Mining Mobile Sequential Patterns in a Mobile Commerce Environment

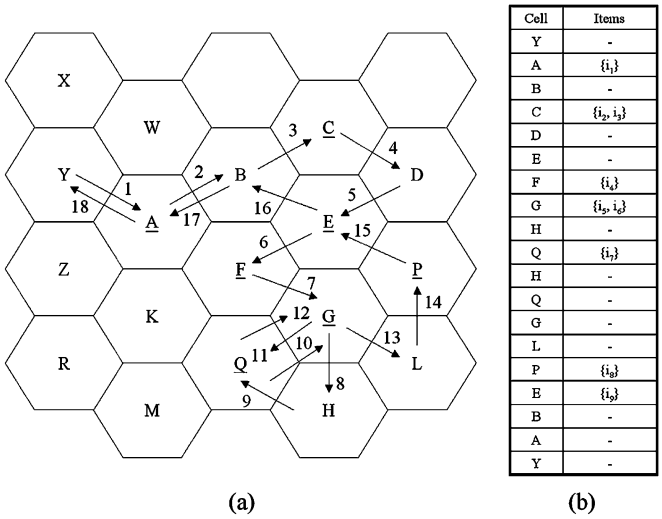
Ching-Huang Yun and Ming-Syan Chen, *Fellow, IEEE*

***Abstract*—In this paper, we explore a new data mining capability for a mobile commerce environment. To better reflect the customer usage patterns in the mobile commerce environment, we propose an innovative mining model, called mining mobile sequential patterns, which takes both the moving patterns and purchase patterns of customers into consideration. How to strike a compromise among the use of various knowledge to solve the mining on mobile sequential patterns is a challenging issue. We devise three algorithms (algorithm $\mathbf{TJ_{LS}}$, algorithm $\mathbf{TJ_{PT}}$, and algorithm $\mathbf{TJ_{PF}}$) for determining the frequent sequential patterns, which are termed large sequential patterns in this paper, from the mobile transaction sequences. Algorithm $\mathbf{TJ_{LS}}$ is devised in light of the concept of association rules and is used as the basic scheme. Algorithm $\mathbf{TJ_{PT}}$ is devised by taking both the concepts of association rules and path traversal patterns into consideration and gains performance improvement by path trimming. Algorithm $\mathbf{TJ_{PF}}$ is devised by utilizing the pattern family technique which is developed to exploit the relationship between moving and purchase behaviors, and thus is able to generate the large sequential patterns very efficiently. A simulation model for the mobile commerce environment is developed, and a synthetic workload is generated for performance studies. In mining mobile sequential patterns, it is shown by our experimental results that algorithm $\mathbf{TJ_{PF}}$ significantly outperforms others in both execution efficiency and memory saving, indicating the usefulness of the pattern family technique devised in this paper. It is shown by our results that by taking both moving and purchase patterns into consideration, one can have a better model for a mobile commerce system and is thus able to exploit the intrinsic relationship between these two important factors for the efficient mining of mobile sequential patterns.**

***Index Terms*—Data mining, mobile computing, mobile sequential patterns, user behavior.**

I. Introduction

THE EMERGENCE of powerful portable devices, along with advance in wireless communication technologies, has made the mobile services available. In the near future, it is expected that tens of millions of users will carry mobile phones or portable devices that use wireless connection to access a worldwide information network for business or personal use from anywhere at any time, making the *mobile commerce (MC)* a reality [1], [57], [58]. For example, eNetwork Web Express [19] enables mobile users to use commercial Web applications over wide-area wireless networks (WANs). Bluetooth technology [20] allows terminals and cash registers to talk directly to each other for the purpose of mobile commerce. The Wireless Access Protocol (WAP) [21] brings the MC environment a world-wide standard for providing Internet communications to digital mobile phones. In an MC environment, customers can make any transaction from anywhere at any time with the payment mechanism provided by banks or credit card companies [58]. In addition, some kind of Nokia mobile phones provide the wallet application that enables customers to get easy access to mobile services and to make convenient online mobile transactions [2]. In the wallet, customers can store sensitive personal information, such as payment and loyalty card details, delivery addresses, and notes, as well as service profiles. In addition, with the wallet application, the Nokia mobile phones have the capability of storing the transactions with moving patterns and purchasing patterns of customers.



Cell	Items
Y	-
A	$\{i_1\}$
B	-
C	$\{i_2, i_3\}$
D	-
E	-
F	$\{i_4\}$
G	$\{i_5, i_6\}$
H	-
Q	$\{i_7\}$
H	-
Q	-
G	-
L	-
P	$\{i_8\}$
E	$\{i_9\}$
B	-
A	-
Y	-

Fig. 1. Illustrative example for a mobile transaction sequence where cells are underlined if items are purchased there.

Example 1.1: One example scenario envisioned for a *mobile transaction sequence* is shown in Fig. 1, where a customer moves in the mobile commerce environment and makes transactions in the corresponding cell through the mobile device. Fig. 1(a) shows the moving patterns of this customer and the mobile transaction sequence data is recorded in Fig. 1(b), where for example, item i_1 was purchased when the customer moved to the cell A.

It is important to note that since customers are moving along an MC environment to search for desired items to purchase, the implications from moving patterns and purchase patterns are in fact entangled, and both are of great importance for studying customer behaviors. Clearly, the distinctive features of knowledge discovery in an MC environment increase the difficulty of

Manuscript received May 2, 2003; revised July 9, 2004. The work was supported in part by the National Science Council of Taiwan, R.O.C., under Contract NSC93-2752-E-002-006-PAE.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. (e-mail: mschen@cc.ee.ntu.edu.tw).

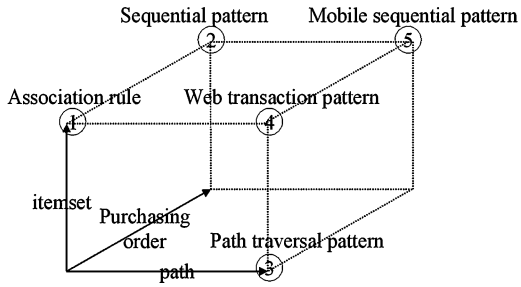
Digital Object Identifier 10.1109/TSMCC.2005.855504

extracting information from the mobile transaction sequences. However, as these mobile commerce services are becoming increasingly popular nowadays, it is imperative to devise efficient algorithms for deriving customer buying behavior to improve the quality of these services. As a result, the design and development of efficient mining algorithms for knowledge discovery in an MC environment while fully exploring the intrinsic relationship between moving and purchase patterns is taken as the objective of this paper. Conducting the mining on the moving and purchase patterns of customers in an MC environment is called the mining of *mobile sequential patterns* (i.e., *large sequential patterns*) in this paper. In addition, a novel knowledge, called *mobile sequential rules*, can be derived from the mobile sequential patterns for the measurement of customer purchase behavior association.

Example 1.2: For the example shown in Fig. 1, the customer has one kind of moving pattern ABC and two kinds of purchasing patterns $\{\langle \mathrm{A}; t_1\rangle$ and $\langle \mathrm{C}; t_9\rangle\}$ where itemset $t_1 = \{i_1\}$ and itemset $t_9 = \{i_2, i_3\}$. If there are sufficient customers having the same patterns, the mobile sequential pattern is an implication of the form $\langle\{\langle \mathrm{A}; t_1\rangle, \langle \mathrm{C}; t_9\rangle\}$: ABC$\rangle$, which means that most customers usually purchase itemset t_1 in cell A and then purchase itemset t_9 in cell C with the specific path ABC. In addition, the mobile sequential rule is an implication of the form $\langle\{\langle \mathrm{A}; t_1\rangle \Longrightarrow \langle \mathrm{C}; t_9\rangle$: ABC$\rangle$ which means that customers purchasing itemset t_1 in cell A are usually moving along path ABC to cell C for purchasing itemset t_9. With the mobile sequential rule, when a customer purchases itemset t_1 in cell A, the cellular phone company could send the coupons of products (i.e., item i_2 and item i_3) in itemset t_9 to boost the sales through the base stations in the cells A, B, or C in accordance with their broadcasting schedules. More description about mobile commerce is available in [1].

The details of related works are given in Section II-A. Despite some efforts having been elaborated upon examining the user behavior, none of the prior work, to the best of our knowledge, has taken both moving and purchase patterns together into consideration to model the customer behavior in a mobile commerce environment. This can in part be explained by the fact that the cost is expensive to track and log detailed movements of mobile users today.[1] However, it is expected that such cost will decrease soon and the cellular phone will become the popular interface of the interconnection networks for accessing various services [57], thus justifying the practicality and necessity of conducting mobile sequential pattern mining. It is understood that the records of cells visited and items purchased, required for mining mobile sequential patterns, may belong to different companies, and for these companies, they may have different considerations on using their data to improve the mobile commerce services provided. It should go without saying that such data analysis should be done solely for the purposes of system and service improvements and should be conducted in a contingent way that neither any law is violated nor is the privacy of customers intruded. Nevertheless, with the legality and privacy issues considered, the knowledge discovery from the MC data is believed to be an increasingly challenging technical problem which is of great practical importance for the evolving MC techniques.

[1]The cost to locate a mobile user is estimated to be about US $0.01 to US $0.05 each time according to a major mobile phone service provider.



No.	Mining capability	Quantities used	Applications
1	Association rule	itemset	Basket market
2	Sequential pattern	itemset, purchasing order	Basket market
3	Path traversal pattern	path	Web traverse
4	Web transaction pattern	itemset, path	E-Commerce
5	Mobile sequential pattern	itemset, purchasing order, path	M-Commerce

Fig. 2. Notion of mining mobile sequential patterns.

Consequently, to better reflect the customer buying behavior in the MC environment, we propose an innovative mining model that takes both the moving patterns and purchase patterns into consideration. In essence, the mining of mobile sequential patterns aggregates the concepts on mining association rules, mining path traversal patterns, and mining sequential patterns, and thus requires a combined use of corresponding techniques. The notion of mining mobile sequential patterns is shown in Fig. 2, where the relationship among these mining capabilities is depicted. How to strike a compromise among the use of various knowledge to solve the mining on mobile sequential patterns is a challenging issue. As an effort to solve this problem, we devise a procedure, namely *mobile sequential patterns MSPs*), to conduct the mining of mobile sequential patterns. With the details described in the Section II-C, the procedure MSP splits the problem of mining mobile sequential patterns into four phases, namely: 1) the *large-transaction generation phase*; 2) the *large-transaction transformation phase*; 3) the *sequential-pattern generation phase*; and 4) the *sequential-rule generation phase*.

In this paper, the performance bottleneck is in phase 3), i.e., the sequential-pattern generation phase. By having different priorities on the factors involving large itemsets, traversal paths and orders of purchases, we devise three algorithms (algorithm $\mathrm{TJ_{LS}}$, algorithm $\mathrm{TJ_{PT}}$, and algorithm $\mathrm{TJ_{PF}}$) to determine mobile sequential patterns. First, algorithm $\mathrm{TJ_{LS}}$ is devised in light of the concept of itemset joining in association rules mining [6]. However, as will be seen later, without fully utilizing the traversal paths of mobile sequential patterns, algorithm $\mathrm{TJ_{LS}}$ tends to count the supports of a lot of *out-of-path sequential patterns* (i.e., the sequential patterns which do not stay within a path), thus degrading the performance. Next, to eliminate the out-of-path sequential patterns, algorithm $\mathrm{TJ_{PT}}$ is devised by taking both the concepts of association

rules [6] and path traversal patterns [12] into consideration and gains performance improvement by path trimming. However, algorithm $\mathrm{TJ_{PT}}$ incurs the comparison along the path with time complexity $O(P)$, where P is the average path length, and may generate some *uncertain candidate sequential patterns* which undesirably require further subpattern identifications.

Consequently, by fully exploring the intrinsic relationship between moving and purchase behaviors of customers, algorithm $\mathrm{TJ_{PF}}$ is developed in light of the *pattern family* technique. The pattern family of a pattern consists of the pattern itself and all its subpatterns generated in each round. In essence, the pattern family technique is a filtering technique that fully exploits the purchase patterns generated in the intermediate stages of mobile sequential pattern mining. For better readability, we defer the detailed description of the pattern family technique and the corresponding theoretical properties to Section III. It will be shown that utilizing the information of pattern family, algorithm $\mathrm{TJ_{PF}}$ compares the path with time complexity $O(1)$ and generates fewer uncertain candidate sequential patterns, thus reducing the corresponding computational overhead and memory consumption.

After all mobile sequential patterns are obtained, the mobile sequential rules can be derived with a straightforward way and it is described clearly in Section II-C4. A simulation model for the MC environment is developed and a synthetic workload is generated for performance studies. By utilizing pattern family technique, $\mathrm{TJ_{PF}}$ is shown to be able to determine large sequential patterns very efficiently. As validated by the synthetic workload, it is shown by our experimental results that algorithm $\mathrm{TJ_{PF}}$ significantly outperforms others in both the execution efficiency and the memory saving. It is shown by our results that by taking both moving patterns and purchase patterns into consideration, one can have a better model for an MC system and is thus able to exploit the intrinsic relationship between these two customer behaviors.

This paper is organized as follows. Preliminaries are given in Section II. In Section III, three algorithms ($\mathrm{TJ_{LS}}$, $\mathrm{TJ_{PT}}$, and $\mathrm{TJ_{PF}}$) are devised for determining large sequential patterns. Experimental studies are conducted in Section IV. This paper concludes with Section V.

II. Preliminaries

In this section, the problem of mining mobile sequential patterns is described in Section II-A, the related works are described in Section II-B, and the procedure of mining mobile sequential rules is outlined in Section II-C.

A. Problem Formulation

In the mobile commerce environment where items are sold in various cells, customers may move among the cells to purchase items of interest with either traditional or electronic commerce trading mechanisms. In either case, customers pay for items through the mobile devices and the purchasing records are logged. Let $N = \{n_1, n_2, \ldots, n_g\}$ be a set of cells in the MC environment and $I = \{i_1, i_2, \ldots, i_h\}$ be a set of items sold in that environment. We are given a database of mobile transaction sequences, where each mobile transaction sequence consists of sequence-id, cells visited, and a list of itemsets purchased in the corresponding cells, ordered by customer movements among cells.

TABLE I
Patterns and Their Notations

Term	Notation
itemset	$\{i_1, i_2, ..., i_p\}$
transaction	$<C; \{i_1, i_2, ..., i_p\}>$
litemset	The itemset is large
L-transaction	The transaction is large
path	$n_1 n_2 ... n_y$
large k-sequential pattern	$<\{x_1, x_2, ..., x_k\}:\ n_1 n_2 ... n_y>$
mobile sequential rule	$<\mathrm{X} \Longrightarrow \mathrm{Z} : n_1 n_2 ... n_y>$
maximal sequential pattern	$<\{x_1, x_2, ..., x_k\}:\ n_1 n_2 ... n_y>$
maximal-path large 2-sequential pattern ($\mathrm{MS_2}$)	$<\{x_1, x_k\}: m_1 m_2 ... m_q>$
centro-subtransactionset	$\{x_2, ..., x_{k-1}\}$

A path is denoted by $\langle n_1 n_2 \ldots n_y \rangle$, where $n_j \in N$, for $1 \leq j \leq y$. Thus, the sequence of cells visited implicitly forms the path of the mobile transaction sequence. In this paper, the discovered patterns and their notation are given in Table I. A *transaction*, denoted as $\langle C; \{i_1, i_2, \ldots, i_p\} \rangle$, means that itemset $\{i_1, i_2, \ldots, i_p\}$ was bought in cell C, where $C \in N$, and $\{i_1, i_2, \ldots, i_p\} \subseteq I$. Thus, the list of itemsets purchased in the corresponding cells implicitly forms the list of transactions of the mobile transaction sequence. As a result, each mobile transaction sequence contains the information of path and a list of transactions. Given a database D_M of mobile transaction sequences, the problem of mining mobile sequential patterns is to discover the frequent sequential patternss among all mobile transaction sequences. A sequential pattern is represented by the form $\langle$*list of transactions: path*$\rangle$, where the transactions are made along the path. The support for a sequential pattern is defined as the number of mobile transaction sequences which support this sequential pattern. A *large sequential pattern* is a sequential pattern with the minimum support (i.e., a sequential pattern that appeared in a sufficient number of mobile transaction sequences).

The length of a large sequential pattern is the number of transactions in that large sequential pattern. A large sequential pattern of length k is called a *large k-sequential pattern*. Thus, a large 1-sequential pattern can be represented by the form $\langle$transaction: cell$\rangle$, where the transaction is made in the cell. Note that each transaction in a large k-sequential pattern must meet the minimum support. In [7], an itemset with minimum support is called a *large itemset* or *litemset*. Similarly, we call a transaction with minimum support *large transaction* or *L-transaction*, which can be represented as $\langle C; t_j \rangle$, where t_j represents a litemset in cell C. Thus, if the transaction $\langle C; \{i_1, i_2, \ldots, i_p\} \rangle$ has the minimum support, the litemset $\{i_1, i_2, \ldots, i_p\}$ will be represented as t_j and the L-transaction $\langle C; \{i_1, i_2, \ldots, i_p\} \rangle$ will be represented as $\langle C; t_j \rangle$. Since each transaction in a large k -sequential pattern will have the minimum support, a large k-sequential pattern can be represented as $\langle \{x_1, x_2, \ldots, x_k\}: n_1 n_2 \ldots n_y \rangle$, where x_j is an L-transaction made along the path $n_1 n_2 \ldots n_y$.

Recall that in association rules [6], a large itemset is a frequently purchased itemset. In sequential patterns [7], a large

sequence is a frequently purchased set of itemsets ordered by the purchase time. In traversal patterns [12], a large reference is a frequently traveled path. In this paper, a large sequential pattern is a pattern containing: 1) the frequently purchased itemset (meaning that the itemset in an L-transaction must have the minimum support); 2) the frequently purchased set of itemsets ordered by the purchase time (meaning that the set of L-transactions in a large sequential pattern must have the minimum support); and 3) the frequently traveled path (meaning that the path in a large sequential pattern must have the minimum support).

B. Related Work

Recently, mining of databases has attracted a growing amount of attention in database communities due to its wide applicability to studying the buying behaviors of customers [11], [18]. Mining association rules is employed to discover the important associations among items such that the presence of some items in a transaction will imply the presence of other items in the same transaction [5]. After that, several technologies on association rule mining have been developed including: 1) algorithm improvements [3], [6], [10], [15], [27], [32], [43], [68]; 2) constraint-based [25], [28], [46]; 3) incremental updating [9], [14], [30]; 4) multiple minimum supports [35], [60]; 5) frequent closed itemsets [45], [47], [67]; and 6) generalized [53], multilevel [23], intertransaction [56], quantitative [54], and multidimensional [61], [62].

Mining sequential patterns was first introduced in [7] for finding the intertransaction patterns in the traditional retailing environments. After that, several technologies on sequential pattern mining have been developed, including: 1) algorithm improvements [26], [39], [48], [66]; 2) constraint-based [36], [55], [65]; 3) incremental updating [33], [44], [69]; and 4) generalized [55].

Several temporal association rule mining techniques are addressed in [8], [13], [29], and [41]. Episode mining has been studied in [31] and [38] for discovering frequent patterns in a sequence of time events. Das *et al*. [17] investigated the problem of finding rules relating patterns in a time series to other patterns in that series, or patterns in one series to patterns in another series. Mining series of interval events was discussed in [59] for discovering the temporal containment relationships of event sequences. A temporal logic approach is proposed in [42] for finding temporal patterns. Mining partial orders from the sequential data is explored in [37]. Mining segmentwise periodic patterns is discussed in [24]. Mining asynchronous periodic patterns is investigated within a subsequence shifted by disturbance [63]. Searching for partial periodic patterns in time-series databases is discussed in [22].

A study on efficient mining of path traversal patterns for capturing Web user behavior was conducted in [12]. WEBMINER [16] was designed for mining Web usage association rules and sequential patterns. Several WWW server logs are analyzed in [50] for deriving the path distribution patterns of the Web users. With mining longest repeated subsequences, a robust method was proposed in [51] for reducing the complexity

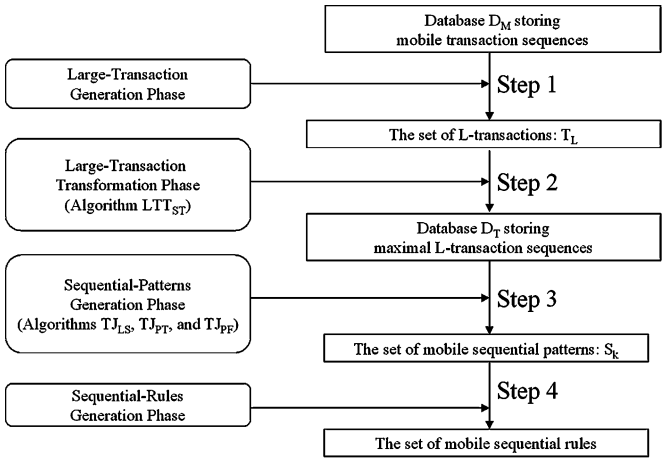


Fig. 3. Flowchart of the whole procedure of mining mobile sequential patterns.

while preserving the predictability of the Web user surfing paths. Recently, several EC Web sites have been using recommended systems for analyzing the customer behaviors to help their customers to find products for possible purchases [4], [52]. For capturing the user behavior in the EC environment, a study on efficient mining Web transaction patterns was reported in [64].

Note that by treating the cells as other items in the patterns, one may extend algorithm GSP, which was designed for mining conventional sequential patterns in [55], to find mobile sequential patterns. However, such an extension to algorithm GSP is not deemed ideal for mining mobile sequential patterns for two reasons. First, since our approaches process the cells and the items individually, and the modified GSP treats the cell as another item in the patterns, it is expected that the former will have a smaller domain to process, thereby having better efficiency, than the latter. More importantly, by simply treating the cell as another attribute, GSP is not able to utilize the intrinsic relationship between moving and purchase behaviors of customers, thus not attaining the mining efficiency we could have owing to the nature of this problem.

C. Procedure for Mining Mobile Sequential Patterns

With the aggregate concept of mining on association rules, path traversal patterns and sequential patterns, the problem of mining mobile sequential patterns cannot be solved by a simple addition of prior techniques since factors in these companion mining capabilities are in fact entangled. This fact justifies the necessity of devising a new mining procedure for mobile sequential patterns. As the mobile commerce business has been identified by several leading industrial companies as the key direction to move for years to come, it is believed that mining mobile sequential patterns has become a very timely and important issue to address.

The flowchart for the whole procedure is shown in Fig. 3 and the meanings of symbols are given in Fig. 4. In the overall procedure, the proposed methods for mining mobile sequential patterns is outlined as follows.

Notation	Meaning
D_M	Database storing mobile transaction sequences
<C; $\{i_k\}$>	Item i_k purchased in cell C
T_L	L-transaction set: the set of large transactions
<C; t_k>	L-transaction <C; t_k> : litemset t_k in cell C
LTT_{ST}	(Algorithm) Large-Transaction Transformation with Sequence-Trimming
D_T	Database storing maximal L-transaction sequences
TJ_{LS}	(Algorithm) Transactionset Join with L-transaction Set
TJ_{PT}	(Algorithm) Transactionset Join with Path Trimming
TJ_{PF}	(Algorithm) Transactionset Join with Pattern Family
S_k	The set of large k-sequential patterns
R_k	The set of candidate k-L-transaction set
C_k	The set of candidate k-sequential patterns
C'_k	The set of uncertain candidate k-sequential patterns

Fig. 4. Meanings of symbol used in mining mobile sequential patterns.

100	
Cell	Items
Y	-
A	$\{i_1\}$
B	-
C	$\{i_2, i_3\}$
D	-
E	-
F	$\{i_4\}$
G	$\{i_5, i_6\}$
H	-
Q	$\{i_7\}$
H	-
Q	-
G	-
L	-
P	$\{i_8\}$
E	$\{i_9\}$
B	-
A	-
Y	-

200	
Cell	Items
X	-
W	-
A	$\{i_1\}$
B	-
C	$\{i_3\}$
D	-
E	-
F	$\{i_4\}$
G	$\{i_5\}$
H	-
Q	$\{i_7\}$
M	-

300	
Cell	Items
A	$\{i_1\}$
W	-
B	-
C	$\{i_3\}$
E	-
F	$\{i_4\}$
G	$\{i_5\}$

400	
Cell	Items
W	-
A	$\{i_1\}$
B	-
C	$\{i_3\}$
D	-
E	-
F	$\{i_4\}$
G	-
Q	-
H	-
G	-
H	$\{i_{10}\}$
G	-
L	-
P	$\{i_8\}$

500	
Cell	Items
A	$\{i_1\}$
B	-
C	$\{i_3\}$
D	-
E	-
F	$\{i_4\}$
G	-
H	$\{i_{10}\}$

600	
Cell	Items
X	-
A	$\{i_1\}$
B	-
C	$\{i_2, i_3\}$
B	-
A	-

Fig. 5. Illustrative example database D_M that stores six mobile transaction sequences.

Procedure MSP (Mobile Sequential Patterns):

1) **Large-Transaction Generation Phase:** Determine the (*L-transactions large transactions*) from the mobile transaction sequences.
2) **Large-Transaction Transformation Phase:** Employ algorithm *Large-Transaction Transformation with Sequence-Trimming* ($\mathrm{LTT_{ST}}$) to transform all mobile transaction sequences into the *maximal L-transaction sequences*.
3) **Sequential-Pattern Generation Phase:** Employ one of the following three algorithms [$\mathrm{TJ_{LS}}$ (Transactionset Join with Large-transaction set), $\mathrm{TJ_{PT}}$ *(Transactionset Join with Path Trimming)*, and $\mathrm{TJ_{PF}}$ *(Transactionset Join with Pattern Family)*] to determine the *large sequential patterns* from the maximal L-transaction sequences.
4) **Sequential-Rule Generation Phase:** Derive *mobile sequential rules* from the large sequential patterns.

1) Large-Transaction Generation Phase: For each cell, we apply a modified algorithm DHP [43] for finding the set of all L-transactions T_L. Similarly to the approach taken by [7], the set of litemsets is mapped to a set of contiguous integers for reducing the time required to check if a mobile sequential pattern is contained in a mobile transaction sequence. Note that we are able to simultaneously discover the set of all large 1-sequential patterns, since this set is mainly $\{\langle x\colon C\rangle | x \in T_L, C$ is the cell containing itemset $x\}$.

Example 2.1: An illustrative database for this problem is shown in Fig. 5, where Sequence IDentification (SID) 100 is the mobile transaction sequence shown in Fig. 1. For the example database in Fig. 5, the L-transactions are shown in Fig. 6(a). For the L-transactions shown in Fig. 6(a), after the mapping shown in Fig. 6(b), the set of large 1-sequential patterns is shown in Fig. 6(c).

2) Large-Transaction Transformation (LTT) Phase: As will be seen in Section III, we need to repeatedly determine which part of a given set of large sequential patterns will appear in the mobile sequential patterns. For efficiently mining the patterns, we employ algorithm $\mathrm{LTT_{ST}}$ to transform each mobile transaction sequence into a maximal L-transaction sequence in this phase.

Large 1-Transactions

Cell	Itemset	Sup.
A	$\{i_1\}$	6
C	$\{i_2\}$	2
C	$\{i_3\}$	6
F	$\{i_4\}$	5
G	$\{i_5\}$	3
Q	$\{i_7\}$	2
P	$\{i_8\}$	2
H	$\{i_{10}\}$	2

Large 2-Transactions

Cell	Itemset	Sup.
C	$\{i_2, i_3\}$	2

(a)

Cell	Itemset	L-Transaction
A	$\{i_1\}$	<A; t_1>
C	$\{i_2\}$	<C; t_2>
C	$\{i_3\}$	<C; t_3>
F	$\{i_4\}$	<F; t_4>
G	$\{i_5\}$	<G; t_5>
Q	$\{i_7\}$	<Q; t_6>
P	$\{i_8\}$	<P; t_7>
H	$\{i_{10}\}$	<H; t_8>
C	$\{i_2, i_3\}$	<C; t_9>

(b)

L-Transaction	Path	Sup.
<A; t_1>	A	6
<C; t_2>	C	2
<C; t_3>	C	6
<F; t_4>	F	5
<G; t_5>	G	3
<Q; t_6>	Q	2
<P; t_7>	P	2
<H; t_8>	H	2
<C; t_9>	C	2

(c)

Fig. 6. Mapping table shown in (b) maps the large transactions in (a) to the large 1-sequential patterns in (c).

Example 2.2: With the mobile transaction sequence shown in Fig. 1 and the mapping table shown in Fig. 6(b), Fig. 7 illustrates the operations in algorithm $\mathrm{LTT_{ST}}$. In Fig. 7, the first column corresponds to the sequence of movements, the second column contains the nodes visited and the third column has the items purchased in SID 100. The fourth column gives the on-going L-transaction in the buffer and the fifth column gives the on-going string in the buffer. The sixth column shows the L-transaction set and the seventh column shows the path of the maximal L-transaction sequence generated by $\mathrm{LTT_{ST}}$.

Note that the same itemsets in different cells are viewed as different transactions. Thus, the same litemsets sold in different cells will be transformed to different integers.

Example 2.3: For example, transactions $\langle A; \{i_g, i_h\}\rangle$, $\langle B; \{i_g, i_h\}\rangle$, and $\langle C; \{\mathrm{i}_g, \mathrm{i}_h\}\rangle$ all have itemset $\{\mathrm{i}_g, \mathrm{i}_h\}$. In addition, $\{\mathrm{i}_g, \mathrm{i}_h\}$ is a litemset in both cells A and B but is not in cell C. After this phase, $\{\mathrm{i}_g, \mathrm{i}_h\}$ in cell A and $\{\mathrm{i}_g, \mathrm{i}_h\}$ in cell B are transformed to different integers (say, t_y and t_z) whereas $\{\mathrm{i}_g, \mathrm{i}_h\}$ in cell C will be trimmed. For the database D_M shown

	SID=100		Buffer		Maximal L-Transaction Sequence	
Move	Cell	Items	L-Transaction	String	L-Transaction set	Path
1	Y	-	-	-	-	-
2	A	$\{i_1\}$	<A; t_1>	A	<A; t_1>	A
3	B	-	<A; t_1>	AB	<A; t_1>	A
4	C	$\{i_2, i_3\}$	<C; t_2, t_3, t_9>	C	<A; t_1>, <C; t_2, t_3, t_9>	ABC
5	D	-	<C; t_2, t_3, t_9>	CD	<A; t_1>, <C; t_2, t_3, t_9>	ABC
6	E	-	<C; t_2, t_3, t_9>	CDE	<A; t_1>, <C; t_2, t_3, t_9>	ABC
7	F	$\{i_4\}$	<F; t_4>	F	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>	ABCDEF
8	G	$\{i_5, i_6\}$	<G; t_5>	G	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>	ABCDEFG
9	H	-	<G; t_5>	GH	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>	ABCDEFG
10	Q	$\{i_7\}$	<Q; t_6>	Q	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ
11	G	-	<Q; t_6>	QG	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ
12	Q	-	<Q; t_6>	Q	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ
13	G	-	<Q; t_6>	QG	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ
14	L	-	<Q; t_6>	QGL	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ
15	P	$\{i_8\}$	<P; t_7>	P	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>, <P; t_7>	ABCDEFGHQGLP
16	E	$\{i_9\}$	<P; t_7>	PE	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>, <P; t_7>	ABCDEFGHQGLP
17	B	-	<P; t_7>	PEB	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>, <P; t_7>	ABCDEFGHQGLP
18	A	-	<P; t_7>	PEBA	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>, <P; t_7>	ABCDEFGHQGLP
19	Y	end	-	-	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>, <P; t_7>	ABCDEFGHQGLP

Fig. 7. Example for producing the maximal large transaction sequences.

in Fig. 5, the transformed database D_T, storing maximal L-transaction sequences, is shown in the first table of Fig. 8 for illustrative purposes.

3) Sequential-Pattern Generation Phase: After all the mobile transaction sequences are transformed to maximal L-transaction sequences, three algorithms (algorithm $\mathrm{TJ_{LS}}$, algorithm $\mathrm{TJ_{PT}}$, and algorithm $\mathrm{TJ_{PF}}$) are devised for mining large sequential patterns from the transformed database D_T. A large k-sequential pattern is represented as $\langle\{x_1, x_2, \ldots, x_k\}$: $n_1 n_2 \ldots n_y\rangle$, where x_j is an L-transaction made along the path $\{n_1 n_2 \ldots n_y\}$. The details of algorithms of this phase will be described in Section III.

Example 2.4: The large sequential patterns, generated in the sequential-pattern generation phase from the example database D_T, are shown in Fig. 8. For example, $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle\}$: ABCDEF$\rangle$ is one large 3-sequential pattern, whose L-transaction set and path appear in SID 100, SID 200, SID 400, and SID 500. The support is thus 4.

4) Sequential-Rule Generation Phase: After the sequential-pattern generation phase, we can find the mobile sequential rules from the large sequential patterns in this phase in a straightforward manner. Unlike the association rule [6], the mobile sequential rule, derived from mobile sequential patterns in this paper, is an implication of the form $\langle \mathrm{X} \Longrightarrow \mathrm{Z}$:$n_1, n_2, \ldots, n_y\rangle$, where X and Z are both sets of L-transactions, $\mathrm{X} \cap \mathrm{Z} = \Phi$, and $\{n_1, n_2, \ldots, n_y\} \subseteq \mathrm{N}$. The rule $\langle \mathrm{X} \Longrightarrow \mathrm{Z}$: $n_1 n_2 \ldots n_y\rangle$ has *support* s if the number of mobile transaction sequences in D_M containing $\langle \mathrm{X} \cup \mathrm{Z}$: $n_1 n_2 \ldots n_y\rangle$ is s. Also, the rule $\langle \mathrm{X} \Longrightarrow \mathrm{Z}$: $n_1, n_2, \ldots, n_y\rangle$ holds with *confidence* c if $c\%$ of mobile transaction sequences in D_M that contain X also contain Z along the path $\{n_1, n_2, \ldots, n_y\}$. Explicitly, support $(\langle \mathrm{X} \Longrightarrow \mathrm{Z}$: $n_1 n_2 \ldots n_y\rangle) = \mathrm{support}(\langle \mathrm{X} \cup \mathrm{Z}$: $n_1 n_2 \ldots n_y\rangle)$, and confidence $(\langle \mathrm{X} \Longrightarrow \mathrm{Z}$: $n_1, n_2, \ldots, n_y\rangle) = (\langle \mathrm{X} \cup \mathrm{Z}$: $n_1 n_2 \ldots n_h \ldots n_y\rangle)/(\langle \mathrm{X}$:$n_1 n_2 \ldots n_h\rangle\})$.

Example 2.5: For example, suppose that $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\langle, \langle \mathrm{F}; \mathrm{t}_4\rangle\}$: ABCDEF$\rangle$ is one large 3-sequential pattern with $\mathrm{support} = 4$ and $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle\}$: ABC$\rangle$ is one large 2-sequential pattern with $\mathrm{support} = 5$. Then, we can derive one mobile sequential rule $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle\} \Longrightarrow \{\langle \mathrm{F}; \mathrm{t}_4\rangle\}$: ABCDEF$\rangle$ with the support equal to $\mathrm{support}(\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle\} \Longrightarrow \{\langle \mathrm{F}; \mathrm{t}_4\rangle\}$: ABCDEF$\rangle) = \mathrm{support}(\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle\}$: ABCDEF$\rangle) = 4$ and the confidence $(\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle\} \Longrightarrow \{\langle \mathrm{F}; \mathrm{t}_4\rangle\}$: ABCDEF$\rangle) = (\mathrm{support}(\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle\}$:ABCDEF$\rangle))/(\mathrm{support}(\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle\}$: ABC$\rangle)\} = 80\%$.

III. Algorithms for Mining Mobile Sequential Patterns

Once the database contains maximal L-transaction sequences for all mobile users, we can derive the large sequential patterns by identifying the frequently occurring transaction sequences. Let S_k be the set of large k-sequential patterns, R_k be the set of candidate k-L-transaction sets, and C_k represent the set of candidate k-sequential patterns. R_k is the transaction component of C_k, and S_k is a subset of C_k. By having different priorities on the factors involving large itemsets, traversal paths and orders of purchases, we devise three algorithms (algorithm $\mathrm{TJ_{LS}}$, algorithm $\mathrm{TJ_{PT}}$, and algorithm $\mathrm{TJ_{PF}}$) to determine large sequential

Database D_T (minimum support = 2)

SID	L-Transaction Set	Path
100	<A; t_1>, <C; t_2, t_3, t_9>, <F; t_4>, <G; t_5>, <Q; t_6>, <P; t_7>	ABCDEFGHQGLP
200	<A; t_1>, <C; t_3>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ
300	<A; t_1>, <C; t_3>, <F; t_4>, <G; t_5>	AWBCEFG
400	<A; t_1>, <C; t_3>, <F; t_4>, <H; t_8>, <P; t_7>	ABCDEFGHGLP
500	<A; t_1>, <C; t_3>, <F; t_4>, <H; t_8>	ABCDEFGH
600	<A; t_1>, <C; t_2, t_3, t_9>	ABC

Large 1-Sequential Patterns (S_1)

L-Transaction	Path	Sup.
<A; t_1>	A	6
<C; t_2>	C	2
<C; t_3>	C	6
<F; t_4>	F	5
<G; t_5>	G	3
<Q; t_6>	Q	2
<P; t_7>	P	2
<H; t_8>	H	2
<C; t_9>	C	2

Large 2-sequential patterns (S_2)

L-Transaction Set	Path	Sup.
<A; t_1>, <C; t_2>	ABC	2
<A; t_1>, <C; t_3>	ABC	5
<A; t_1>, <F; t_4>	ABCDEF	4
<A; t_1>, <G; t_5>	ABCDEFG	2
<A; t_1>, <Q; t_6>	ABCDEFGHQ	2
<A; t_1>, <P; t_7>	ABCDEFGLP	2
<A; t_1>, <H; t_8>	ABCDEFGH	2
<A; t_1>, <C; t_9>	ABC	2
<C; t_3>, <F; t_4>	CDEF	5
<C; t_3>, <G; t_5>	CDEFG	2
<C; t_3>, <Q; t_6>	CDEFGHQ	2
<C; t_3>, <P; t_7>	CDEFGLP	2
<C; t_3>, <H; t_8>	CDEFGH	2
<F; t_4>, <G; t_5>	FG	3
<F; t_4>, <Q; t_6>	FGHQ	2
<F; t_4>, <P; t_7>	FGLP	2
<F; t_4>, <H; t_8>	FGH	2
<G; t_5>, <Q; t_6>	GHQ	2

Large 3-sequential patterns (S_3)

L-Transaction Set	Path	Sup.
<A; t_1>, <C; t_3>, <F; t_4>	ABCDEF	4
<A; t_1>, <C; t_3>, <G; t_5>	ABCDEFG	2
<A; t_1>, <C; t_3>, <Q; t_6>	ABCDEFGHQ	2
<A; t_1>, <C; t_3>, <P; t_7>	ABCDEFGLP	2
<A; t_1>, <C; t_3>, <H; t_8>	ABCDEFGH	2
<A; t_1>, <F; t_4>, <G; t_5>	ABCDEFG	2
<A; t_1>, <F; t_4>, <Q; t_6>	ABCDEFGHQ	2
<A; t_1>, <F; t_4>, <P; t_7>	ABCDEFGLP	2
<A; t_1>, <F; t_4>, <H; t_8>	ABCDEFGH	2
<A; t_1>, <G; t_5>, <Q; t_6>	ABCDEFGHQ	2
<C; t_3>, <F; t_4>, <G; t_5>	CDEFG	2
<C; t_3>, <F; t_4>, <Q; t_6>	CDEFGHQ	2
<C; t_3>, <F; t_4>, <P; t_7>	CDEFGLP	2
<C; t_3>, <F; t_4>, <H; t_8>	CDEFGH	2
<C; t_3>, <G; t_5>, <Q; t_6>	CDEFGHQ	2
<F; t_4>, <G; t_5>, <Q; t_6>	FGHQ	2

Large 4-sequential patterns (S_4)

L-Transaction Set	Path	Sup.
<A; t_1>, <C; t_3>, <F; t_4>, <G; t_5>	ABCDEFG	2
<A; t_1>, <C; t_3>, <F; t_4>, <Q; t_6>	ABCDEFGHQ	2
<A; t_1>, <C; t_3>, <F; t_4>, <P; t_7>	ABCDEFGLP	2
<A; t_1>, <C; t_3>, <F; t_4>, <H; t_8>	ABCDEFGH	2
<A; t_1>, <C; t_3>, <G; t_5>, <Q; t_6>	ABCDEFGHQ	2
<A; t_1>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ	2
<C; t_3>, <F; t_4>, <G; t_5>, <Q; t_6>	CDEFGHQ	2

Large 5-sequential patterns (S_5)

L-Transaction Set	Path	Sup.
<A; t_1>, <C; t_3>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ	2

Fig. 8. Large sequential patterns generated in sequential-pattern generation phase from the example database D_T.

patterns. Because both algorithm $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ generate S_k along with the generation of C_{k+1}, we use *round* k to refer to the procedure performed to obtain (S_k, C_{k+1}). For algorithm $\mathrm{TJ_{LS}}$, we use *round* k to refer to the procedure performed to obtain (S_k, R_{k+1}). Note that S_1 is obtained in the large-transaction generation phase, we thus use *round one* to refer to the procedure performed to obtain (R_2). These algorithms are devised step by step in light of the features of the candidate generation of sequential patterns and are outlined as follows.

Generalized Descriptions of Algorithms:

1) **Algorithm $\mathrm{TJ_{LS}}$:** By deriving a straightforward extension from prior works, algorithm $\mathrm{TJ_{LS}}$ is devised as a variant of algorithm *a priori* in [6] by using a two-level hash tree in mining large sequential patterns.

2) **Algorithm TJ$_{\mathrm{PT}}$** : In light of the concept of the path trimming technique, algorithm $\mathrm{TJ_{PT}}$ is devised by taking the path into consideration in generating the candidate patterns.
3) **Algorithm TJ$_{\mathrm{PF}}$**: In light of the concept of the pattern family technique, algorithm $\mathrm{TJ_{PF}}$ is devised by using the shared-path tree in generating the candidate patterns.

A. Algorithm TJ_{LS} (Transactionset Join With Large-Transaction Set)

Algorithm $\mathrm{TJ_{LS}}$ is a variant of algorithm *a priori* in [6]. Algorithm $\mathrm{TJ_{LS}}$ essentially utilizes the concept of joining itemsets in association rule mining [6], [55] while solving the discrepancy between large sequential patterns and large itemsets. Similarly to algorithm *a priori* [6], $\mathrm{TJ_{LS}}$ joins the L-transaction sets of large $(k-1)$-sequential patterns for the generation of candidate k-L-transaction sets in the procedure to discover large sequential patterns. However, unlike algorithm *a priori*, $\mathrm{TJ_{LS}}$ employs a two-level hash tree, called the mobile sequence tree, to store the candidate sequential patterns. By utilizing the two-level hashing technique, $\mathrm{TJ_{LS}}$ can join the L-transaction sets to construct the transaction component of the mobile sequence tree in the candidate generation. Then, in the database scan for counting the support, $\mathrm{TJ_{LS}}$ constructs the path component by extracting the corresponding path from the maximal L-transaction sequences whose L-transaction sets contain the corresponding candidate L-transaction sets.

In the two-level hash tree, a node either contains a list of patterns (a leaf node) or a hash table (an internal node). In an internal node, each bucket of the hash table points to another node. The patterns are stored in the leaf nodes. The root of the hash tree is defined to be at depth 1. An internal node at depth d points to nodes at depth $d+1$. When $\mathrm{TJ_{LS}}$ adds a pattern p, $\mathrm{TJ_{LS}}$ starts from the root and go down the tree until reaching a leaf. At an internal node at depth d in the transaction component, $\mathrm{TJ_{LS}}$ decides which branch to follow by applying a hash function to the dth L-transaction of the L-transaction set of pattern p. Similarly, at an internal node at depth g in the path component, $\mathrm{TJ_{LS}}$ decides which branch to follow by applying a hash function to the gth cell of the path of pattern p.

In the beginning of hashing a maximal L-transaction sequence m, $\mathrm{TJ_{LS}}$ finds all the candidate sequential patterns contained in m as follows. If $\mathrm{TJ_{LS}}$ reaches an internal node by hashing the L-transaction l (cell c), it hashes on each L-transaction (cell) that comes after l (c) in m and recursively applies this procedure to the node in the corresponding bucket. If $\mathrm{TJ_{LS}}$ reaches a leaf node, it finds which of the patterns in the leaf node are contained in m and adds support counts to them.

Example 3.1: Fig. 8 is the large sequential patterns generated in sequential-pattern generation phase from the example database D_T, and Fig. 9 is the mobile sequence tree storing S_4 in Fig. 8.

For mining mobile sequential patterns, the first round is executed with large-transaction generation phase to obtain S_1, the set of large 1-sequential patterns, as shown in Fig. 6(c). In addition, algorithm $\mathrm{TJ_{LS}}$ utilizes the L-transactions in S_1 as the seed set for generating R_2, the set of candidate 2-L-transaction sets, which is stored in the transaction component of a mobile sequence tree. In the second round, $\mathrm{TJ_{LS}}$ constructs the complete mobile sequence tree by hashing each combination of 2-L-transactions set in each maximal L-transaction sequence into the transaction component and hashing the corresponding path for constructing the path component, to count the support of each candidate sequential pattern. Then, $\mathrm{TJ_{LS}}$ destructs the mobile sequence tree for deriving S_2, the set of large 2-sequential patterns, and utilizes the L-transaction sets in S_2 for generating R_3. In each subsequent round, $\mathrm{TJ_{LS}}$ starts with candidate L-transaction sets found in the previous round for the counting of supports of candidate sequential patterns and then identifies large sequential patterns. $\mathrm{TJ_{LS}}$ proceeds to the generation of new candidate L-transaction sets and stores them to the mobile sequence tree. The procedure continues until no large sequential patterns are derived.

Example 3.2: To illustrate the operations of algorithm $\mathrm{TJ_{LS}}$, it can be seen from Fig. 9, $\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle\}$ is a candidate 4-L-transaction set generated by joining the L-transaction sets $\{\langle \mathrm{A};\mathrm{t}_1\rangle,\ \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle\}$ and $\{\langle \mathrm{A};\mathrm{t}_1\rangle,\ \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle\}$, respectively, from large 3-sequential patterns $\langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle\}$: ABCDEF$\rangle$ and $\langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle C;\mathrm{t}_3\rangle, \langle G;\mathrm{t}_5\rangle\}$: ABCDEFG$\rangle$. In scanning database phase, algorithm $\mathrm{TJ_{LS}}$ constructs the path component of the mobile sequential tree and counts supports of the candidate sequential patterns. For example, after the transaction component of tree is constructed in the Fig. 9(a), $\mathrm{TJ_{LS}}$ scans the database D_T in Fig. 8 to obtain the path component in Fig. 9(b) while also counting supports. In SID 100, when the support for candidate L-transaction set $\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle G;\mathrm{t}_5\rangle\}$ is being counted, the corresponding path $\langle$ABCDEFG$\rangle$ will be generated in the path component of the mobile sequence tree to account for one support count of $\langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle G;\mathrm{t}_5\rangle\}$: ABCDEFG$\rangle$. Hence, the final support of $\langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle\}$: ABCDEFG$\rangle$ is 2, i.e., from SID 100 and SID 200. Explicitly, the corresponding path is divided into several subpaths by identifying the cells of L-transactions. For the example shown in Fig. 10, algorithm $\mathrm{TJ_{LS}}$ counts the support of L-transaction set $\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle C;\mathrm{t}_3\rangle, \langle \mathrm{P};\mathrm{t}_7\rangle\}$ in SID 100. $\mathrm{TJ_{LS}}$ first locates $\langle \mathrm{A};\mathrm{t}_1\rangle$ on position (1) and $\langle \mathrm{C};\mathrm{t}_3\rangle$ on position (2) in the L-transaction set, and the corresponding subpath $\langle$ABC$\rangle$ is extracted from the subpath $\langle$ABC$\rangle$ shown in (3). Then, $\mathrm{TJ_{LS}}$ locates $\langle \mathrm{C};\mathrm{t}_3\rangle$ on position (2) and $\langle P;\mathrm{t}_7\rangle$ on position (4) in the L-transaction set, and the corresponding subpath $\langle$CDEFGLP$\rangle$ is extracted from the subpath $\langle$CDEFGHQGLP$\rangle$ shown in (5). Note that the redundancy of HQG is eliminated because they cause a cycle between L-transaction $\langle \mathrm{C};\mathrm{t}_3\rangle$ and L-transaction $\langle \mathrm{P};\mathrm{t}_7\rangle$. After scanning database for counting the support of candidate sequential patterns, $\mathrm{TJ_{LS}}$ obtains large sequential patterns in the procedure of destructing the mobile sequence tree. Each large sequential pattern is generated when its support exceeds the minimum support. For example, one can destruct the mobile sequence tree in Fig. 9 to determine S_4 in Fig. 8.

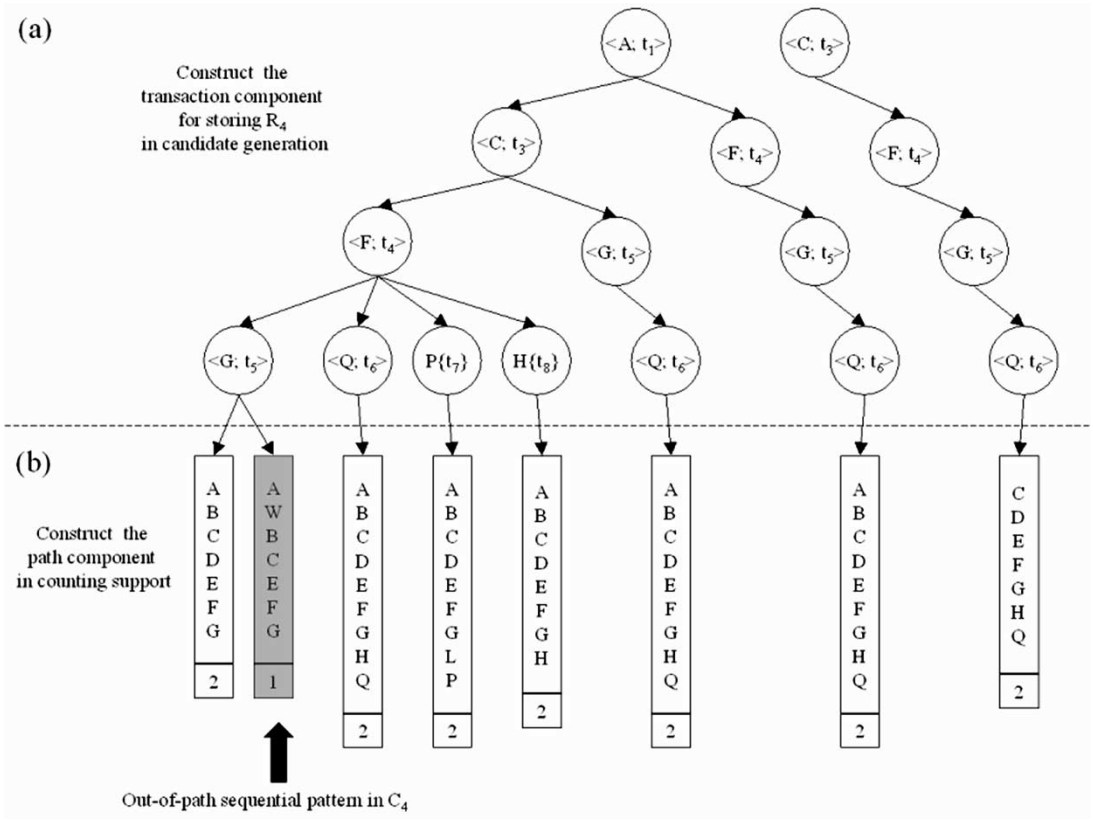


Fig. 9. Data structure of a mobile sequential tree for storing candidate 4-sequential patterns in algorithm $\mathrm{TJ_{LS}}$.

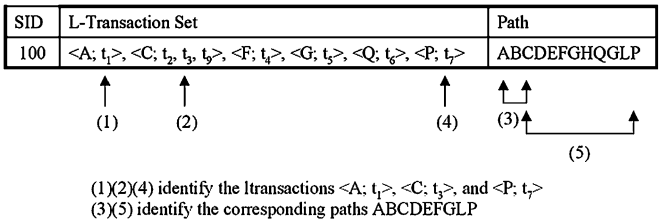


Fig. 10. Procedure of counting support of L-transaction set $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{P}; \mathrm{t}_7\rangle\}$: ABCDEFGLP$\rangle$ in SID 100.

B. Algorithm TJ_{PT} (Transactionset Join With Path Trimming)

Without exploiting the paths of large sequential patterns, algorithm $\mathrm{TJ_{LS}}$ tends to count the supports of a lot of out-of-path sequential patterns (i.e., the sequential patterns that do not stay within the path), thus degrading the performance. In light of the concept of path trimming, algorithm $\mathrm{TJ_{PT}}$ is designed by taking both the L-transaction sets and paths of large sequential patterns into consideration to generate candidate sequential patterns. Explicitly, during the generation of large sequential patterns, by destructing the mobile sequence tree, $\mathrm{TJ_{PT}}$ not only determines large sequential patterns but also maintains a buffer that contains the leaf nodes in the transaction component and the corresponding paths in the path component so as to classify the patterns. The purpose of classifying the patterns is that the patterns, whose paths do not contain each other, need not be considered to generate candidate sequential patterns together. Thus, $\mathrm{TJ_{PT}}$ can trim the generation of candidate sequential patterns according to the paths. This is referred to as the path trimming technique. As a result, $\mathrm{TJ_{PT}}$ utilizes large sequential patterns to generate candidate sequential patterns in the candidate generation for solving the out-of-path sequential pattern problem in $\mathrm{TJ_{LS}}$ mentioned above.

Example 3.3: For the example shown in Fig. 8, in SID 300, when the support for candidate 4-L-transaction set $\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle\}$ is being counted, the corresponding path $\langle$AWBCEFG$\rangle$ will be generated in the path component of mobile sequence tree to account for one support count for $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle\}$: AWBCEFG$\rangle$. Note that $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle\}$: AWBCEFG$\rangle$ has four subpatterns including $\langle\{\langle A; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle\}$: AWBCEF$\rangle$, $\rangle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle\}$: AWBCEFG$\rangle$, $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle\}\rangle$: AWBCEFG$\rangle$, and $\langle\{\langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle\}$: CEFG$\rangle$. However, all of them are not large 3-sequential patterns. Instead, they are out-of-path 3-sequential patterns in round 3 in the sense that not all of their subpatterns are large 2-sequential patterns. Explicitly, only $\langle\{\langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle\}$: FG$\rangle$ is a large 2-sequential pattern in this case. However, algorithm $\mathrm{TJ_{LS}}$ still counts the supports of them in round 3. In algorithm $\mathrm{TJ_{LS}}$, out-of-path sequential patterns will be generated in each round if the candidate L-transaction sets are contained in the L-transaction sets of maximal L-transaction sequences in D_T. For example, the out-of-path sequential patterns generated by

SID	L-Transaction Set	Path
300	<A; t_1>, <C; t_3>, <F; t_4>, <G; t_5>	AWBCEFG

(a)

L-Transaction Set (in R_3)	Path	Sup.
<A; t_1>, <C; t_3>, <F; t_4>	AWBCEF	1
<A; t_1>, <C; t_3>, <G; t_5>	AWBCEFG	1
<A; t_1>, <F; t_4>, <G; t_5>	AWBCEFG	1
<C; t_3>, <F; t_4>, <G; t_5>	CEFG	1

(b)

L-Transaction Set (in R_4)	Path	Sup.
<A; t_1>, <C; t_3>, <F; t_4>, <G; t_5>	AWBCEFG	1

(c)

Fig. 11. Example for describing the out-of-path sequential pattern problem in SID 300 caused by algorithm $\mathrm{TJ_{LS}}$.

SID 300 are shown in Fig. 11. Such an out-of-path sequential pattern problem will happen in round k for $k > 2$. This in turn implies that one can trim the support counting of the redundant sequential patterns according to the paths traversed.

Recall that S_k represents the set of large k-sequential patterns and C_k is the set of candidate k-sequential patterns. In the candidate generation phase, $\mathrm{TJ_{PT}}$ constructs both the transaction and path components of mobile sequential tree for storing C_k. In the candidate generation, $\mathrm{TJ_{PT}}$ joins the L-transaction sets of large $(k-1)$-sequential patterns for the generation of candidate k -L-transaction set and compares the paths of large $(k-1)$-sequential patterns. If one path does not contain the other path, the generated candidate k-L-transaction set is trimmed. If one path p contains the other path q, $\mathrm{TJ_{PT}}$ generates a candidate k-sequential patterns consisting of the candidate k-L-transaction set and p.

Example 3.4: Consider the example scenario shown in Fig. 12. In algorithm $\mathrm{TJ_{PT}}$, the candidate 5-sequential pattern $\langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle, \langle \mathrm{Q};\mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$ in Fig. 12(a) is generated by joining L-transaction set $\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle\}$ in subpattern $\langle 1\rangle$ and L-transaction set $\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{Q};\mathrm{t}_6\rangle\}$ in subpattern $\langle 2\rangle$ with the path trimming technique to identify the fact that path $\langle$ABCDEFGHQ$\rangle$ contains path $\langle$ABCDEFG$\rangle$. Finally, $\langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle, \langle \mathrm{Q};\mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$ is qualified as a candidate 5-sequential pattern after $\mathrm{TJ_{PT}}$ identifies that the other subpatterns (i.e., subpatterns $\langle 3\rangle, \langle 4\rangle$, and $\langle 5\rangle$) are large 4-sequential patterns in Fig. 12(b).

By classifying the large k-sequential patterns, $\mathrm{TJ_{PT}}$ can efficiently generate candidate k-sequential patterns. Particularly, by classifying the patterns in S_k for $k \geq 2$, $\mathrm{TJ_{PT}}$ will not generate any out-of-path $(k+1)$-sequential pattern. This demonstrates the very advantage of the path trimming technique $\mathrm{TJ_{PT}}$ employs.

Example 3.5: For the example shown in Fig. 9, $\mathrm{TJ_{PT}}$ generates the complete mobile sequence tree by hashing not only L-transaction sets but also paths in candidate generation so that the path $\langle$AWBCEFG$\rangle$ will not be counted for the support. Note that such out-of-path sequential patterns as the one shown in Fig. 11 will not occur anymore, showing a significant performance improvement of $\mathrm{TJ_{PT}}$ over $\mathrm{TJ_{LS}}$.

C. Algorithm $\mathrm{TJ_{PF}}$ (Transactionset Join With Pattern Family)

Algorithm $\mathrm{TJ_{PF}}$ is similar to algorithm $\mathrm{TJ_{PT}}$ in that it employs the concept of utilizing large sequential patterns for generating candidate sequential patterns to reduce the computational overhead caused by out-of-path sequential patterns but is different from the latter in that algorithm $\mathrm{TJ_{PF}}$ by utilizing the information in patterns and is able to reduce the number of uncertain candidate sequential patterns and store candidate sequential patterns with a compact approach, thus further reducing the corresponding overhead. Recall that algorithm $\mathrm{TJ_{PT}}$ utilizes path trimming technique for the generation of candidate sequential patterns by comparing the paths of its subpatterns to identify if one path contains another.

Example 3.6: For the example in Fig. 12, algorithm $\mathrm{TJ_{PT}}$ generates the candidate 5-sequential pattern in Fig. 12(a) by joining L-transaction set $\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle\}$ in subpattern $\langle 1\rangle$ and L-transaction set $\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{Q};\mathrm{t}_6\rangle\}$ in subpattern $\langle 2\rangle$ with the path trimming technique to identify that path $\langle$ABCDEFGHQ$\rangle$ contains path $\langle$ABCDEFG$\rangle$.

Comparing the paths of subpatterns incurs $O(|P|)$ computation, where $|P|$ is the average path length of large sequential patterns. In addition, algorithm $\mathrm{TJ_{PT}}$ is required to store the same paths as the branches in different subtrees of the transaction component in the mobile sequence tree, which incurs an excessive use of memory. Note that even by treating the cells as other items in the patterns, modified algorithm GSP still needs to compare the whole cells in the path and incurs $O(|P|)$ computation. Hence, algorithm $\mathrm{TJ_{PF}}$ surpasses algorithm $\mathrm{TJ_{PT}}$ and the modified GSP in that with the pattern family technique, $\mathrm{TJ_{PF}}$ is able to generate a more compact tree to store the patterns to minimize the corresponding overhead.

1) Remarks of Algorithm $\mathrm{TJ_{PF}}$*:* Algorithm $\mathrm{TJ_{PF}}$ is devised in light of the pattern family technique. To facilitate our description of algorithm $\mathrm{TJ_{PF}}$, some theoretical properties of pattern family are devised below.

Definition 1: A *maximal sequential pattern* is a large sequential pattern that is not contained in any other large sequential pattern. For each maximal sequential pattern, its *pattern family* consists of the pattern itself and all its subpatterns generated in each round.

Example 3.7: For the example shown in Fig. 8, one of the maximal sequential patterns is $\langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \langle \mathrm{C};\mathrm{t}_3\rangle, \langle \mathrm{F};\mathrm{t}_4\rangle, \langle \mathrm{G};\mathrm{t}_5\rangle, \langle Q;\mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$ which is also a large 5-sequential pattern. The corresponding pattern family is shown in Fig. 13.

Definition 2: For a pattern family whose maximal sequential pattern is s_k which consists of L-transactions $\{x_1, x_2, \ldots, x_k\}$ and path $\langle m_1 m_2 \ldots m_q\rangle$, a *maximal-path large 2-sequential pattern* (abbreviatedly as $\mathrm{MS_2}$), $\rangle\{x_1, x_k\}$: $m_1 m_2 \ldots m_q\rangle$, is a large 2-sequential pattern which has the same path as the maximal sequential pattern of this pattern family.

Example 3.8: For each pattern family, it is noted that $\mathrm{MS_2}$ is the large 2-sequential subpattern with the maximal path. For the example shown in Fig. 13, patterns marked gray are the patterns with $\mathrm{MS_2} = \langle\{\langle \mathrm{A};\mathrm{t}_1\rangle, \rangle Q;\mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$, which is the

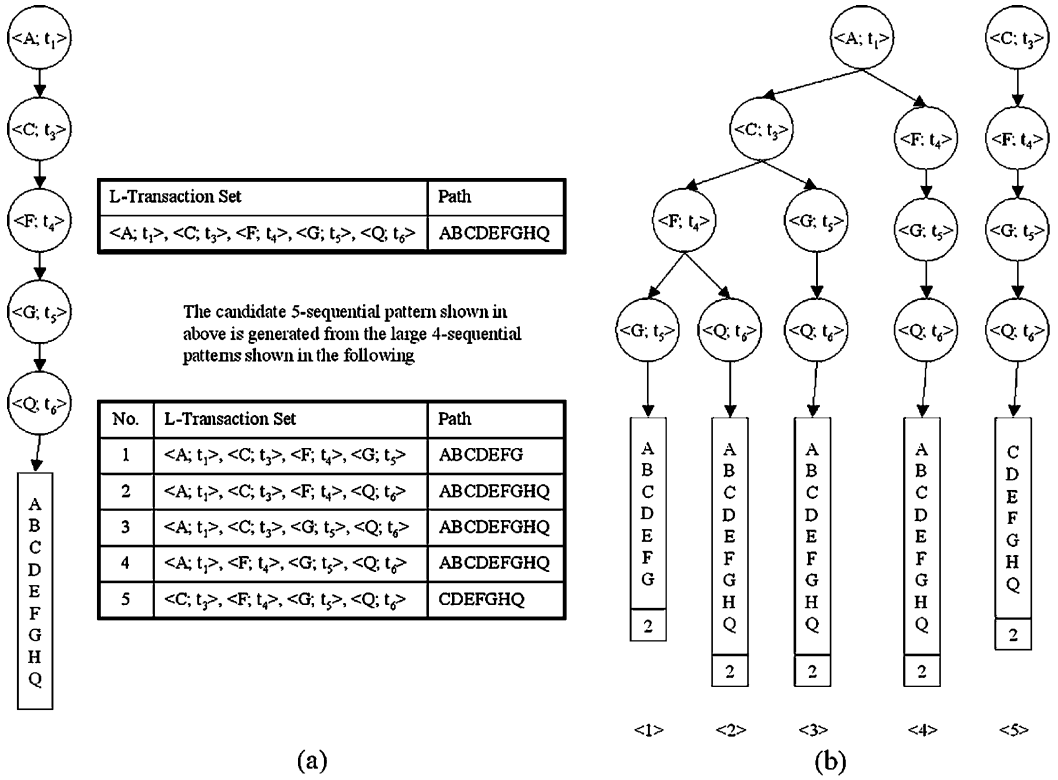


Fig. 12. Candidate 5-sequential pattern shown in (a) is generated by identifying the existence of its five large 4-sequential subpatterns shown in (b).

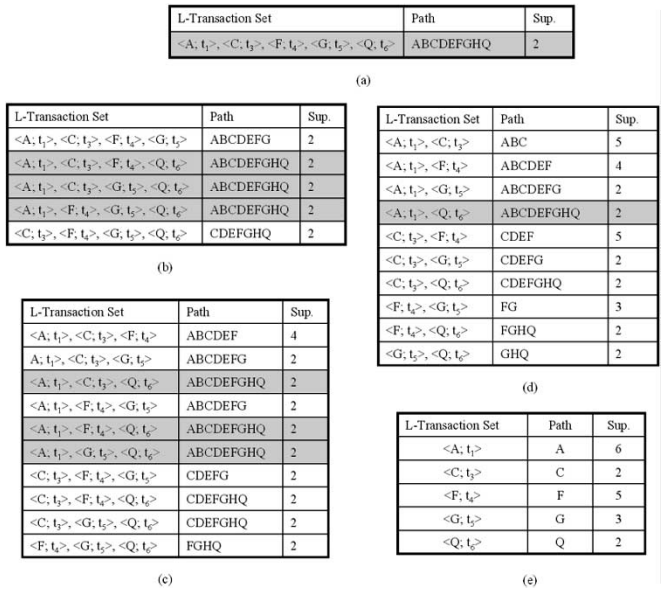


Fig. 13. One pattern family example. Patterns marked gray are the patterns having the maximal-path large 2-sequential pattern.

large 2-sequential pattern with path length equal to 9, larger than those of other large 2-sequential patterns.

Definition 3: Suppose s_k, $k \geq 3$, is the maximal sequential pattern of a pattern family, and s_k consists of L-transactions $\{x_1, x_2, \ldots, x_k\}$ and path $\langle m_1 m_2 \ldots m_q \rangle$. The *centro-subtransactionset* of a pattern in this pattern family is $\{x_2, \ldots, x_{k-1}\}$.

Example 3.9: For example, the large 4-sequential pattern $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \rangle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{Q}; \mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$ can be viewed as two parts, i.e., $\mathrm{MS}_2 = \langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{Q}; \mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$ and the centro-subtransactionset being $\{\langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle\}$. Note that a large k-sequential pattern is a pattern consisting of k L-transactions and a path. For each large k-sequential pattern, all its k $(k-1)$-sequential subpatterns are large. Explicitly, for a large k-sequential pattern with L-transaction $\{x_1, x_2, \ldots, x_k\}$, the L-transactions of its k $(k-1)$-sequential subpatterns can be represented by $\{x_2, x_3, \ldots, x_k\}, \{x_1, x_3, \ldots, x_k\}, \ldots$, and $\{x_1, x_2, \ldots, x_{k-1}\}$. Then, we have the following remarks.

Remark 1: For a large k-sequential pattern p_k, $k \geq 3$, there exist at least $k-2$ large $(k-1)$-subpatterns whose paths are identical to that of p_k.

Example 3.10: For the large 5-sequential pattern $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle, \langle \mathrm{Q}; \mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$ shown in Fig. 13(a), there exist 3 large 4-subpatterns, $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle Q; \mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$, $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{C}; \mathrm{t}_3\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle, \langle \mathrm{Q}; \mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$, and $\langle\{\langle \mathrm{A}; \mathrm{t}_1\rangle, \langle \mathrm{F}; \mathrm{t}_4\rangle, \langle \mathrm{G}; \mathrm{t}_5\rangle, \langle \mathrm{Q}; \mathrm{t}_6\rangle\}$: ABCDEFGHQ$\rangle$, shown in Fig. 13(b), whose paths are the same with the one of the large 5-sequential pattern.

Remark 2: Note that a maximal sequential pattern is also a large sequential pattern. Thus, for a maximal sequential pattern s_k with L-transactions $\{x_1, x_2, \ldots, x_k\}$ and path $\langle m_1 m_2 \ldots m_q \rangle$, there exist $k-2$ large $(k-1)$-sequential subpatterns which have identical maximal-path large 2 -sequential pattern $\langle\{x_1, x_k\}: m_1 m_2 \ldots m_q\rangle$.

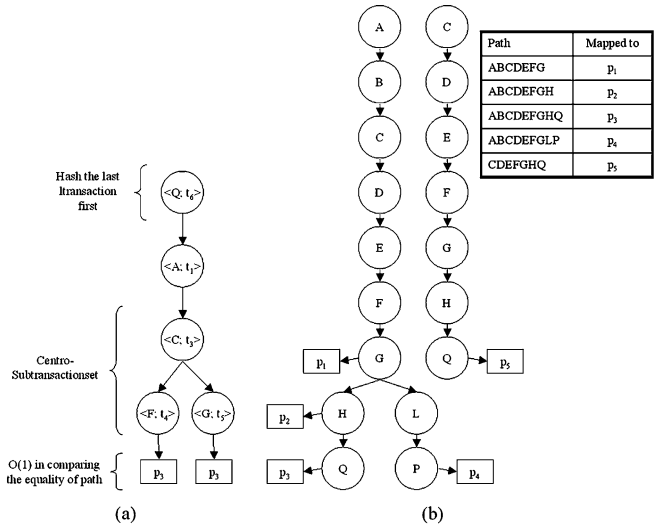


Fig. 14. Algorithm $\mathrm{TJ_{PF}}$ hashes the last L-transaction first so that the centro-subtransactionset is identified and compares the individual integers stored in shared-path tree.

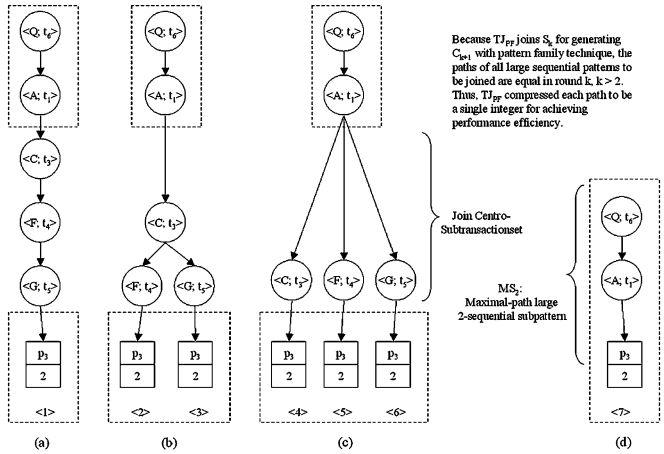


Fig. 15. Algorithm $\mathrm{TJ_{PF}}$ joins the centro-subtransactionsets with integer comparison.

2) Algorithm $\mathrm{TJ_{PF}}$ *Using Shared-Path Tree in Candidate Generation:* Algorithm $\mathrm{TJ_{PF}}$ is able to generate a maximal sequential pattern s_k with L-transactions $\{x_1, x_2, \ldots, x_k\}$ and path $\langle m_1 m_2 \ldots m_q \rangle$ as a candidate sequential pattern by joining the previous large sequential patterns $\langle \{x_1, (x_2, x_3, \ldots, x_{k-3}, x_{k-2}), x_k\}: m_1, m_2, \ldots, m_q \rangle$ and $\langle \{x_1, (x_2, x_3, \ldots, x_{k-3}, x_{k-1}), x_k\} : m_1, m_2, \ldots, m_q \rangle$ with the pattern family technique. Explicitly, $\mathrm{TJ_{PF}}$ obtains: 1) $\{x_2, x_3, \ldots, x_{k-3}, x_{k-2}, x_{k-1}\}$ by joining the centro-subtransactionsets $\{x_2, x_3, \ldots, x_{k-3}, x_{k-2}\}$ and $\{x_2, x_3, \ldots, x_{k-3}, x_{k-1}\}$ and 2) the new $\mathrm{MS}_2 = \langle \{x_1, x_k\}: m_1, m_2, \ldots, m_q \rangle$ by comparing the MS_2's in the previous large sequential patterns. By hashing the last L-transaction first and storing an integer for each path, $\mathrm{TJ_{PF}}$ constructs the mobile sequence tree with a form that for each candidate sequential patterns, two L-transactions of MS_2 come first, a centro-subtransactionset is in the middle, and an integer, which is returned from shared-path tree for indexing the corresponding path, is in the leaf.

Example 3.11: For example, $\mathrm{TJ_{PF}}$ stores the candidate sequential patterns $\langle \{\langle \mathrm{A}; \mathrm{t}_1 \rangle, \langle \mathrm{C}; \mathrm{t}_3 \rangle, \langle \mathrm{F}; \mathrm{t}_4 \rangle, \langle \mathrm{Q}; \mathrm{t}_6 \rangle\}$: ABCDEFGHQ$\rangle$ and $\langle \{\langle \mathrm{A}; \mathrm{t}_1 \rangle, \langle \mathrm{C}; \mathrm{t}_3 \rangle, \langle \mathrm{G}; \mathrm{t}_5 \rangle, \langle Q; \mathrm{t}_6 \rangle\}$: ABCDEFGHQ$\rangle$ in Fig. 12(b) into the mobile sequence tree as in Fig. 14(a). $\mathrm{TJ_{PF}}$ hashes the last L-transaction, $\langle \mathrm{Q}; \mathrm{t}_6 \rangle$, in the first position of the mobile sequence tree. Note that path $\langle$ABCDEFGHQ$\rangle$ is represented by the integer $\langle \mathrm{p}_3 \rangle$, derived from the mapping of the shared-path tree in Fig. 14(b). Then, $\mathrm{TJ_{PF}}$ can join the L-transaction sets of these two large 4 -sequential patterns, i.e., pattern $\langle 2 \rangle$ and $\langle 3 \rangle$ shown in Fig. 15(b), with a buffer to keep a block that contains the leaf nodes in the transaction component and the corresponding integers in the path component to classify the patterns for generating the candidate 5-sequential pattern in Fig. 12(a) efficiently.

From Remark 2, we know that $\mathrm{TJ_{PF}}$ joins S_k for generating C_{k+1} with the pattern family technique, and the paths of all large sequential patterns to be joined are identical to one another in round k, $k \geq 3$. Thus, $\mathrm{TJ_{PF}}$ joins the centro-subtransactionsets with comparing individual integers for achieving performance improvement.

Example 3.12: For the example shown in Fig. 15, $\mathrm{TJ_{PF}}$ joins the centro-subtransactionsets in pattern $\langle 4 \rangle$ and $\langle 5 \rangle$ shown in Fig. 15(c) with comparing integers which are equal to each other (i.e., p_3) to generate pattern $\langle 2 \rangle$ in Fig. 15(b). Similarly, $\mathrm{TJ_{PF}}$ joins the centro-subtransactionsets in pattern $\langle 2 \rangle$ and $\langle 3 \rangle$ shown in Fig. 15(b) by comparing integers to generate pattern $\langle 1 \rangle$ in Fig. 15(b).

The method for algorithm $\mathrm{TJ_{PF}}$ to reduce computational overhead and memory consumption is as follows. In the first round, algorithm $\mathrm{TJ_{PF}}$ also joins the L-transactions in S_1 for generating R_2 to be stored in the transaction component of a mobile sequence tree. However, in the second round, $\mathrm{TJ_{PF}}$ hashes each combination of 2-L-transactions set in each maximal L-transaction sequence into the transaction component by hashing the last L-transaction first. Then, $\mathrm{TJ_{PF}}$ hashes the corresponding path into the shared-path tree which has an assigned integer in each leaf node for representing the path from the root node to the parent node of that leaf node. $\mathrm{TJ_{PF}}$ next returns the integer for constructing the path component of the mobile sequence tree while keeping counting the support. After the candidate 2 -sequential patterns with the minimum support are identified as the large 2-sequential patterns, algorithm $\mathrm{TJ_{PF}}$ joins the L-transaction sets with the path trimming technique to generate candidate 3-sequential patterns. Note that the shared-path tree constructed in round two will be used for the mapping between paths and integers in the following rounds.

Example 3.13: For example, the shared-path tree shown in Fig. 14(b) maps the paths of large 4-sequential patterns in Fig. 12(b) into integers $\{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3, \mathrm{p}_4, \mathrm{p}_5\}$.

In the third round, $\mathrm{TJ_{PF}}$ counts the supports of candidate 3-sequential patterns by hashing into the mobile sequence tree the L-transaction sets and integers returned from the shared-path tree. In destructing the mobile sequence tree, $\mathrm{TJ_{PF}}$ not only determines large 3-sequential patterns but also uses a buffer

L-Transaction Set	Path
<A; t_1>, <C; t_2>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ
<A; t_1>, <C; t_2>, <F; t_4>, <G; t_5>, <P; t_7>	ABCDEFGLP
<A; t_1>, <C; t_2>, <F; t_4>, <G; t_5>, <H; t_8>	ABCDEFGH
<A; t_1>, <C; t_2>, <F; t_4>, <H; t_8>, <Q; t_9>	ABCDEFGHQ

(a)

L-Transaction Set	Path
<A; t_1>, <C; t_2>, <F; t_4>, <G; t_5>, <Q; t_6>	ABCDEFGHQ

(b)

Fig. 16. Uncertain candidate transaction patterns generated by $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$.

to keep a block that contains the leaf nodes in the transaction component and the corresponding integers in the path component to classify the patterns. By hashing the last L-transaction first, the mobile sequence tree is well-structured and $\mathrm{TJ_{PF}}$ compares the individual integers to trim the generation of candidate sequential patterns according to the paths. In each subsequent round, $\mathrm{TJ_{PF}}$ constructs the mobile sequence tree by hashing the last L-transaction first and utilizing the shared-path tree for mapping so that $\mathrm{TJ_{PF}}$ compares the individual integers in trimming the generation of candidate sequential patterns. Algorithm $\mathrm{TJ_{PF}}$ thus has $O(1)$ execution time complexity in this step, better than $O(|P|)$ by algorithm $\mathrm{TJ_{PT}}$, where $|P|$ is the average path length of large sequential patterns. In addition, unlike algorithm $\mathrm{TJ_{PT}}$, algorithm $\mathrm{TJ_{PF}}$ utilizes MS_2 to filter out some uncertain candidate sequential patterns before subpattern identification. For a candidate k-sequential patterns, it should have k large $(k-1)$-sequential patterns. For an uncertain candidate k-sequential patterns, it is generated by joining two large $(k-1)$-sequential patterns. Thus, for proving that an uncertain candidate k-sequential patterns is qualified as a candidate k-sequential patterns, there are $k-2$ subpattern identifications the need to be conducted.

Example 3.14: For example, taking the large 4 -sequential patterns shown in Fig. 8, $\mathrm{TJ_{PT}}$ generates four uncertain candidate 5-sequential patterns shown in Fig. 16(a). However, by utilizing the pattern family technique, $\mathrm{TJ_{PF}}$ only generates one uncertain candidate 5-sequential pattern shown in Fig. 16(b). Thus, $\mathrm{TJ_{PT}}$ conducts 12 subpattern identifications and $\mathrm{TJ_{PF}}$ conducts three subpattern identifications. This demonstrates the very advantage of the pattern family technique $\mathrm{TJ_{PF}}$ employs.

IV. Experimental Results

To assess the performance of $\mathrm{TJ_{LS}}$, $\mathrm{TJ_{PT}}$, and $\mathrm{TJ_{PF}}$, we conducted several experiments to determine large sequential patterns. These experiments are performed on a computer with a 1-GHz Intel CPU and 512 MB of memory. The method used to generate synthetic data is described in Section IV-A. In Section IV-B, performance of $\mathrm{TJ_{LS}}$, $\mathrm{TJ_{PT}}$, and $\mathrm{TJ_{PF}}$ is comparatively studied.

A. Generation of Synthetic Mobile Transaction Sequences

In the experiments, the moving scenario with transactions made in a mobile commerce environment is simulated. Since the mobile commerce service is a new application in the near future, we believe that the customers have the similar behaviors to those of them in the current data network when they first use this service. After this service is used by customers, the behaviors will then be changed according to their usage experiences. Currently, there is no real scenario that we can mimic. Thus, in this paper, the simulation model for generating synthetic mobile transaction sequences is in fact similar to that in the companion papers [43], [49]. Explicitly, the method for generating moving patterns is similar to that in [49] and the method for generating transactions is similar to that in [43].

Notation	Meaning
$\|D\|$	The number of mobile transaction sequences (size of database)
s	Minimum support
$\|T\|$	Average number of items per transaction
$\|P\|$	Average size of path length per mobile transaction sequence
P_0	Backward weight in probability to the cell user came from
P_a	Advancing power of items sold in a neighbor
P_d	Damping factor because of backward movement
n_I	The range of the number of items sold in each cell
P_b	The probability that user makes the transaction in the cell

Fig. 17. Parameters used in the simulation.

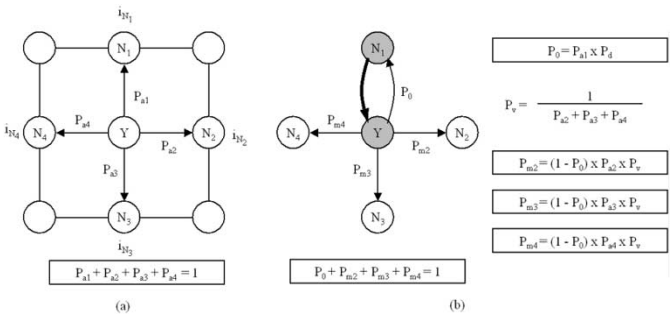


Fig. 18. Mesh network to simulate mobile commerce environment.

Fig. 17 summarizes the meanings of various parameters used in the experiments. First, we construct an $n \times n$ mesh network [40] with a modification by taking the geographic boundary into consideration to limit the number of neighbors so as to mimic the mobile environment, where each node represents one cell [49]. The number of items in each cell is determined from a uniform distribution within a given range, denoted by n_I. For each cell, the advancing probability P_a of each neighbor is the probability for a customer to move to neighboring cells to purchase the items sold there. In essence, each directed edge from one cell A to another cell B is assigned with a weight, corresponding to the advancing probability of B for A. In the model, the advancing probability is obtained by the ratio of the number of items sold in each neighbor to those numbers of other neighbors. For the 3×3 mesh network example shown in Fig. 18(a), there are four neighbors $\langle N_1, N_2, N_3, N_4 \rangle$ for cell Y with the corresponding advancing probabilities $\langle P_{a1}, P_{a2}, P_{a3}, P_{a4} \rangle$. In addition, $\langle i_{N_1}, i_{N_2}, i_{N_3}, i_{N_4} \rangle$ are the numbers of items sold in cells $\langle N_1, N_2, N_3, N_4 \rangle$ and we have $P_{a1} = (i_{N_1})/(i_{N_1} + i_{N_2} + i_{N_3} + i_{N_4})$ and $P_{a2} = (i_{N_2})/(i_{N_1} + i_{N_2} + i_{N_3} + i_{N_4})$.

In the experiments, $|D|$ is the number of mobile transaction sequences generated. When a customer moves among cells for

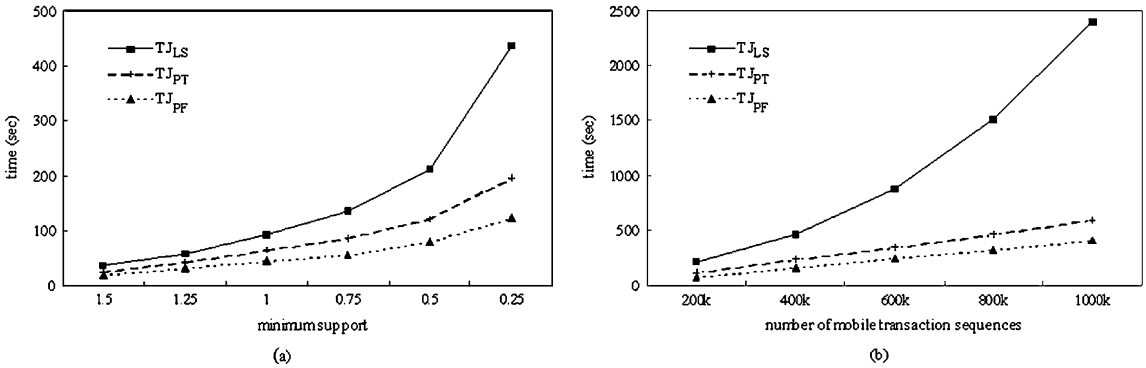


Fig. 19. (a) Execution time of algorithms $\mathrm{TJ_{LS}}$, $\mathrm{TJ_{PT}}$, and $\mathrm{TJ_{PF}}$ when minimum support varies and (b) execution time of algorithms $\mathrm{TJ_{LS}}$, $\mathrm{TJ_{PT}}$, and $\mathrm{TJ_{PF}}$ when number of mobile transaction sequences varies.

shopping in the MC environment, the mobile transaction sequence completed by this customer consists of a moving path and a set of transactions made in the corresponding cells. The starting position of each mobile sequential pattern can be either vistor location register (VLR) or home location register (HLR) and is randomly selected among these cells [34]. A moving path consists of cells moved by a user. The size of each moving path is determined from a Poisson distribution with mean equal to $|P|$. When a customer moves to a cell, the probability that this customer makes the transaction in this cell is denoted by P_b. Note that the number of items in each cell is determined from a uniform distribution within a given range n_I. For each cell, once the number of items is determined, the items that could be purchased in each cell are fixed. The method for generating transaction data in each cell is similar to the one in the prior work [43]. In the mobile commerce environment, people tend to buy sets of items together, which are also called potential maximal frequent sets. The size of the maximal elements is clustered around a mean with a few long itemsets. A transaction may contain one or more of such frequent sets. The transaction size is also clustered around a mean, which is denoted $|T|$. The probability that a user will move from the current cell back to the cell from which he/she came, called the backward weight, is denoted by P_0, which is equal to $P_a \times P_d$, where P_d is a damping factor because of the backward movement. Without loss of generality, P_d is set to 0.8 in our experiments. The probability of moving to each neighbor P_m is also determined by the advancing probability and the sum of the weights for all these cells is equal to $1 - P_0$. For the mesh network shown in Fig. 18(a), when one user visits cell Y from cell N_1, the probabilities of the neighbors that this user will move to are shown in Fig. 18(b).

B. Performance Comparison

In the following experiments, we construct an 8×8 mesh network and set $|D| = 200$ K, $s = 0.5\%, n_I = 200, P_b = 0.5, P_d = 0.8, |T| = 4$, and $|P| = 20$.

1) Experiment One: When the Minimum Support Varies: In this experiment, s varies from 1.5% to 0.25%. Fig. 19(a) shows that $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ in general, outperform $\mathrm{TJ_{LS}}$ for various minimum supports. With the path trimming and the pattern family techniques, both $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ can generate fewer candidate sequential patterns than $\mathrm{TJ_{LS}}$, which suffers a lot of out-of-path sequential patterns in every round. As the minimum support decreases, the execution times of all the algorithms increase because of the increases in the total number of candidate and large sequential patterns.

2) Experiment Two: When the Number of Mobile Transaction Sequences Varies: In this experiment, $|D|$ varies from 200 to 1000 K. Fig. 19(b) shows that the execution times of $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ increase linearly as the database size increases, indicating the good scale-up feature of $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$.

3) Experiment Three: When Purchasing Probability Varies: Note that algorithm $\mathrm{TJ_{LS}}$ suffers the out-of-path sequential pattern problem. To address this problem, we conduct this experiment with the purchase probability P_b varying from 0.5 to 0.3, and the result is shown in Fig. 20(a). For each algorithm, its execution time is taken as the base point when P_b is 0.5, and Fig. 20(a) shows the execution time when P_b varies. When the purchase probability decreases, the execution times of all the algorithms decrease because of the decreases in the total number of candidate and large sequential patterns. However, the path lengths of the out-of-path sequential patterns increase because the average number of cells visited per transaction increases. Note that although the total number of candidate and large sequential patterns decreases, the out-of-path sequential pattern problem causes algorithm $\mathrm{TJ_{LS}}$ to still count the supports of nonlarge sequential patterns. As a result, when P_b decreases, the decrease of the execution time of $\mathrm{TJ_{LS}}$ is not as prominent as those of $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$. To provide more insight into the performance comparisons of algorithms, it is shown in Fig. 21 that $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ outperform $\mathrm{TJ_{LS}}$ in different database sizes, which indicates that $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ are robust in the sensitivity analysis of the purchasing probability.

4) Experiment Four: When the Average Path Length Varies: To examine the sensitivity of varying the average path length, $|P|$ varies from 10 to 30. The result is shown in Fig. 20(b). For each algorithm, its execution time is taken as the base point when $|P|$ is 10, and Fig. 20(b) shows the execution time when $|P|$ varies. It can be seen that $\mathrm{TJ_{PF}}$ is less sensitive to the variation of path length than $\mathrm{TJ_{PT}}$. This agrees with the fact that $\mathrm{TJ_{PF}}$ has $O(1)$ execution time for comparing the path in the candidate generation stage, whereas the corresponding complexity of $\mathrm{TJ_{PT}}$ is

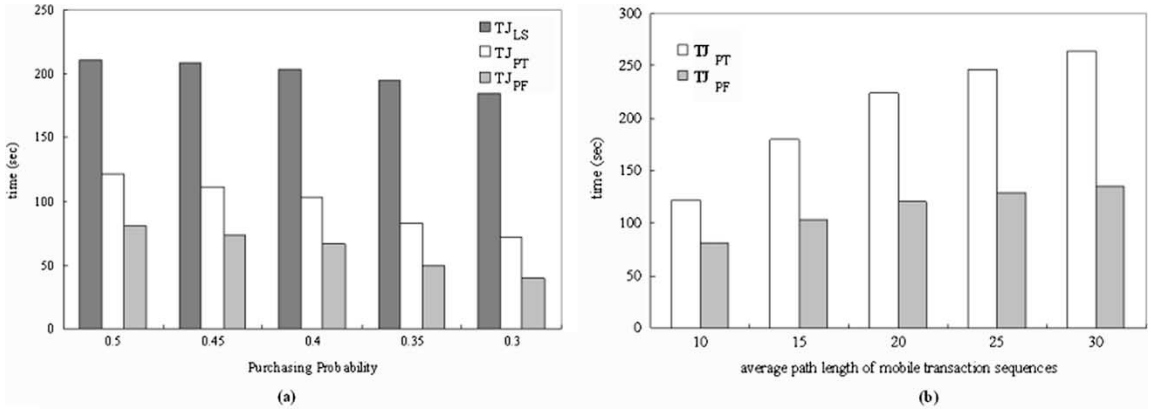


Fig. 20. (a) Execution time of algorithms $\mathrm{TJ_{LS}}$, $\mathrm{TJ_{PT}}$, and $\mathrm{TJ_{PF}}$ when purchasing probability varies and (b) execution time of algorithms $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ when average path length of mobile transaction sequences varies.

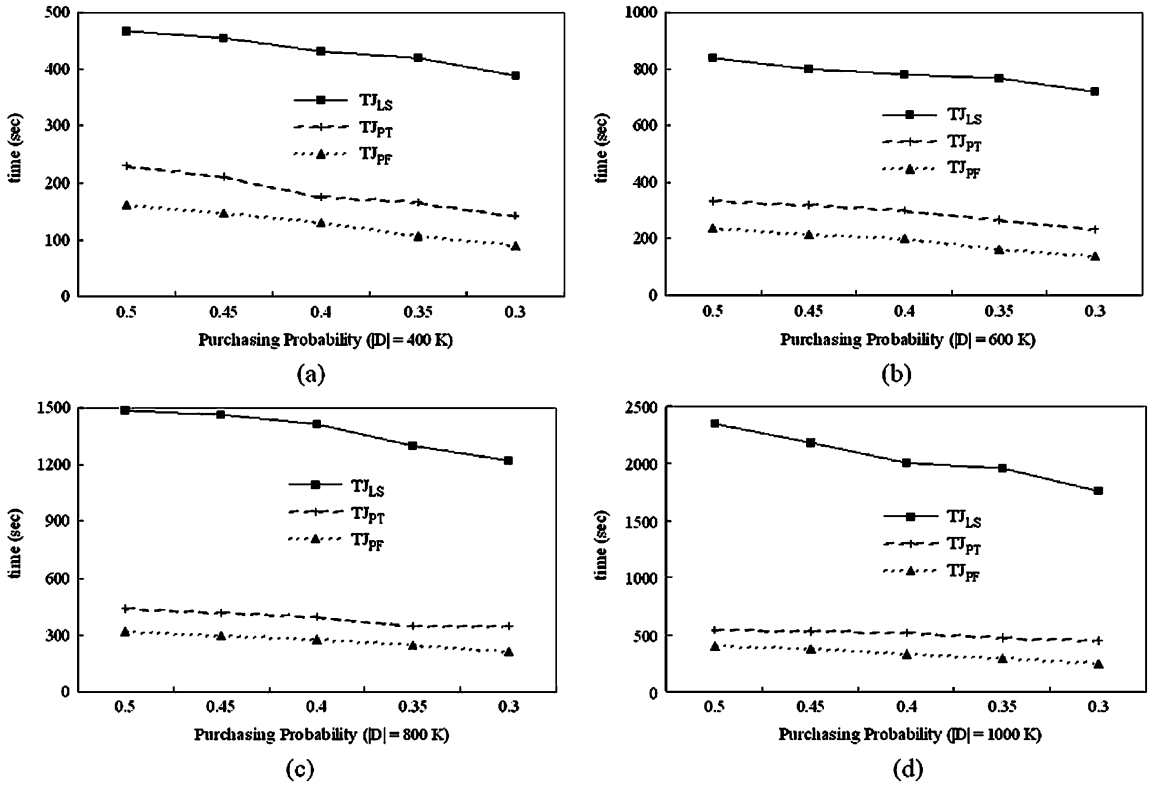


Fig. 21. Execution time of algorithms $\mathrm{TJ_{LS}}$, $\mathrm{TJ_{PT}}$, and $\mathrm{TJ_{PF}}$ when purchasing probability varies in different database sizes.

$O(|P|)$. In addition, it is also shown in Fig. 22 that $\mathrm{TJ_{PF}}$ outperforms $\mathrm{TJ_{PT}}$ in the sensitivity analysis of the average path length with different database sizes. To provide more insight into the candidate generation stage of $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$, it is shown in Fig. 23 that the ratio $(TJ_{PT})/(TJ_{PF})$ of execution time which is incurred by comparing the path is almost equal to $(O(|P|))/(O(1))$.

5) Experiment Five: Performance Comparison Between $\mathrm{TJ_{PT}}$ *and* $\mathrm{TJ_{PF}}$ *in Each Round:* To provide more insights into the shared path tree feature exploited by pattern family technique, we set $|D| = 200\ 000, s = 0.5\%, n_I = 200, P_b = 0.5, P_d = 0.8, |T| = 4$, and $|P| = 20$ and compare the performance of $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ in each round. Because S_1 is obtained in the large-transaction generation phase, we thus use round one to refer to the procedure performed to obtain (R_2) and use round two to refer to the procedure performed to obtain (C_2, S_2, C_3). Note that $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ generate S_k along with the generation of C_{k+1}, we use round $k, k \geq 3$ to refer to the procedure performed to obtain (S_k, C_{k+1}). As shown in Fig. 24(a), $\mathrm{TJ_{PF}}$ consistently outperforms $\mathrm{TJ_{PT}}$ in all rounds, except round one. This agrees with our intuition. Note that in round one, without any path information, both $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$

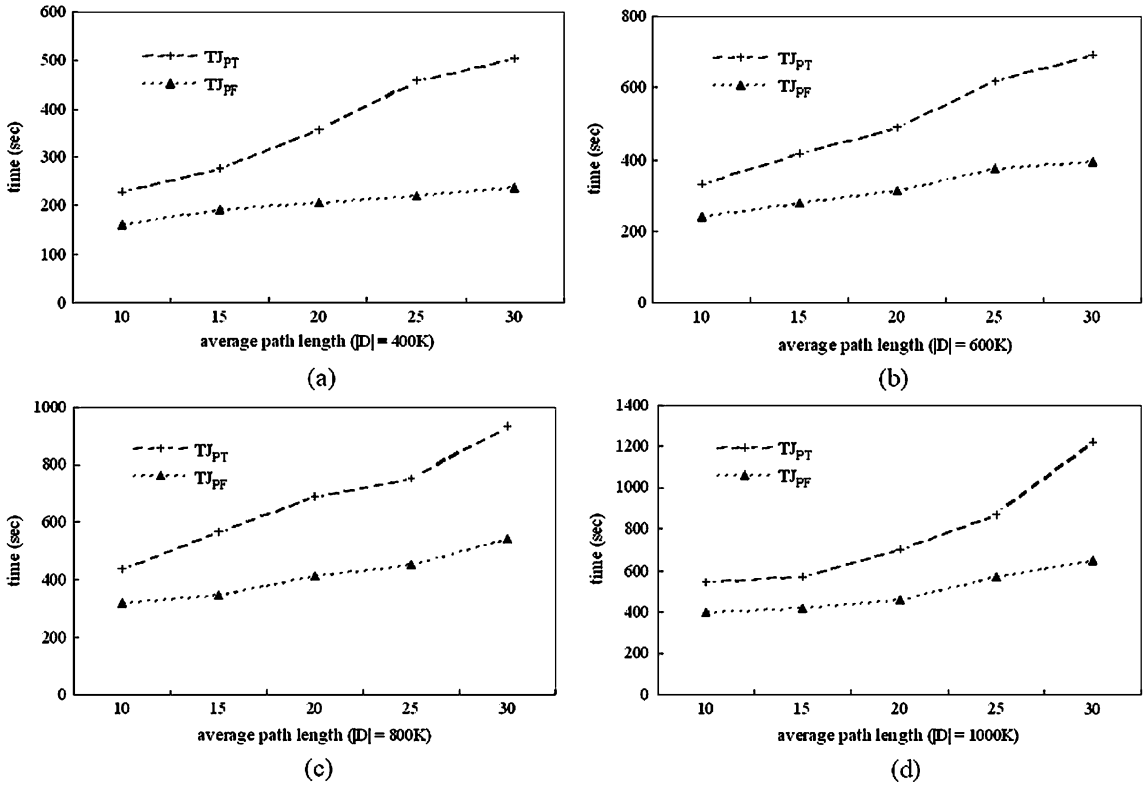


Fig. 22. Execution time of algorithms $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ when average path length of mobile transaction sequences varies in different database sizes.

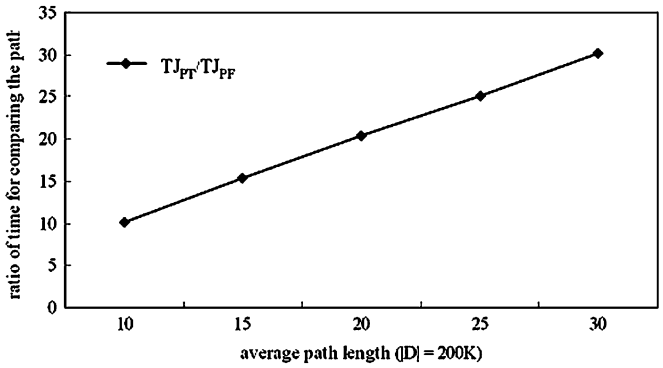
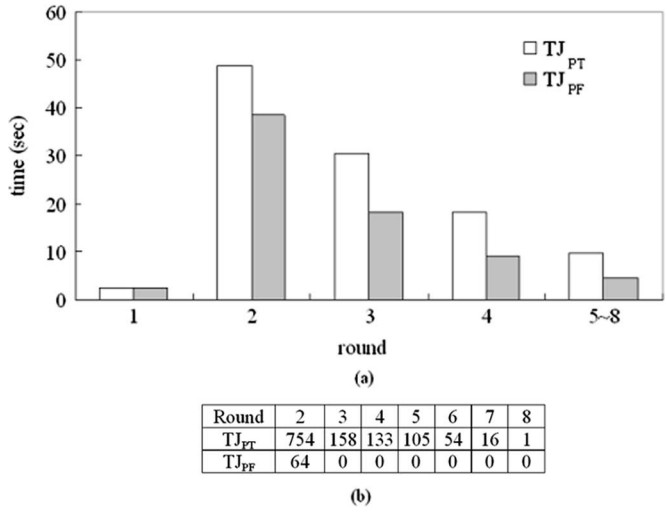


Fig. 23. Ratio of execution time which is incurred by comparing the path.



Round	2	3	4	5	6	7	8
$\mathrm{TJ_{PT}}$	754	158	133	105	54	16	1
$\mathrm{TJ_{PF}}$	64	0	0	0	0	0	0

(b)

Fig. 24. Performance comparison between $\mathrm{TJ_{PT}}$ and $\mathrm{TJ_{PF}}$ in each round. (a) Execution time and (b) the number of paths stored.

join the L-transactions in S_1 for generating R_2 to be stored in the transaction component of a mobile sequence tree. In round two, $\mathrm{TJ_{PT}}$ constructs the path component of the mobile sequential tree for storing C_2. In the following rounds, when $\mathrm{TJ_{PT}}$ stores the path information of C_k, $k \geq 3$, $\mathrm{TJ_{PT}}$ still needs to construct the path component of the mobile sequential tree for storing C_k. However, by utilizing the pattern family relationship, $\mathrm{TJ_{PF}}$ can use the shared-path tree generated in C_2 for indexing the path information of C_k, $k \geq 3$, in the following rounds, leading to more efficient execution. In addition, to provide more insights into $\mathrm{TJ_{PF}}$ and $\mathrm{TJ_{PT}}$, the numbers of branches for storing the path information of C_k are shown in Fig. 24(b). In $\mathrm{TJ_{PT}}$, these branches are stored in the mobile sequential trees for all rounds. In $\mathrm{TJ_{PF}}$, these branches are stored in the shared-path tree generated in round two, and thus, the amount of memory savings is 25.8 MB.

V. Conclusion

In this paper, we explored a data mining capability which involves mining mobile sequential patterns for an MC environment. In essence, the mining of mobile sequential patterns aggregates the concepts of mining association rules (mining path traversal patterns and mining sequential patterns) and thus

requires a combined use of corresponding techniques. By having different priorities on the factors involving large itemsets, traversal paths, and orders of purchases, we have devised three algorithms (algorithm $\mathrm{TJ_{LS}}$, algorithm $\mathrm{TJ_{PT}}$, and algorithm $\mathrm{TJ_{PF}}$) for determining large sequential patterns from mobile transaction sequences. $\mathrm{TJ_{LS}}$ is devised in light of the concept of association rules, and $\mathrm{TJ_{PT}}$ is devised by taking both the concepts of association rules and path traversal patterns into consideration. By utilizing the pattern family technique, $\mathrm{TJ_{PF}}$ is able to generate large sequential patterns very efficiently. A simulation model for the MC environment was developed, and a synthetic workload was generated for performance studies. In our performance study, the proposed algorithm $\mathrm{TJ_{PF}}$ significantly outperforms others in both execution efficiency and memory saving, indicating the usefulness of the pattern family technique. It is shown by our results that by taking both moving patterns and purchase patterns into consideration, one can have a better model for an MC system and thus be able to exploit the intrinsic relationship between these two important customer behaviors for the efficient mining of mobile sequential patterns.

REFERENCES

[1] [Online]. Available: http://www.mobilecommerceworld.com

[2] Wallet Application [Online]. Available: http://www.forum.nokia.com.

[3] R. Agarwal, C. Aggarwal, and V. V. V. Prasad, "A tree projection algorithm for generation of frequent itemsets," *J. Parallel Distrib. Comput*., 2000.

[4] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu, "The intelligent recommendation analyzer," in *Proc. ICDCS Int. Workshop of Knowledge Discovery and Data Mining in the World-Wide Web*, Taipei, Taiwan, Apr. 10–13, 2000, pp. F67–F72.

[5] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, May 1993, pp. 207–216.

[6] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th VLDB Conf.*, Sep. 1994, pp. 478–499.

[7] ——, "Mining sequential patterns," in *Proc. 11th Int. Conf. Data Engineering*, Mar. 1995, pp. 3–14.

[8] J. M. Ale and G. H. Rossi, "An approach to discovering temporal association rules," in *Proc. ACM Symp. Applied Computing*, Como, Italy, Mar. 2000, pp. 294–300.

[9] A. M. Ayad, N. M. El-Makky, and Y. Taha, "Incremental mining of constrainted association rules," in *Proc. 1st SIAM Conf. Data Mining,* Chicago, IL, Apr. 2001.

[10] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal, "Mining frequent patterns with counting inference," *ACM SIGKDD Explor. Newslett*., vol. 2, no. 2, pp. 66–75, Dec. 2000.

[11] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: An overview from a database perspective," *IEEE Trans. Knowl. Data Eng.* vol. 8, no. 6, pp. 866–833, Dec. 1996.

[12] M.-S. Chen, J.-S. Park, and P. S. Yu, "Efficient data mining for path traversal patterns," *IEEE Trans. Knowl. Data Eng*., vol. 10, no. 2, pp. 209–221, Apr. 1998.

[13] X. Chen and I. Petrounias, "Discovering temporal association rules: Algorithms, language and system," in *Proc. 16th Int. Conf. Data Eng*., San Diego, CA, Feb. 28–Mar. 3, 2000, p. 306.

[14] D. Cheung, J. Han, V. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating techniques," in *Proc. Int. Conf. Data Eng*., Feb. 1996, pp. 106–114.

[15] F. Coenen, G. Goulbourne, and P. Leng, "Computing association rules using partial totals," in *Proc. 5th Eur. Conf. Principles of Data Mining and Knowledge Discovery*, Freiburg, Germany, Sep. 3–5, 2001, pp. 54–66.

[16] R. Cooley, B. Mobasher, and J. Srivastava, "Data preparation for mining world wide web browsing patterns," *J. Knowl. Inf. Syst*., vol. 1, no. 1, 1999.

[17] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth, "Rule discovery from time series," *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD-98)*, Aug. 1998.

[18] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurasamy, *Advances in Knowledge Discovery and Data Mining*. Cambridge, MA: MIT Press, 1996.

[19] R. Floyd, B. Housel, and C. Tait, "Mobile web access using enetwork web express," *IEEE Pers. Commun*., vol. 5, no. 5, pp. 47–52, Oct. 1998.

[20] Bluetooth Overview, 1999 [Online]. Available: http://www.bluetooth. com

[21] WAP Forum Wireless Application Protocol. [Online]. Available: http://www.wapforum. org/

[22] J. Han, G. Dong, and Y. Yin, "Efficient mining of partial periodic patterns in time series database," in *Proc. 15th Int. Conf. Data Engineering*, Mar. 1999.

[23] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *Proc. 21th VLDB Conf.*, Sep. 1995, pp. 420–431.

[24] J. Han, W. Gong, and Y. Yin, "Mining segment-wise periodic patterns in time-related databases," in *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD-98)*, Aug. 1998, pp. 214–218.

[25] J. Han, L. V. S. Lakshmanan, and R. T. Ng, "Constraint-based, multidimensional data mining," *IEEE Comput*., vol. 32, no. 8, pp. 46–50, Aug. 1999.

[26] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "FreeSpan: Frequent pattern-projected sequential pattern mining," in *Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining*, Aug. 2000, pp. 355–359.

[27] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. ACM SIGMOD Conf.*, 2000.

[28] L. V. S. Lakshmanan, R. Ng, J. Han, and A. Pang, "Optimization of constrained frequent set queries with 2-variable constraints," in *Proc. ACM-SIGMOD Conf. Management of Data*, Jun. 1999, pp. 157–168.

[29] C.-H. Lee, C.-R. Lin, and M.-S. Chen, "On mining general temporal association rules in a publication database," in *Proc. 1st IEEE Int. Conf. Data Mining (ICDM 2001)*, Nov./Dec. 2001.

[30] ——, "Sliding-window filtering: An efficient algorithm for incremental mining," in *Proc. ACM 10th Int. Conf. Inf. and Knowledge Management*, Nov. 2001.

[31] C.-R. Lin, C.-H. Yun, and M.-S. Chen, "Using slice scan and selective hash for episode mining," in *Proc. Workshop on Temporal Data Mining (SIGKDD 2001)*, Aug. 2001.

[32] J.-L. Lin and M. H. Dunham, "Mining association rules: Anti-skew algorithms," in *Proc. Int. Conf. Data Engineering*, 1998, pp. 486–493.

[33] M.-Y. Lin and S.-Y. Lee, "Incremental update on sequential patterns in large databases," in *Proc. 10th IEEE Int. Conf. Tools with Artificial Intelligence*, Nov. 1998, pp. 24–31.

[34] Y.-B. Lin, "Modeling techniques for PCS networks," *IEEE Commun. Mag*., vol. 35, no. 2, pp. 102–107, Feb. 1997.

[35] B. Liu, W. Hsu, and Y. Ma, "Mining association rules with multiple minimum supports," in *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Aug. 1999.

[36] Ky. S. M. N. Garofalakis and R. Rastogi, "SPIRIT: Sequential pattern mining with regular expression constraints," *VLDB J*., 1999.

[37] H. Mannila and C. Meek, "Global partial orders from sequential data," in *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Boston, MA, 2000, pp. 161–168.

[38] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining Knowl. Discov.*, vol. 1, no. 3, pp. 259–289, 1997.

[39] F. Masseglia, F. Cathala, and P. Poncelet, "The PSP approach for mining sequential patterns," in *Proc. 2nd Eur. Conf. Principles of Data Mining and Knowledge Discovery*, Nantes, France, Sep. 1998, vol. 1510, pp. 176–184.

[40] E. Modiano and A. Ephremides, "Efficient algorithms for performing packet broadcasts in a mesh network," *IEEE/ACM Trans. Networking*, vol. 4, no. 4, pp. 639–648, Aug. 1996.

[41] R. Muntz, W. Wang, and J. Yang, "TAR: Temporal association rules on evolving numerical attributes," in *Proc. 17th Int. Conf. Data Engineering*, 2001.

[42] B. Padmanabhan and A. Tuzhilin, "Pattern discovery in temporal databases: A temporal logic approach," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, 1996, pp. 35 –354.

[43] J.-S. Park, M.-S. Chen, and P. S. Yu, "An effective hash based algorithm for mining association rules," in *Proc. ACM SIGMOD Conf.*, May 1995, pp. 175–186.

[44] S. Parthasarathy, M. J. Zaki, M. Ogihara, and S. Dwarkadas, "Incremental and interactive sequence mining," in *Proc. ACM CIKM Int. Conf. Inf. and Knowledge Management*, 1999, pp. 251–258.

[45] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *Proc. 7th Int. Conf. Database Theory*, Jan. 1999.
[46] J. Pei and J. Han, "Can we push more constraints into frequent pattern mining?" in *Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining*, Aug. 2000.
[47] J. Pei, J. Han, and R. Mao, "CLOSET: An efficient algorithm for mining frequent closed itemsets," in *Proc. ACM-SIGMOD Worksho on Res. Issues in Data Mining and Knowledge Discovery*, 2000, pp. 21–30.
[48] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proc. 17th Int. Conf. Data Eng.*, Apr. 2001.
[49] W.-C. Peng and M.-S. Chen, "Developing data allocation schemes by incremental mining of user moving patterns in a mobile computing system," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 1, pp. 70–85, Feb. 2003.
[50] P. Pirolli and J. E. Pitkow, "Distributions of Surfers' paths through the world wide web: Empirical characterization," *World Wide Web*, vol. 2, no. 1–2, pp. 29–45, 1999.
[51] J. E. Pitkow and P. Pirolli, "Mining longest repeated subsequences to predict world wide web surfing," in *Proc. 2nd USENIX Symp. Internet Technologies and Syst.*, Oct. 1999.
[52] J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in E-commerce," in *Proc. 1st ACM Conf. Electronic Commerce*, Denver, CO, Nov. 1999, pp. 158–166.
[53] R. Srikant and R. Agrawal, "Mining generalized association rules," in *Proc. 21th VLDB Conf.*, Sep. 1995, pp. 407–419.
[54] R. Agrawal and R. Srikant, "Mining quantitative association rules in large relational tables," in *Proc. 1996 ACM-SIGMOD Conf. Management of Data*, 1996, pp. 1–12.
[55] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96)*, Mar. 1996, pp. 201–212.
[56] A. K. H. Tung, H. Lu, J. Han, and L. Feng, "Breaking the barrier of transactions: Mining intertransaction association rules," in *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Aug. 1999, pp. 297–301.
[57] U. Varshney, R. J. Vetter, and R. Kalakota, "Mobile commerce: A new frontier," *IEEE Comput.*, vol. 33, no. 10, pp. 32–38, Oct. 2000.
[58] J. Veijalainene, "Transactions in mobile electronic commerce," in *Proc. 8th Int. Workshop on Foundations of Models and Languages for Data and Objects*, Sep. 1999, pp. 203–224.
[59] R. Villafane, K. A. Hua, D. Tran, and B. Maulik, "Knowledge discovery from series of interval events," *J. Intell. Inf. Syst.*, vol. 15, no. 1, pp. 71–89, 2000.
[60] K. Wang, Y. He, and J. Han, "Mining frequent itemsets using support constraints," in *Proc. 26th VLDB Conf.*, Sep. 2000, pp. 43–52.
[61] K. Wang, S. Q. Zhou, and S. C. Liew, "Building hierarchical classifiers using class proximity," in *Proc. 25th VLDB Conf.*, 1999, pp. 363–374.
[62] C. Yang, U. Fayyad, and P. Bradley, "Efficient discovery of error-tolerant frequent itemsets in high dimensions," in *Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, 2001, pp. 194–203.
[63] J. Yang, W. Wang, and P. S. Yu, "Mining asynchronous periodic patterns in time series data," in *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2000, pp. 275–279.
[64] C.-H. Yun and M.-S. Chen, "Using pattern-join and purchase-combination for mining web transaction patterns in an electronic commerce environment," in *Proc. 24th IEEE Annu. Int. Computer Software and Application Conf.*, Oct. 2000, pp. 99–104.
[65] M. J. Zaki, "Sequence mining in categorical domains: Incorporating constraints," in *Proc. 2000 ACM CIKM Int. Conf. Inf. and Knowledge Management*, Nov. 2000.
[66] ——, "SPADE: An efficient algorithm for mining frequent sequences," *Mach. Learn.*, vol. 42, no. 1/2, pp. 31–60, 2001.
[67] M. J. Zaki and C. Hsiao, "CHARM: An efficient algorithm for closed itemset mining," in *Proc. 2nd SIAM Conf. Data Mining*, Apr. 2002. [Online]. Available: http://www.siam.org/meetings/sdm02/proceedings/sdm02-27.pdf
[68] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," in *Proc. 3rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Aug. 1997, pp. 283–286.
[69] Q. Zheng, K. Xu, S. Ma, and W. Lv, "The algorithms of updating sequential patterns," in *Proc. 5th Int. Workshop on High Performance Data Mining in conjunction with 2nd SIAM Conf. Data Mining*, 2002. [Online]. Available: http://arxiv.org/abs/cs.DB/0203027

Ching-Huang Yun received the B.S. and Ph.D. degrees in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1997 and 2005, respectively.

He is currently an R&D manager with IBTek Inc., Taipei. His research interests include data mining, web technologies, and video surveillance systems.

Ming-Syan Chen (S'88–M'88–SM'93–F'04) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., and the M.S. and Ph.D. degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, in 1985 and 1988, respectively.

He is currently a Professor and the Chairman of the Graduate Institute of Communication Engineering and a Professor with the Electrical Engineering Department and the Computer Science and Information Engineering Department, National Taiwan University. He was a Research Staff Member at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 200 papers in his research areas. He holds, or has applied for, 18 U.S. patents and seven R.O.C. patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He is currently on the editorial board of the *Very Large Data Base (VLDB) Journal*, the *Knowledge and Information Systems (KAIS) Journal*, the *Journal of Information Science and Engineering*, and the *International Journal of Electrical Engineering*.

Dr. Chen served as an Associate Editor of IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING from 1997 to 2001. He was a Distinguished Visitor of the IEEE Computer Society for Asia-Pacific from 1998 to 2000. He served as the Program Chair of Pacific Area Knowledge Discovery and Data Mining, International Vice Chair of INFOCOM 2005, and Program Vice-Chair of IEEE ICDCS 2005, ICPP 2003, and VLDB-2002. 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 at DASFAA-1999, and on parallel databases at the 11th IEEE ICDE in 1995. He was also a Guest Coeditor for IEEE TKDE on a special issue on data mining in December 1996. He is a recipient of the National Science Council Distinguished Research Award and K.-T. Li Research Penetration Award for his research work. He also received the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product and received numerous awards for his research, teaching, inventions, and patent applications. He is a Member of the ACM.