm SD:/* Performing both SD-local and SD-global for
        shared data allocation */
Input: All user moving patterns of mobile users.
Output: The set of replicated servers, i.e., R.
begin
1. Determine, from all frequent moving patterns obtained
   by algorithm LM, both user occurrence counts
   (for SD-local) and movement occurrences counts
   (for SD-global) of all frequent $L_2's$
2. Repeat Until $|V| = 0$; /* V is the number of replicated
        servers yet to determine*/
3. begin
4.   Include those $L_2's$ that have maximal support from the
        set of all $L_2's$ into the set c-max. Also, c denotes
        an $L_2$ pair in c-max.
5.   if $|R| = 0$ /* R is the set of replicated servers */
6.     begin
7.       Choose an $L_2$ pair from c-max;
8.       Include this $L_2$ pair into R and exclude this $L_2$ pair
         from the set of all $L_2's$;
9.       $|V| = |V| - 2$;
10.    end
11.  else if ($\exists c \in$ c-max and $R \cap c \neq 0$)
12.    begin
13.      In c-max, choose an $L_2$ pair that has an intersection
         with pairs in R and exclude this $L_2$ pair from
         the set of all $L_2's$;
14.      $|V| = |V| - 1$;
15.    end
16.  else/* In c-max, there is no $L_2$ pair that has an
           intersection with pairs in R */
17.    begin
18.      Choose an $L_2$ pair from c-max and exclude this
           $L_2$ pair from the set of all $L_2's$;
19.      $|V| = |V| - 2$;
20.    end
21.  $R = R \cup c$;
22. end
end
```

Consider the execution of SD-local as an example. Let R denote the set of replicated servers we identify thus far. Once the supports of all L_2 pairs are obtained, we include the L_2 which has maximal user occurrence count (i.e., {BC} according to the profile in Table 5) into the set R in line 4 of algorithm SD. In general, if the number of replicated server, $|R|$, is not equal to the number of replicated servers required, select, from existing L_2 pairs that have maximal supports (i.e., c-max), one that has an intersection with pairs in R (from line 12 to line 15 of algorithm SD). The pair {CG} is hence selected. This step is similar to Prim's algorithm for finding minimal-cost-spanning-tree (MCST) [8]. The difference between SD and MCST is that even if the maximal support of an L_2 pair does not have any intersection with R, this pair can still be included into R as described from line 17 to line 20 of algorithm SD . After the inclusion of {CG}, R becomes {BCG}. Following this procedure, we shall identify and include more proper L_2 pairs until $|R|$ reaches the number of replicated servers required (i.e., $|V| = 0$). After the inclusion of {CG}, there are two possible L_2 pairs (i.e., {GF} and {GK}) to be considered. Thus, one can choose

TABLE 5
An Example Profile for the Counting in Algorithm SD

L_2	User occurrence count	Movement occurrence count.
AB	1	$n_{AB}(U_1)$= 800
BC	3	$n_{BC}(U_1)+n_{BC}(U_2)+n_{BC}(U_3)$= 400+50+150=600
CD	1	$n_{CD}(U_3)$=200
CG	2	$n_{CG}(U_2)+n_{CG}(U_4)$=250+150=400
GK	1	$n_{GK}(U_4)$=100
GF	1	$n_{GF}(U_2)$=100
AE	1	$n_{AE}(U_1)$=500

either K or F to be included into the set of R. In this illustrative example, without loss of generality, K is included in R. Finally, R is composed of the most frequent moving sites for all mobile users in the sense of local optimization. The replicated servers by SD-global can be obtained similarly. Table 6 shows the example execution by algorithm SD with the profile given in Table 5. Note that the local hit of mobile users using SD is higher than that using DF. Also note that the nonfrequent mobile users (such as U_2, U_3, and U_4) will have better local access hits when using SD-local than using SD-global. On the other hand, a frequent user like U_1 performs better under SD-global than under SD-local. These agree with our intuition in that SD-local deals with user occurrence counts and SD-global considers mainly movement occurrence counts, resulting in the situation that SD-local will favor users moving infrequently and SD-global is good for users moving frequently.

4 Performance Study

The effectiveness of using the knowledge on user moving patterns for data allocation is evaluated empirically in this section. The simulation model for the mobile system considered is described in Section 4.1. Experiments results of personal data allocation, including those of DF, DS, and DP, are shown in Section 4.2. Experiments results of algorithm SD for shared data allocation are shown in Section 4.3.

4.1 Simulation Model for a Mobile System

To simulate the servers in a mobile computing system, we use a four by four mesh network [20], [23], where each node represents one server and, hence, there are 16 servers in this model. It is assumed that there are 100,000 mobile users in our simulations. A moving path is a sequence of servers accessed by a mobile user. The size of each moving path is modeled as a uniform distribution between MAXLEN-5 and MAXLEN+5. As explained before, the starting position of a moving path for a mobile user can be either VLR or HLR and is randomly selected between 1 and 16 in each run of simulation. The number of operations submitted by a mobile user to its nearby server is modeled by a uniform distribution between SITEOP-2 and SITEOP+2. After the server has completed these operations, the mobile user moves to one of its neighboring servers depending on a probabilistic model. Explicitly, the probability that a mobile user moves to the server where this user came from is modeled by P_{back} and the probability that the mobile user routes to the other servers is determined by $(1 - P_{back})/(n - 1)$, where n is the number of possible servers this mobile user can

TABLE 6
The Scenarios under Different Data Allocation Schemes for Shared Data

User ID with example moving path	Replicated Server under SD-local	Local hit ratio of SD-local
U_1 with ABCG	BCGK	$N_B+N_C+N_G$
U_2 with BCGF	BCGK	$N_B+N_C+N_G$
U_3 with BCDH	BCGK	N_B+N_C
U_4 with CGH	BCGK	N_C+N_G

(a)

User ID with example moving path	Replicated Server under SD-global	Local hit ratio of SD-global
U_1 with ABCG	ABCE	$N_A+N_B+N_C$
U_2 with BCGF	ABCE	N_B+N_C
U_3 with BCDH	ABCE	N_B+N_C
U_4 with CGH	ABCE	N_C

(b)

(a) The scenario under SD-local. (b) The scenario under SD-global.

TABLE 7
The Parameters Used in the Simulation

Notation	Definition	Value
MAXLEN	length of a moving path	uniform distri. with mean MAXLEN
SITEOP	number of operations performed in a server	unifrom distri. with mean SITEOP
retrospective factor	number of referenced moving paths for incremental mining	various values used
exploring factor	number of moving paths for a mobile user	various values used
P_{back}	backward probability for user movement	various value used
P_{fetch}	pre-fetch size for DP	various value used
P_{fm}	percentage of frequent moving mobile users	various value used

move to. As mentioned before, the number of consecutive moving paths employed for incremental mining of user moving patterns in LM is called the *retrospective factor* (denoted by w). The number of moving paths used to analyze the performance of data allocation schemes is called the *exploring factor*. The prefetch size used by scheme DP is denoted by P_{fetch}. Table 7 summaries the definitions and the values used for some primary simulation parameters. The local hit ratio means the percentage that among all data accesses, data can be obtained from local servers of mobile users, and the support is the threshold used to qualify large moving sequences.

4.2 Experiments for Utilizing Personal Data Allocation

The effectiveness of personal data allocation schemes based on moving patterns will be evaluated in this section. We first examine the performance of schemes DF and DS in Section 4.2.1, and then evaluate the performance of schemes DP in Section 4.2.2.

4.2.1 Experiments of DF and DS for Personal Data Allocation

To conduct the experiments to evaluate DF and DS, we set the value of the retrospective factor to be 10 and the value of the support to be 0.4. Both the local hit ratios and the corresponding server loads of DF and DS are examined with the exploring factor varied. Fig. 6 shows the local hit ratios of DF and DS. It can be seen from Fig. 6 that the local hit ratio of DS is significantly larger than that of DF, showing the very advantage of using the knowledge of user moving patterns in scheme DS. Note that the local hit ratio of DS tends to increase when exploring factor increases. This is due to the fact that with a fix support, more moving patterns are discovered incrementally by DS as the value of exploring factor increases. The server loads of DF and DS with the exploring factor varied are shown in Fig. 7 where the server load corresponds to the number of instructions executed by a server.

From Fig. 7, it can be seen that the load is more balanced under DS than under DF, showing another advantage of properly selecting replicated sites in light of the knowledge on user moving patterns. This agrees with our discussion in Section 3.2.2. The sensitivity of varying the values of the retrospective factor for incremental mining is next investigated. Specifically, we examine the impact of varying the retrospective factor to the performance of DS. Without loss of generality, we set the values of the retrospective factor w to be 10, 20, and 50, and that of the support to be 0.4, and examine the performance of DS as w varies. Fig. 8a shows the cases of using smaller retrospective factors will outperform those using larger retrospective factors. This can be explained by the feature of temporal locality since a larger value of w may involve some obsolete moving patterns and, thus, cannot well predict future moving patterns of mobile users. However, as can be seen from Fig. 8b, performance might degrade when the value of w is too small to be used as a window size for mining (e.g., when $w < 10$), in which cases, there are insufficient moving paths available for determining moving patterns. Clearly, the selection of the value of w will be dependent upon the temporal locality in the workload and can be determined empirically.

As mentioned before, mining results obtained by algorithms MM and LM are used by scheme DS in a set level, which will cause scheme DS to incur more cost to

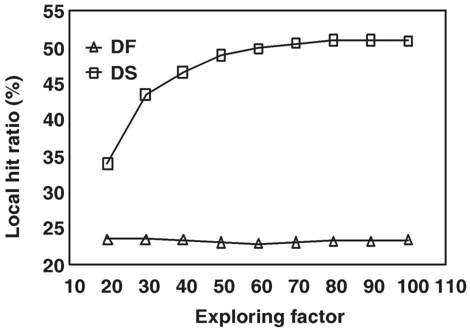


Fig. 6. The local hit ratios of DF and DS with the exploring factor varied.

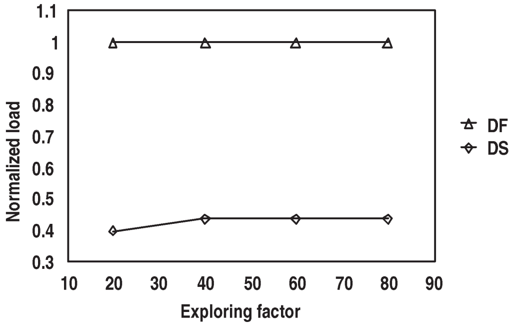
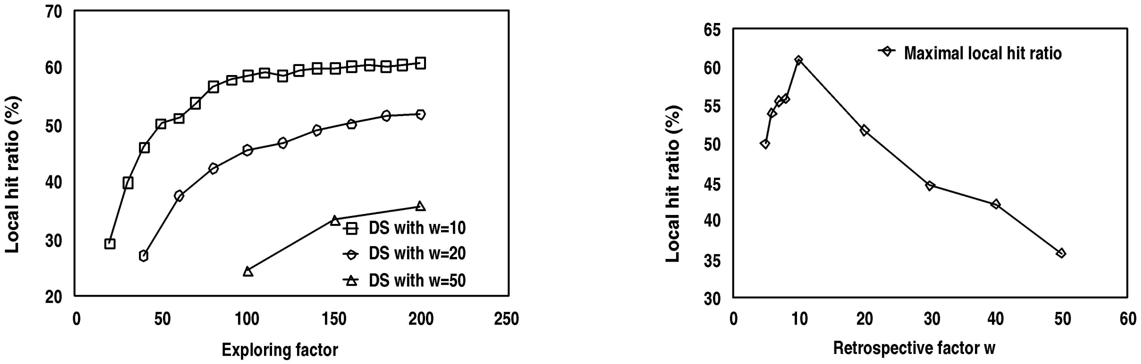


Fig. 7. Server loads of DF and DS with the exploring factor varied (normalized by that of DF).



(a)

(b)

Fig. 8. The performance study on retrospective factor. (a) The performance of DS with the retrospective factor w and the exploring factor varied. (b) The maximal local hit ratio with the retrospective factor varied.

maintain replicated servers. The approach of using the mining results in a path level, i.e., scheme DP, is devised so as to fully utilize the knowledge discovered and reduce the cost incurred by using scheme DS.

4.2.2 Experiments of DP for Personal Data Allocation

As described above, using moving patterns in a set level will unavoidably cause scheme DS to incur more cost to maintain replicated servers, which is not suitable for update data objects. In view of this, scheme DP is designed to exploit the user moving patterns discovered in the path level and the data is prefetched in accordance with the knowledge discovered. As shown in Section 4.2.1, scheme DS in general outperforms scheme DF, we shall thus focus on comparing the performance of scheme DS and that of scheme DP in this section. The effectiveness of DS and DP will be comparatively evaluated.

Without loss of generality, we set the value of P_{fetch} to be 2 in scheme DP and the value of the support to be 0.3. The local hit ratios of DS and DP with various exploring factor are shown in Fig. 9. From Fig. 9, it can be seen that by having the most replicated servers, the local hit ratio of DS is larger than that of DP.

Note, however, that though performing better than DP in local hit ratios, DS incurs more cost to maintain data consistency of replicated servers. A measurement, hit/cost ratio, is used to represent the local hit ratio gained by having an additional replicated server. A larger value of this measurement means that each replicated server is more efficiently used to contribute to the occurrences of local data hits. Fig. 10 shows the hit/cost ratios of DS and DP. Note that DP has a larger hit/cost ratio than DS, meaning that DP employs the approach of data allocation more cost-effectively to increase the local data hit ratio, showing the very advantage of using the path level knowledge to do prefetches.

We next evaluate the effect of varying the values of support and set the value of P_{fetch} to be 2. The local hit ratios of DP with various support values are shown in Fig. 11, where it can be seen that the local hit ratio of DP with a given exploring factor tends to increase as the value of support decreases. Note that more moving patterns have to be explored when the support decreases. While DS suffers more replication overhead as the support decreases, DP avoids this problem since its number of replicated servers is controlled by P_{fetch}. Without increasing the replication cost, decreasing the support can utilize the advantage of DP and improve the corresponding local hit ratios.

From above experimental results, it can be seen that the local hit ratio of DS is significantly larger than that of DF, showing the very advantage of using the knowledge of user moving patterns in scheme DS. However, scheme DS incurs more replicated sites as the number of sites in user moving patterns increases, which results in many sites replicated, thus compromising the overall performance improvement achieved. In order to eliminate the cost of maintaining replicated sites, one should take the properties of data

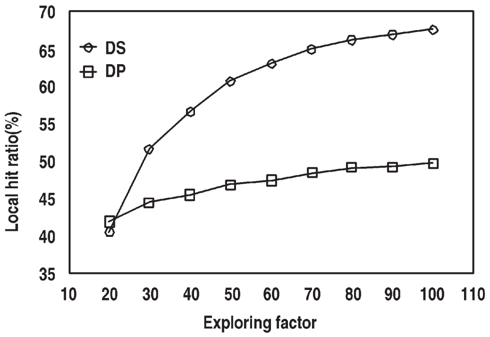


Fig. 9. The local hit ratios of DS and DP with various exploring factor.

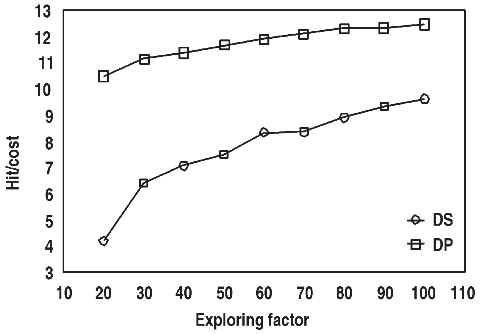


Fig. 10. The local hit ratios of DP with support and the exploring factor varied.

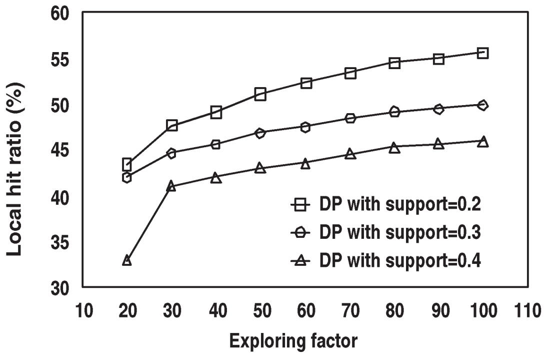


Fig. 11. The local hit ratios of DP with support and the exploring factor varied.

objects into consideration. Clearly, for read data objects, data replication is preferable in order to increase the likelihood of local hits. As a result, scheme DS favors those read-intensive data objects. As pointed out before, it is observed from Fig. 10 that DP has a larger hit/cost ratio than DS, meaning that DP employs the approach of data allocation more cost-effectively to increase the local data hit ratio, showing the benefit of using the path level knowledge to do prefetches. Scheme DP is thus suitable for update-intensive data objects.

4.3 Experiments for Utilizing Shared Data Allocation

We now examine the performance of SD and DF for shared data allocation with the number of mobile users varied. The number of movements by a mobile user is modeled by a uniform distribution between 100 and 1,000. Fig. 12 shows the local hit ratios of DF and SD.

As can be seen from Fig. 12, algorithm SD significantly outperforms DF. In the situation that the difference between the number of movements for frequent users and that for nonfrequent users is not prominent, such as in this experiment, SD-global and SD-local perform similarly. In order to investigate the difference of SD-global and SD-local, we set the number of movements of a frequent moving user to follow a uniform distribution between 900 and 1,000, and the number of movements of a nonfrequent moving users to follow a uniform distribution between 100 and 150. Let P_{fm} denote the percentage of users who move frequently. Fig. 13 shows the local hit ratios of SD-global and SD-local with P_{fm} varied and Fig. 14 shows the total hit counts of SD-global and SD-local with P_{fm} varied. As shown in Fig. 13, when P_{fm} increases, the local hit ratio of SD-global increases and that of SD-local decreases, showing the results from having different replicated servers employed by SD-local and SD-global. This also agrees with our intuition in that as mentioned before, SD-local will favor users moving infrequently and SD-global is good for users moving frequently. It is worth mentioning that although the hit ratio of SD-local is larger than that of SD-global when P_{fm} is less than 0.35, the total hit count of SD-global is larger than that of SD-local, showing the very difference in these two optimizations criteria described in Section 3.3. It is noted that SD-global achieves the global optimization in that the total hit ratio under SD-global is large than that of SD-local despite the local hit ratios of some nonfrequent users under SD-global are smaller than those under SD-local. Clearly, the choice of SD-global and SD-local will be a design issue that is dependent on the system objective.

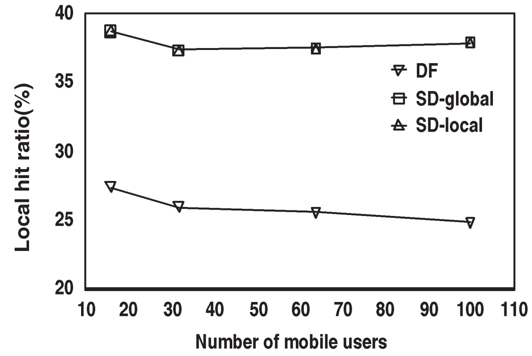


Fig. 12. The local hit ratios of DF, SD-local, and SD-global with the number of mobile users varied.

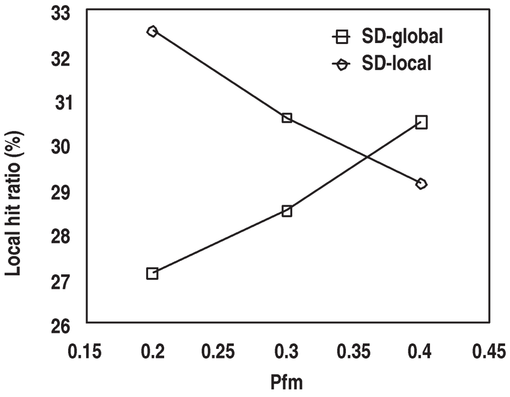


Fig. 13. The local hit ratios of SD-global and SD-local with Pfm varied.

5 Conclusions

In this paper, we presented a new data mining algorithm which involves incremental mining for user moving patterns in a mobile computing environment and utilized the mining results to develop data allocation schemes so as to improve the overall performance of a mobile system. First, we proposed algorithms (i.e., algorithm MM and LM) to capture the frequent user moving patterns from a set of

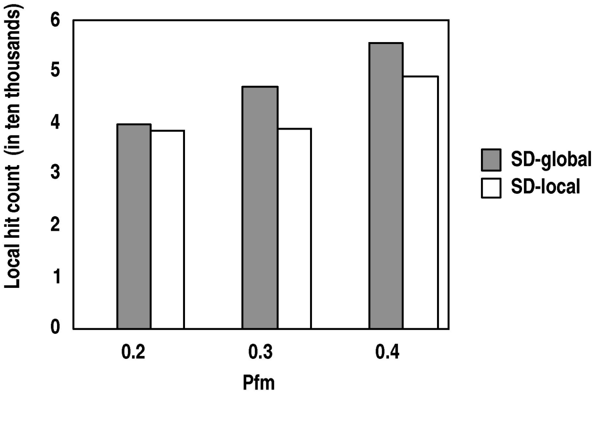


Fig. 14. The total hit count of SD-global and SD-local with Pfm varied.

log data in a mobile environment. The algorithm proposed is enhanced with the incremental mining capability and is able to discover new moving patterns efficiently without compromising the quality of results obtained. In this paper, we devised data allocation schemes that can utilize the knowledge of user moving patterns for proper allocation of both personal and shared data. For personal data allocation, two data allocation schemes, which involve different levels of mining results, have been devised: one utilizes the set level of moving patterns (i.e., scheme DS) and the other utilizes the path level of moving patterns (i.e., scheme DP). As can be seen, the former is useful for the allocation of read data objects, whereas the latter is good for the allocation of update data objects. The data allocation schemes for shared data, which can achieve local optimization (i.e., SD-local) and global optimization (i.e., SD-global), were also developed. Sensitivity analysis on various parameters, including the retrospective factor, was conducted and performance of those data allocation schemes was comparatively analyzed. It was shown by our simulation results that the knowledge obtained from the user moving patterns is very important in devising effective data allocation schemes. Specifically, for the personal data allocation, the local hit ratio of DS is improved due to its knowledge of user moving patterns. However, scheme DS incurs more replicated sites, thus compromising the overall performance improvement achieved. By utilizing the technique of prefetch, DP employs the approach of data allocation more cost-effectively to increase the local data hit ratio. From the performance study of share data allocation, SD-local will favor users moving infrequently and SD-global is good for users moving frequently. It is worth mentioning that the knowledge of user moving patterns is useful on developing efficient location management and querying strategy, and can lead to significant performance improvement in a mobile computing system.

Acknowledgments

The authors would like to thank the anonymous referees for their helpful comments to improve this paper.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Associations between Sets of Items in Massive Databases," *Proc. ACM SIGMOD,* pp. 207-216, May 1993.

[2] R. Agrawal and J. Shafer, "Parallel Mining of Association Rules," *IEEE Trans. Knowledge and Data Eng,* vol. 8, no. 6, pp. 866-883, Dec. 1996.

[3] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int'l Conf. Data Eng.,* pp. 3-14, Mar. 1995.

[4] B. Bruegge and B. Bennington, "Applications of Mobile Computing and Communication," *IEEE Personal Comm.,* pp. 64-71, Feb. 1996.

[5] M.-S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from Database Perspective," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no 6, pp. 866-883, Dec. 1996.

[6] M.-S. Chen, J.-S. Park, and P.S. Yu, "Efficient Data Mining for Path Traversal Patterns," *IEEE Trans. Knowledge and Data Eng.,* vol. 10, no. 2, pp. 209-221, Apr. 1998.

[7] D.W. Cheung, V.T. Ng, W. Fu, and Y. Fu, "Efficient Mining Association Rules in Distributed Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 6, pp. 911-922, Dec. 1996.

[8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithm.* MIT Press, 1989.

[9] N. Davies, G.S. Blair, K. Cheverst, and A. Friday, "Supporting Collaborative Application in a Heterogeneous Mobile Environment," *Computer Comm. Specical Issues on Mobile Computing,* 1996.

[10] M.H. Dunham, A. Helal, and S. Balakrishnan, "A Mobile Transaction Model That Captures Both the Data and Movement Behavior," *ACM J. Mobile Networks and Applications,* vol. 2, pp. 149-162, 1997.

[11] M.H. Dunham and V. Kumar, "Location Dependent Data and its Management in Mobile Databases," *Proc. Ninth Int'l Workshop Database and Expert Systems Applications,* pp. 26-29, Aug. 1998.

[12] EIA/TIA. Cellular Radio Telecommunication Intersystem Operations, 1991.

[13] A. Elmagarmid, J. Jain, and T. Furukawa, "Wireless Client/Server Computing for Personal Information Services and Applications," *ACM SIGMOD RECORD,* vol. 24, no. 4, pp. 16-21, Dec. 1995.

[14] J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database," *Proc. 15th Int'l Conf. Data Eng.,* Mar. 1999.

[15] T. Imielinski and B.R. Badrinath, "Querying in Highly Mobile and Distributed Environment," *Proc. 18th Int'l Conf. Vary Large Data Bases,* pp. 41-52, Aug. 1992.

[16] T. Imielinski and B.R. Badrinath, "Mobile Wireless Computing," *Comm. of ACM* vol. 37, no. 10, pp. 18-28, Oct. 1994.

[17] J. Jannink, D. Lam, N. Shivakumar, J. Widom, and D. Cox, "Efficient amd Flexible Location Management Techniques for Wireless Communication Systems," *ACM J. Wireless Networks,* vol. 3, no. 5, pp. 361-374, 1997.

[18] D.N. Knisely, S. Kumar, S. Laha, and S. Nanda, "Evolution of Wireless Data Services: IS-95 to cdma2000," *IEEE Comm. Magazine,* pp. 140-149, Oct. 1998.

[19] N. Krishnakumar and R. Jain, "Escrow Techniques for Mobile Sales and Inventory Applications," *ACM J. Wireless Network,* vol. 3, no. 3, pp. 235-246, July 1997.

[20] D. Lam, D.C. Cox, and J. Widom, "Teletrafic Modeling for Personal Communication Services," *IEEE Comm.,* vo. 35, no. 2, pp. 79-87, Feb. 1997.

[21] J.-L. Lin and M.H. Dunham, "Mining Association Rules: Anti-Skew Algorithms," *Proc. 14th Int'l Conf. Data Eng.,* pp. 486-493, Feb. 1998.

[22] Y.-B. Lin, "GSM Network Signaling," *ACM Mobile Computing and Comm.,* vol. 1, no. 2, pp. 11-16, 1997.

[23] Y.-B. Lin, "Modeling Techniques for Large-Scale PCS Networks," *IEEE Comm. Magazine,* vol. 35, no. 2, pp. 102-107, Feb. 1997.

[24] Y.-B. Lin, "Reducing Location Update Cost in a PCS Network," *IEEE/ACM Trans. Networking,* vol. 5, no. 1, pp. 25-33, Feb. 1997.

[25] R. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. 18th Int'l Conf. Very Large Data Bases,* pp. 144-155, Sept. 1994.

[26] S.K. Palat and S. Andresen, "Comparison of Replication of the User Mobility Profile with Caching for Reduction of HLR Accesses," *1997 IEEE Int'l Conf. Personal Wireless Comm.,* pp. 173-177, 1997.

[27] J.-S. Park, M.-S. Chen, and P.S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, no. 5, pp. 813-825, Oct. 1997.

[28] W.-C. Peng and M.-S. Chen, "A Dynamic and Adaptive Cache Retrieval Scheme for Mobile Computing Systems," *Proc. Third IFCIS Conf. Cooperative Information Systems (CoopIS '98),* pp. 251-258, Aug. 1998.

[29] M. Satyanarayanan, "Mobile Information Access," *IEEE Personal Comm.,* pp. 26-33, Feb. 1996.

[30] J. Shafer, R. Agrawal, and M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining," *Proc. 22th Int'l Conf. Very Large Databases,* Sept. 1996.

[31] N. Shivakumar and J. Jannink, J. Widom, "Per-User Profile Replication in Mobile Environments: Algorithms, Analysis and Simulation Results," *ACM J. Mobile Networks and Applications,* vol. 2, pp. 129-140, 1997.

[32] N.R. Sollenberger, N. Seshadri, and R. Cox, "The Evolution of IS-136 TDMA for Third-Generation Wireless Services," *IEEE Personal Comm.,* pp. 8-18, June 1999.

[33] WAP Forum. http://www.wapforum.org/, 2002.

[34] O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm," *ACM Trans. Database Systems,* vol. 22, no. 4, pp. 255-314, June 1997.

Wen-Chih Peng received the BS and MS degrees from National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the PhD degree in electrical engineering from the National Taiwan University, Taipei, Republic of China in 2001. During his PhD program, he was mainly involved in the projects related to mobile computing, telelcommunication databases, and data mining. His research interests include mobile computing, mobile data management, and data mining. He is a member of the Phi Tau Phi scholastic honor society.

Ming-Syan Chen received the BS degree in electrical engineering from National Taiwan University, Taipei, and the MS and PhD degrees in computer, information and control engineering from The University of Michigan, Ann Arbor, in 1985 and 1988, respectively. Dr. Chen is currently a professor in the Electrical Engineering Department, National Taiwan University. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 140 papers in his research areas. In addition to serving as a program committee member in many conferences, Dr. Chen is an associate editor of *IEEE Transactions on Knowledge and Data Engineering* on data mining and parallel database areas from 1997 to 2001, an editor of *Journal of Information Science and Engineering*, a distinguished visitor of IEEE Computer Society for Asia-Pacific from 1998 to 2000, and program chair of PAKDD-02 (Pacific Area Knowledge Discovery and Data Mining), program vice-chair of VLDB-2002 (Very Large Data Bases), general chair of Real-Time Multimedia System Workshop in 2001, program chair of IEEE ICDCS Workshop on Knowledge Discovery and Data Mining in the World Wide Web in 2000, and program cochair of the International Conference on Mobile Data Management in 2003, and also of International Computer Symposium on Computer Networks, Internet and Multimedia in 1998 and 2000. He was a keynote speaker on Web data mining at the International Computer Congress in Hong Kong, 1999, a tutorial speaker on Web data mining in DASFAA-1999 and on parallel databases in the 11th IEEE International Conference on Data Engineering in 1995 and a guest coeditor for *IEEE Transactions on Knowledge and Data Engineering* on a special issue for data mining in December 1996. He holds, or has applied for, eighteen U.S. patents and seven ROC patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He received the Outstanding Innovation Award from IBM Corporate in 1994 for his contribution to parallel transaction design and implementation for a major database product, and numerous awards for his research, teaching, inventions, and patent applications. Dr. Chen is a senior member of IEEE and a member of ACM.