Knowl Inf Syst (2013) 37:363–387
DOI 10.1007/s10115-012-0483-z

REGULAR PAPER

Efficient algorithms for discovering high utility user behavior patterns in mobile commerce environments

Bai-En Shie · Hui-Fang Hsiao · Vincent S. Tseng

Received: 31 March 2011 / Revised: 2 December 2011 / Accepted: 4 March 2012 /
Published online: 23 March 2012

Abstract Mining user behavior patterns in mobile environments is an emerging topic in data mining fields with wide applications. By integrating moving paths with purchasing transactions, one can find the sequential purchasing patterns with the moving paths, which are called *mobile sequential patterns* of the mobile users. Mobile sequential patterns can be applied not only for planning mobile commerce environments but also for analyzing and managing online shopping websites. However, unit profits and purchased numbers of the items are not considered in traditional framework of mobile sequential pattern mining. Thus, the patterns with high utility (i.e., profit here) cannot be found. In view of this, we aim at integrating mobile data mining with utility mining for finding *high-utility mobile sequential patterns* in this study. Two types of algorithms, namely level-wise and tree-based methods, are proposed for mining high-utility mobile sequential patterns. A series of analyses and comparisons on the performance of the two different types of algorithms are conducted through experimental evaluations. The results show that the proposed algorithms outperform the state-of-the-art mobile sequential pattern algorithms and that the tree-based algorithms deliver better performance than the level-wise ones under various conditions.

Keywords Utility mining · Mobility pattern mining · Mobile environments · High-utility mobile sequential pattern

B.-E. Shie · H.-F. Hsiao · V. S. Tseng (✉)
Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC
e-mail: tsengsm@mail.ncku.edu.tw
URL: http://idb.csie.ncku.edu.tw/tsengsm

B.-E. Shie
e-mail: brianshie@gmail.com

H.-F. Hsiao
e-mail: karolter1130@gmail.com

1 Introduction

Data mining refers to the process of discovering potentially useful information from large databases. In behavior informatics [5], previous studies have discovered many kinds of user behavior patterns for different applications, such as cross marketing in business domains [1–3,19,24], websites design and management [7,18,25], and mobile environments planning [9–11,14–16,20,21,26]. Among these issues, mining user behavior patterns in mobile environments plays an emerging role in the past decade since mobile devices and wireless applications have become one of the most popular communication media in the world. With a series of users' moving logs recorded by the mobile devices with GPS services, we can acquire the moving paths of mobile users. Combining moving logs and payment records, mobile transaction sequences that are the sequences of moving paths with purchased transactions can be generated. There exists useful information recorded in mobile transaction sequences. Yun et al. [26] first proposed a framework combining moving paths and sequential patterns to find mobile sequential patterns, i.e., the sequential patterns with their moving paths, in mobile transaction sequences. For example, a mobile sequential pattern <{<A; clothes> <C; lipsticks>}; ABC> means the customers often move through the path <ABC> and bought clothes and lipsticks in A and C, respectively. If the shopkeepers acquire this pattern, they can prepare promotions of lipsticks when they meet the customers who had bought clothes in A to raise the customers' desires to purchase.

However, mobile sequential patterns cannot reflect the actual profit of items in the databases. Valuable user behavior patterns, such as the patterns with purchasing diamond rings, may not be discovered since their frequencies are not enough. Utility mining [4,6,12,13,19,22–24] is proposed for solving this problem in the traditional transaction databases. Given pre-defined utility values, which may be the importance, interestingness or profits of the items, utility mining is to find the high-utility patterns, which are the patterns with high-utility values, from the databases. From this viewpoint, we can realize that pushing utility mining into the framework of mobility pattern mining is an essential topic. If decision makers know which patterns are more valuable, they can choose more proper reactions based on the useful information. For example, assume there exist two user behavior patterns: UBP_1=<{<A; clothes <H; lipsticks >}; ABCDH> and UBP_2=<{<A; clothes> <H; diamond rings>}; AEFGH>. The two patterns show that although these customers moved from A to H, they may buy different items since they passed through different paths. In general, we can know the profits of diamond rings are much higher than lipsticks. If the shopkeepers in H know both the two patterns, they can prepare some promotions or activities for UBP_2 to attract the customers who buy clothes in A and go through the path AEFGH, such as to raise their desires to purchase diamond rings in H.

Although there exists a number of prominent research works about mobility pattern mining and utility mining, there is no work done on combination of the two topics. In view of this, we attempt to integrate mobile sequential pattern mining with utility mining for finding *high-utility mobile sequential patterns* in this paper. Two different types of methods, namely level-wise and tree-based ones, are proposed for this problem. For level-wise method, an algorithm called $UMSP_L$ (*mining high Utility Mobile Sequential Patterns by a Level-wise method*) is proposed. Not only supports but also utilities of patterns are considered in the level-wise mining processes. For tree-based methods, two algorithms $UMSP_{T(DFG)}$ (*mining high Utility Mobile Sequential Patterns by a Tree-based method with a Depth First Generation strategy*) and $UMSP_{T(BFG)}$ (*mining high Utility Mobile Sequential Patterns by a Tree-based method with a Breadth First Generation strategy*) are proposed. Both of the two tree-based algorithms use a tree structure named *MTS-Tree* (*Mobile Transaction*

Sequence Tree) to summarize the corresponding information, such as locations, items, paths and utilities, in mobile transaction databases. The experimental results show that in almost all experiments, tree-based methods have better performance than level-wise ones. Moreover, it is shown that in all experiments, the three proposed methods outperform the compared algorithm which is extended from the state-of-the-art mobile sequential pattern algorithm [26].

Major contributions of this work are described as follows: First, to our best knowledge, this research is the first work that integrates high-utility pattern mining with mobility pattern mining so as to explore the new problem of mining high-utility mobile sequential patterns. Second, different methods extended from two different types of frameworks for frequent pattern mining are proposed for solving this problem. Third, a series of detailed experiments were conducted to evaluate the performance of the proposed methods in different conditions. Through the combination of high-utility patterns and moving paths, highly profitable mobile sequential patterns can be found. High-utility mobile sequential patterns are more crucial in many domains, such as mobile commerce environments, metropolitan planning and online shopping websites which sell a wide selection of merchandise in different web pages. We expect that this useful pattern can deliver novel and insightful information in mobile commerce environments.

The remaining of this paper is organized as follows: Related work is briefly reviewed in Sect. 2. Problem definitions of this work are given in Sect. 3. In Sect. 4, the proposed algorithms are addressed in detail. Experimental evaluation is shown in Sect. 5. The conclusions and future work are given in Sect. 6.

2 Related work

The literature reviews about frequent pattern mining, mobility pattern mining and utility pattern mining are given in this section.

2.1 Frequent pattern mining

Extensive studies have been proposed for finding frequent patterns in transaction databases [2,3,8,17]. Frequent itemset mining [2,8] is the most popular topic among them. Apriori [1] is the pioneer for mining frequent itemsets from transaction databases by a level-wise candidate generation-and-test method. Tree-based frequent itemset mining algorithms such as FP-Growth [8] were proposed afterward. FP-Growth improves the efficiency of frequent itemset mining since it does not have to generate candidate itemsets during the mining process and it only scans the database twice. Afterwards, sequential pattern mining [3,17] is proposed for finding customer behaviors in transaction databases. As an extension method of Apriori, AprioriAll [3] also uses a level-wise framework to find sequential patterns. On the contrary, PrefixSpan [17] finds sequential patterns directly from projected databases without generating any candidate pattern. Thus, the performance can be more improved.

2.2 Mobility pattern mining

Mining user behaviors in mobile environments [10,11,14,20,21,26] is an emerging topic in the frequent pattern mining field. SMAP-Mine [21] was first proposed for finding customers' mobile access patterns. However, in different time periods, users' popular services may be totally different. Thus, T-Map algorithm [11] was proposed to find temporal mobile access patterns in different time intervals. Although users' mobile access patterns are important, their moving paths are also essential. Therefore, Yun et al. [26] proposed a framework which combines moving paths and sequential patterns to find mobile sequential patterns. By the above researches, although we can know the patterns in different time intervals, defining time intervals is not an easy work yet. In view of this, Tseng et al. [20] proposed TMSP-Mine algorithm to automatically find proper time intervals of mobile sequential patterns based on the genetic algorithm. On the other hand, the character of customers is also important in the mobile environment. Different groups of customers may bring different patterns. Thus, Lu et al. [14] proposed a framework to find the cluster-based mobile sequential patterns. The customers whose moving paths and transactions are similar will be clustered into the same cluster. The found patterns may be closer to real customer behaviors by this method.

2.3 Utility mining

The profits of items are not considered in the above researches. In frequent pattern mining fields, a new topic raised for conquering this problem, that is, utility mining [4,6,12,13,19,22–24]. In utility mining, each item may have different profits. Chan et al. first proposed the problem of utility mining [6]. Yao et al. proposed UMining algorithm [23] by applying an estimation method to prune the search space. However, it cannot capture the complete set of high-utility itemsets since some high-utility patterns may be pruned during the mining process. There are some researches that integrate other topics with utility mining, such as streaming environments [19]. In [19], Shie et al. addressed the problem of finding temporal maximal utility patterns by the TMUI-Tree.

Among these researches, Liu et al. [13] proposed Two-Phase algorithm which uses the transaction-weighted downward closure property to maintain the downward closure property in utility mining. Although Two-Phase algorithm can reduce the search space of utility mining, it still generates too many candidates. Thus, Li et al. [12] proposed an isolated items discarding strategy to reduce the number of candidates by pruning isolated items during the level-wise searches.

Some researches employed other frameworks such as tree-based framework [4,22,24] to improve the performance of utility mining. In [24], Yen et al. addressed this problem by AM algorithm which utilizes the AM-Tree to find high-utility itemsets without an extra scan of database for finding real high-utility itemsets. Ahmed et al. [4] proposed a structure named IHUP-Tree which maintains essential information about utility mining. It avoids scanning database multiple times and generating candidates in the mining process. Although IHUP-Tree achieves a better performance than Two-Phase, it still produces too many HTWUIs. Therefore, Tseng et al. proposed a novel algorithm named UP-Growth [22], which applies several strategies during the mining processes. By the proposed strategies, the estimated utilities are effectively decreased in the proposed tree structure named UP-Tree in the mining processes and the number of HTWUIs is further reduced. Therefore, the performance of utility mining can be improved significantly.

By the above literature reviews, although there are many researches about mobility pattern mining and utility mining, there is no research on the combination of the two topics. This paper is the first work which integrates the two topics to find high-utility patterns with frequent moving paths in mobile environments.

Table 1 Notations used in this paper

Symbol	Semantics
l_{loc}	Location*loc*
i_v	Item v
T_j	Transaction j
q_{jp}	Purchased quantity of item i_{jp} in transaction T_j
$w(i_{jp})$	Unit profit of item i_{jp}
$u(Z, S_j)$	Utility of pattern Z in mobile transaction sequence S_j
$u(Z)$	Utility of pattern Z in mobile transaction sequence database
$sup(Z)$	Support of pattern Z in mobile transaction sequence database
$SU(\mathrm{S}_j)$	Sequence utility of sequence S_j
$SWU(Z)$	Sequence weighted utilization of pattern Z
δ	Minimum support threshold
ε	Minimum utility threshold

Table 2 Abbreviations used in this paper

Abbreviation	Semantics
UMSP	High-utility mobile sequential pattern
SWU	Sequence weighted utilization
WULI	High sequence weighted utilization loc-itemset
WULP	High sequence weighted utilization loc-pattern
WUMSP	High sequence weighted utilization mobile sequential pattern
SWDC	Sequence weighted downward closure property

3 Preliminaries and definitions

In this section, we first define basic notations for mining high-utility mobile sequential patterns in mobile environments in detail and then address the problem statement of this work.

Let $L = \{l_1, l_2, \ldots, l_p\}$ be a set of *locations* in the mobile commerce environment and $I = \{i_1, i_2, \ldots, i_g\}$ be a set of *items* sold in the locations. An *itemset* is denoted as $\{i_1, i_2, \ldots, i_k\}$, where each item $i_v \in I$, $1 \leq v \leq k$ and $1 \leq k \leq g$. Given a *mobile transaction sequence database D, a mobile transaction sequence* S = <$T_1, T_2, \ldots, T_n$> is a set of transactions ordered by time, where a *transaction* $T_j = (l_j; \{[i_{j1}, q_{j1}], [i_{j2}, q_{j2}], \ldots, [i_{jh}, q_{jh}]\})$ represents that a user made T_j in l_j, where $1 \leq j \leq n$. In T_j, the *purchased quantity* of item i_{jp} is q_{jp}, where $1 \leq p \leq h$. A *path* is denoted as $l_1 l_2 \ldots l_r$, where $l_j \subseteq L$ and $1 \leq j \leq r$. Tables 1 and 2 summarize the notations and abbreviations used throughout the paper with brief descriptions of semantics.

Definition 1 (*Utility of a loc-item in a mobile transaction sequence*) A *loc-item*, denoted as <$l_{loc}; i_{je}$>, stands for the item i_{je} that happened in the location l_{loc}, where $l_{loc} \in L$ and $i_{je} \subseteq I$. The*utility* of a loc-item <$l_{loc}; i_{je}$> in a mobile transaction sequence S_j is denoted as u(< $l_{loc}; i_{je}$>, S_j), that is defined as $q_{je} \times w(i_{je})$, where $w(i_{je})$ is the *unit profit* of item i_{je}, which is recorded in a *utility table*.

Table 3 Mobile transaction sequence database *DB*

SID	Mobile transaction sequence	SU
S_1	<(A; {[i_1, 2]}), (B; null), (*C*; {[i_2, 1]}), (D; {[i_4, 1]}), (E; null), (F; {[i_5, 2]})>	54
S_2	<(A; {[i_1, 3]}), (B; null), (*C*; {[i_2, 2], [i_3, 5]}), (K; null), (*E*; {[i_6, 10]}), (*F*; {[i_5, 4]}), (*G*; {[i_8, 2]}), (L; null), (*H*; {[i_7, 2]})>	132
S_3	<(A; {[i_1, 3]}), (B; null), (*C*; {[i_2, 1], [i_3, 5]}), (D; {[i_4, 2]}), (E; null), (*F*; {[i_5, 1], [i_6, 2]}), (G; null), (H; {[i_7, 1]})>	72
S_4	<(A; {[i_1, 1]}), (W; null), (*C*; {[i_3, 10]}), (E; null), (F; {[i_5, 1]}), (G; {[i_8, 2]}),(L; null), (H; {[i_7, 1]}), (E; {[i_9, 1]})>	59
S_5	<(A; {[i_1, 4]}), (B; null), (*C*; {[i_3, 10]}), (D; {[i_4, 1]}), (E; null), (F; {[i_5, 1]}), (G; null), (H; {[i_7, 2]})>	73
S_6	<(*C*; {[i_2, 2]}), (D; null), (E; {[i_9, 1]}), (F; {[i_5, 1]})>	31

Take the mobile transaction sequence database *DB* in Table 3 and the utility table in Table 4, for example, $(u < \text{A}; i_1 >, \text{S}_1) = 2 \times 1 = 2$.

Definition 2 (*Utility of a loc-itemset in a mobile transaction sequence database*) A *loc-itemset*, denoted as $<l_{loc}; \{i_1, i_2, \ldots, i_g\}>$, stands for the itemset $\{i_1, i_2, \ldots, i_g\}$ that happened in l_{loc}, where $l_{loc} \in L$ and $\{i_1, i_2, \ldots, i_g\} \subseteq I$. The utility of a loc-itemset $<l_{loc}; \{i_{j1}, i_{j2}, \ldots, i_{jg}\}>$ in a mobile transaction sequence S_j is denoted as $u(<l_{loc}; \{i_{j1}, i_{j2}, \ldots, i_{jg}\}>, S_j)$ and defined as $\sum_{k=1}^{g} u(< l_{loc}; i_{jk} >, S_j)$. Moreover, the utility of a loc-itemset $Y =<l_{loc}; \{i_{j1}, i_{j2}, \ldots, i_{jg}\}>$ in a mobile transaction sequence database D is denoted as $u(Y)$ or $u(<l_{loc}; \{i_{j1}, i_{j2}, \ldots, i_{jg}\}>)$ and defined as $\sum_{(Y \subseteq S_j) \wedge (S_j \in D)} u(Y, S_j)$.

For the example in Table 3, the utility of the loc-itemset <C; {i_2, i_3}> in S_2 is calculated as u(<C;{i_2, i_3}>, S_2)=u(<C; i_2>, S_2)+ u(<C; i_3 >, S_2) $= 2 \times 5 + 5 \times 3 = 25$. The utility of <C; {$i_2$, i_3}> in the database *DB* is calculated as u(<C;{i_2, i_3}>)=u(<C; {i_2, i_3}>, S_1) + u(<C; {i_2, i_3}>, S_4) = 25 + 20 = 45.

Definition 3 (*Utility of a loc-pattern in a mobile transaction sequence database*) A *loc-pattern X*, which is denoted as $<l_1; \{i_{11}, i_{12}, \ldots, i_{1g}\}> <l_2; \{i_{21}, i_{22}, \ldots, i_{2g}\}>\ldots<l_m; \{i_{m1}, i_{m2}, \ldots, i_{mg}\}>$, is a list of loc-itemsets. The utility of a loc-pattern X in S_j is denoted as $u(X, \text{S}_j)$ and defined as $\sum_{\forall Y \in X} u(Y, S_j)$. The utility of a loc-pattern X in D is denoted as $u(X)$ and defined as $\sum_{(X \subseteq S_j) \wedge (S_j \in D)} u(X, S_j)$.

For the example, in Table 3, the utility of the loc-pattern <A; i_1> <C; {i_2, i_3}> in S_2 is calculated as u(<A; i_1><C;{i_2, i_3}>, S_2)= u(<A; i_1>, S_2)+ u(<C; {i_2, i_3}>, S_2) = 3 + 25 = 28. The utility of <A; i_1> <C; {i_2, i_3}> in *DB* is calculated as u(<A; i_1><C; {i_2, i_3}>) = u(<A; i_1><C; {i_2, i_3}>, S_2)+ u(<A; i_1><C; {i_2, i_3}>, S_3) = 28 + 23 = 51.

Definition 4 (*Support and utility of a moving pattern*) A*moving pattern* is composed of a loc-pattern and a path. It is recorded by the form as $<\{<l_1; \{i_{11}, i_{12}, \ldots, i_{1g}\}> <l_2; \{i_{21}, i_{22}, \ldots, i_{2g}\} > \ldots < l_m; \{i_{m1}, i_{m2}, ldots, i_{mg}\}>\}; l_1 l_2 \ldots l_m>$. The*support* of a moving pattern P, denoted as *sup*(P), is defined as the number of mobile transaction sequences which contains P in D. On the other hand, the utility of a moving pattern P, denoted as $u(P)$, is defined as the summation of utilities of the loc-patterns of P in the mobile transaction sequences which contain the path of P in D.

Definition 5 (*High-utility mobile sequential pattern*) Given a *minimum support threshold* δ and a *minimum utility threshold* ε, a moving pattern P is called a *mobile sequential pattern* if sup $(P) \geq \delta$. Further, P is called a *high-utility mobile sequential pattern*, which is abbreviated as *UMSP*, if $\sup(P) \geq \delta$ and $u(P) \geq \varepsilon$. Moreover, the length of a pattern is the number of loc-itemsets in this pattern. A pattern with length k is denoted as k-pattern.

For the example in Table 3, the support of the moving pattern <{<A; i_1><C; {i_2, i_3}>}; ABC>, which is denoted as *sup*(<{<A; i_1><C; {i_2, i_3}>}; ABC>), is 2. The utility of <{<A; i_1><C; {i_2, i_3}>}; ABC > in *DB* is calculated as *u*(<{<A; i_1><C; {i_2, i_3}>}; ABC>)=*u*(<A; i_1><C; {i_2, i_3}>, S_2) + *u*(<A; i_1><C; {i_2, i_3}>, S_3) = 51. If minimum support threshold δ is 2 and minimum utility threshold ε is 50, the moving pattern <{<A; i_1> <C; {i_2, i_3}>}; ABC > is a 2-high utility mobile sequential pattern (2-UMSP).

After addressing the problem definition of mining high-utility mobile sequential patterns in mobile environments, we introduce the *sequence weighted utilization* and *sequence weighted downward closure* property in mobile transaction sequence databases, which are extended from [13].

Definition 6 (*Sequence utility of a mobile transaction sequence*) The*sequence utility* of mobile transaction sequence S_j, which is the sum of the utilities of all items in S_j, is defined as $SU(S_j) = \sum_{<l_{loc};i_{je}>\subseteq S_j} u(< l_{loc}; i_{je} >, S_j)$.

For the example, in Table 3, the sequence utility of the mobile transaction sequence S_6 is computed as $SU(S_6) = u(< C; i_2 >, S_6) + u(< E; i_9 >, S_6) + u(< F; i_5 >, S_6) = 10 + 3 + 18 = 31$.

Definition 7 (*Sequence weighted utilization of patterns*) The *sequence weighted utilization*, abbreviated as SWU, of a loc-itemset Y is defined as $(SWUY) = \sum_{(Y\subseteq S_j)\wedge(S_j\in D)} SU(S_j)$. Moreover, the sequence weighted utilization of a loc-pattern X is defined as $SWU(X) = \sum_{(X\subseteq S_j)\wedge(S_j\in D)} SU(S_j)$. In addition, the sequence weighted utilization of a moving pattern P is defined as the summation of SWUs of the loc-patterns of P in the mobile transaction sequences which contain the path of$P$ in D.

Definition 8 (*High sequence weighted utilization pattern*) A pattern Z is called a *high sequence weighted utilization pattern*, abbreviated as WUP, if $sup(Z) \geq \delta$ and $SWU(Z) \geq \varepsilon$. Similarly, in the following paragraphs, *high sequence weighted utilization loc-itemset*, *high sequence weighted utilization loc-pattern* and *high sequence weighted utilization mobile sequential pattern* are abbreviated as WULI, WULP and WUMSP, respectively.

For the example, in Table 3, the sequence weighted utilization of the loc-itemset <D; i_4> is computed as (*SWU*<D; i_4>) = $SU(S_1) + SU(S_3) + SU(S_5) = 54 + 72 + 73 = 199$. The sequence weighted utilization of the loc-pattern <A; i_1><C; {i_2, i_3} > is computed as (SWU<A; i_1><C; {i_2, i_3}>) = $SU(S_1) + SU(S_4) = 69 + 60 = 129$. The sequence weighted utilization of the moving pattern <{<A; i_1> <C; {i_2, i_3}>}; ABC> is computed as SWU(<{<A; i_1><C;{i_2, i_3}>}; ABC>)=SWU(<A; i_1><C; {i_2, i_3} >, S_2) + SWU(<A; i_1> <C; {i_2, i_3}>, S_3) = $132 + 72 = 204$. If $\delta = 2$ and $\varepsilon = 50$, the loc-itemset <D; i_4> is a 1-WULI, the loc-pattern <A; i_1><C; {i_2, i_3}> is a 2-WULP, and the moving pattern <{<A; i_1><C; {i_2, i_3}>}; ABC> is a 2-WUMSP.

Theorem 1 (*Sequence weighted downward closure property, SWDC*). *For any pattern P, if P is not a WUP, any superset of P is not a WUP.*

Proof By the definition about SWU, for a pattern *P, SWU(P)* is larger than or equal to $SWU(P')$, where P' is a superset of *P*. If *SWU(P)* is less than ε, $SWU(P')$ is also less than ε.

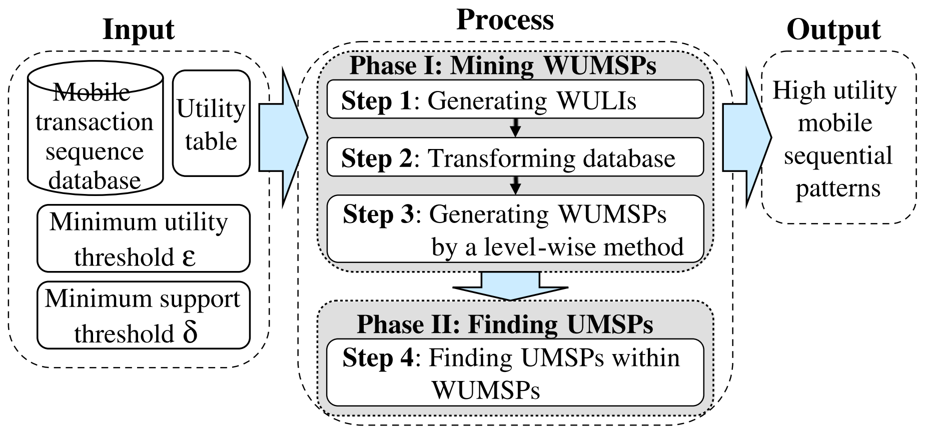

Input
~~Mobile~~ transaction sequence database
Utility table
Minimum utility threshold ε
Minimum support threshold δ
Process
Phase I: Mining WUMSPs
Step 1: Generating WULIs
Step 2: Transforming database
Step 3: Generating WUMSPs by a level-wise method
Phase II: Finding UMSPs
Step 4: Finding UMSPs within WUMSPs
Output
High utility mobile sequential patterns

Fig. 1 Framework of the proposed algorithm UMSP$_L$

Similarly, by the definition about support, $sup(P)$ is larger than or equal to $sup(P')$. If $sup(P)$ is less than δ, $sup(P')$ is also less than δ. By the above two conditions, if $SWU(P) \quad < \varepsilon$ or $sup(P) < \delta$, $SWU(P')$ or $sup(P')$ must be less than ε or δ, respectively.

□

Problem Statement. Given a mobile transaction sequence database D, a pre-defined utility table, a user-specified minimum utility threshold ε and a user-specified minimum support threshold δ, the problem of mining high-utility mobile sequential patterns from D is to discover all high-utility mobile sequential patterns whose supports and utilities are larger than or equal to the two thresholds ε and δ, respectively.

4 Proposed method

In this section, level-wise and tree-based methods are proposed for mining high-utility mobile sequential patterns in mobile commerce environments. General process of this work is described as follows: First, a mobile transaction sequence database, a utility table, a minimum utility threshold and a minimum support threshold are input into the proposed algorithms. After the mining processes, high-utility mobile sequential patterns are generated. In the following, we describe in details the processes of the proposed three algorithms using two different frameworks.

4.1 Level-wise algorithm: UMSP$_L$

The workflow of the proposed method is shown in Fig. 1. The proposed algorithm $UMSP_L$ (*mining high Utility Mobile Sequential Pattern by a Level-wised method*) consists of four steps. The first three steps are to find WUMSPs based on the SWDC property. In step 1, the original database is scanned several times to find all WULIs and each WULI is mapped to a specific identity in a mapping table. Note that the mapped WULIs are 1-WULPs. Then in step 2, the original database is transformed into a trimmed database by mapping the WULIs to their new identities. After this step, the loc-items which are impossible to be the elements of high-utility mobile sequential patterns are removed from the database. Subsequently, the

```
Subroutine: Generate WUMSP (Step 3 of algorithm UMSP_L)
Input: All 1-WULPs, a trimmed database D_T, a minimum support threshold δ
    and a minimum utility threshold ε
Output: WUMSPs

1.  Join the 1-WULPs to form candidate 2-WULPs and then store them into
    2-candidate trees.
2.  Perform an additional scan of D_T to check the supports, SWUs and paths
    of all candidate 2-WULPs, and then generate 2-WULPs. Assume there is
    a candidate 2-WULP X, if sup(X) ≥ δ and SWU(X) ≥ ε, X is a 2-WULP. Generate
    2-WUMSPs by joining the 2-WULPs with their corresponding paths in the
    2-candidate trees.
3.  Generate candidate 3-WULPs by combining the 2-WULPs of two 2-WUMSPs
    if the path of one 2-WUMSP contains another. Store the candidate 3-WULPs
    into 3-candidate trees.
4.  Perform the same process as 2 to find 3-WUMSPs.
5.  Generate candidate k-WULPs, where k > 3, by combining the (k-1)-WULPs
    of the two (k-1)-WUMSPs whose paths are equal to each other. Store
    the generated candidate k-WULPs into k-candidate trees.
6.  Recursively perform the processes 4 and 5 until no candidate WULP is
    generated.
```

Fig. 2 The framework of the proposed algorithm UMSP$_L$

Table 4 Utility table

Item	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9
Utility	1	5	3	11	18	2	5	1	3

Table 5 Mapping table

1-WULP	A;i_1	C;i_2	C;i_3	D;i_4	F;i_5	G;i_8	H;i_7	C;$\{i_2,i_3\}$
After mapping	A;t_1	C;t_2	C;t_3	D;t_4	F;t_5	G;t_6	H;t_7	C;t_8

trimmed database is utilized to find the WUMSPs by the proposed level-wised method in the third step. This step is the key step to the mining performance and its procedure is shown in Fig. 2. Finally in the fourth step, the WUMSPs are checked to find high-utility mobile sequential patterns by an additional scan of the original database.

Next, we use an example to describe the process of the proposed algorithm UMSP$_L$ in detail. Take the mobile transaction sequence database and the utility table in Tables 3 and 4 for example. Assume the minimum support threshold δ is 2 and the minimum utility threshold ε is 100. In the first step, WULIs whose support and SWU are larger than or equal to δ and ε are generated by the processes similar to [13]. In this example, 8 WULIs are generated and mapped into the mapping table as shown in Table 5.

In the second step, using the mapping table, the original database *DB* is mapped into the trimmed database DB_T as shown in Table 6. In DB_T, the original mobile transaction sequences are parsed into the sequences of loc-itemsets and paths. Note that if there is no item in the start or end location of a path, the location will be trimmed. In other words, the paths in DB_T must start and end with loc-itemsets. In Table 6, the last number in a loc-itemset stands for the position of the loc-itemset in the path. Take S$_1$' for example, <F; t_5; 6> means that t_5 has happened in F, and F is in the sixth position of the path.

Table 6 Transformed Database DB_T

SID	Sequence of WULIs	Path
S'_1	<A; t_1; 1> <C;t_2; 3> <D; t_4; 4> <F; t_5; 6>	ABCDEF
S'_2	<A; t_1; 1> <C;t_2, t_3, t_8; 3> <F; t_5; 6> <G; t_6; 7> <H; t_7; 9>	ABCKEFGLH
S'_3	<A; t_1; 1> <C;t_2, t_3, t_8; 3> <D; t_4; 4> <F; t_5; 6> <H; t_7; 8>	ABCDEFGH
S'_4	<A; t_1; 1> <C; t_3; 3> <*F*; t_5; 5> <G; t_6; 7> <H; t_7; 8>	AWCEFGLH
S'_5	<A; t_1; 1> <C; t_3; 3> <D; t_4; 4> <F; t_5; 6> <*H*; t_7; 8>	ABCDEFGH
S'_6	<C; t_2; 1> <F; t_5; 4>	CDEF

In step 3, the candidate 2-WULPs are generated by joining the 1-WULPs in the mapping table. At the same time, they are stored into *k*-candidate trees (*k* is the length of the patterns) which are shown in Fig. 3. Each *k*-candidate tree stores the candidate *k*-WULPs whose last loc-itemsets are the same. When inserting a candidate WULP into a candidate tree, the last loc-itemsets of the WULP will be recorded in the root node of the candidate tree, and the other loc-itemsets are then stored into the tree in their original order sequentially. After constructing 2-candidate trees, an additional scan of DB_T is performed to check the path support and SWU of each candidate 2-WULP and to form the paths in moving patterns. In this example, six 2-WUMSPs whose nodes are marked with solid lines in Fig. 3a are generated.

After generating 2-WUMSPs, candidate 3-WULPs are generated. For two 2-WUMSPs X and Y, they can generate a candidate 3-WULP if the path of X contains that of Y, and vise versa. For example, since the path ABCDEFGH contains CDEFGH, the candidate 3-WULP <{<A; t_1><C; t_3><H;t_7>}> is generated by the two 2-WUMSPs <{<A; t_1><H; t_7 >}; ABCDEFGH> and <{<C;t_3><H;t_7>}; CDEFGH> and it is inserted into the 3-candidate tree in Fig. 3b. After constructing the trees, an additional database scan of DB_T is performed to generate 3-WUMSPs. Finally, candidate *k*-WULPs (where $k \geq 4$) are generated by combining two (*k*-1)-WUMSPs whose paths are equal to each other. For example, since the paths of the two 3-WUMSPs <{<A;t_1><C;t_3><H;t_7>}; ABCDEFGH> and <{<A;t_1><F;t_5><H;t_7>}; ABCDEFGH> are the same, a new candidate 4-WULP <{<A;t_1><C;t_3><F;t_5> <H;t_7>}> is generated and inserted into the 4-candidate tree. The processes will be recursively executed until no candidate moving pattern is generated.

After all WUMSP are generated, an additional scan of database will be performed to check real high-utility mobile sequential patterns in step 4. The WUMSPs whose utilities are larger than or equal to the minimum utility threshold are regarded as high-utility mobile sequential patterns and then output.

4.2 Tree-based algorithm: $\mathrm{UMSP_{T(DFG)}}$

The workflow of the proposed algorithm $\mathrm{UMSP_{T(DFG)}}$ (*mining high Utility Mobile Sequential Pattern by a tree-based method with Depth First Generation strategy*) is shown in Fig. 4. In step 1, WULIs and a mapping table are generated. Then a *MTS-Tree* (*Mobile Transaction Sequence Tree*) is constructed in step 2. In step 3, WUMSPs are generated by mining the MTS-Tree with the depth first generation strategy. Finally, in step 4, UMSPs are generated by checking the actual utility of WUMSPs. In this section, we describe the construction of MTS-tree first and then address the generation of WUMSP.

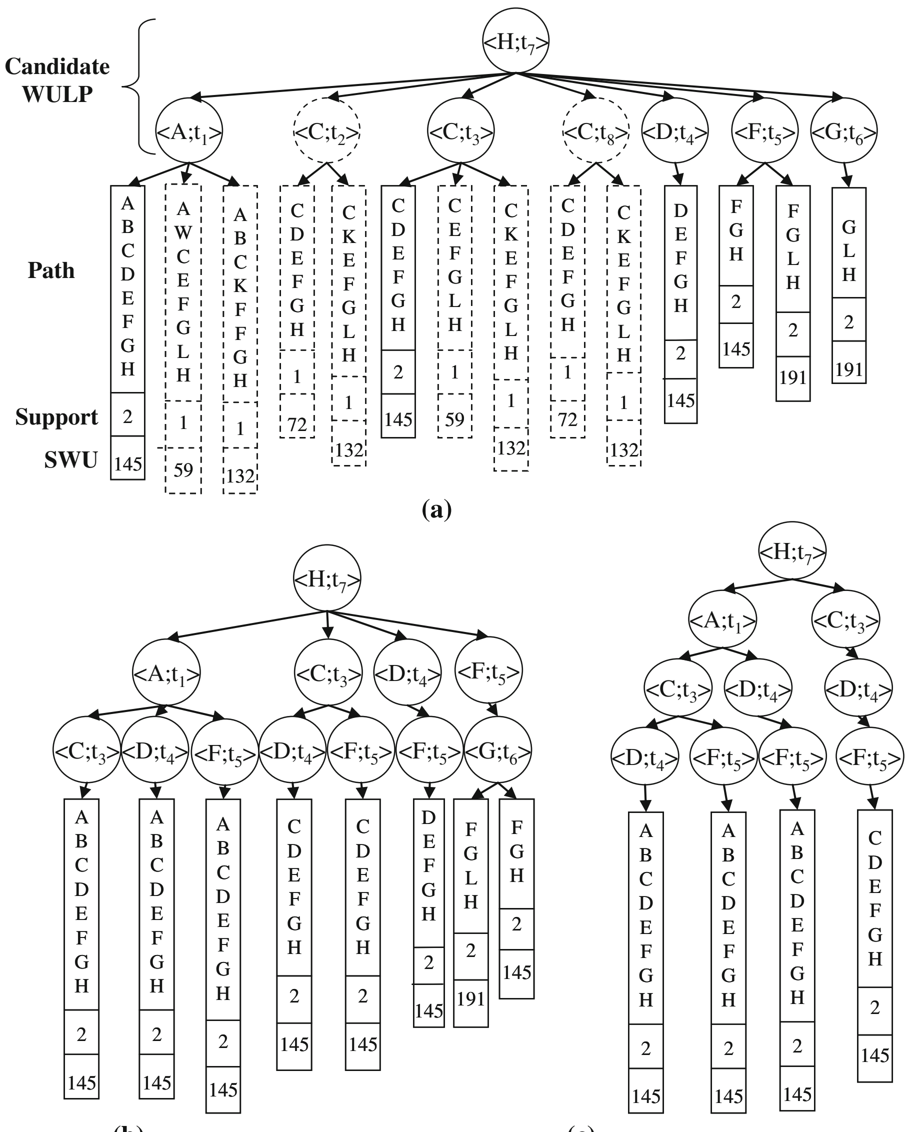


Fig. 3 Candidate-trees of <H : t_7>. **a** 2-candidate-tree. **b** 3-candidate-tree (after pruning). **c** 4-candidate-tree (after pruning)

We describe the process of generating WULIs by an example. Consider the mobile transaction sequence database in Table 3 and the utility table in Table 4. Assume the minimum support threshold is 2 and the minimum utility threshold is 100. In the first step, WULIs whose supports and SWUs are larger than or equal to the two thresholds are generated by the processes similar to the first step in Sect. 4.1. In this case, eight WULIs shown in Table 5 are generated. Note that they are also 1-WULPs. Then the 1-WULPs are mapped sequentially into a mapping table as shown in Table 5.

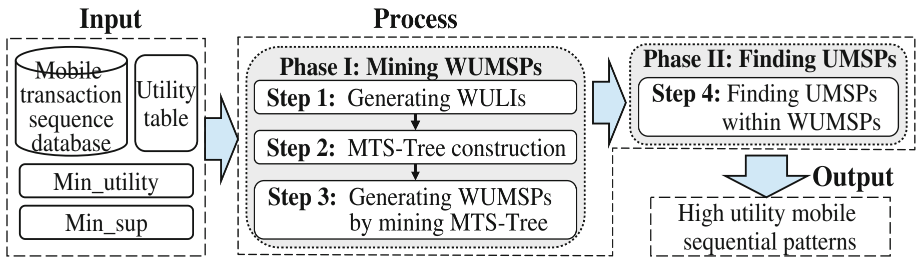

Input
Process
Mobile transaction sequence database
Utility table
Min_utility
Min_sup
Phase I: Mining WUMSPs
Step 1: Generating WULIs
Step 2: MTS-Tree construction
Step 3: Generating WUMSPs by mining MTS-Tree
Phase II: Finding UMSPs
Step 4: Finding UMSPs within WUMSPs
Output
High utility mobile sequential patterns

Fig. 4 The framework of the proposed algorithm $\mathrm{UMSP}_{\mathrm{T(DFG)}}$

```
Algorithm (Step 2 of UMSP_T(DFG): MTS-Tree construction)
Input: Mobile transaction sequence database DB, mapping table MT
Output: MTS-Tree
1.  create a header table H
2.  create a root R for an MTS-Tree T
3.  foreach mobile transaction sequence $S_i$ in DB do
4.    let path_start = false
5.    call InsertMTS_Tree($S_i$, R, MT, sid)

Procedure InsertMTS_Tree($S_i$, R, MT, sid)
1.  if $S_i$ is not NULL then
2.    divide $S_i$ into [x|X]
    /* x: the first loc-itemset of $S_i$. X: the remaining list of $S_i$ */
3.    let temppath = NULL
4.    if there is a combination y' of x exists in MT then
5.      convert x to the HTWULI y in MT
6.      if R has a child node C where C.location = y.location then
7.        if y.item exists in C.items then
8.          insert sid into C.[y.item].sid
9.        else
10.         create a new item y.item to C.items
11.         insert sid to C.[y.item].sid
12.     else
13.       create a new node C as a child node of R
14.       let C.location = y.location
15.       let C.item = y.item
16.       append sid to C.[y.item].sid
17.     update y's WULI, sup, TWU in H
18.     if temppath = NULL then
19.       append temppath and sid to C.pathtable
20.       let temppath = NULL
21.   else
22.     append x.location to temppath
23.   if X is not NULL then
24.     call InsertMTS_Tree(X, C, MT, sid)
```

Fig. 5 The procedure of MTS-Tree construction

4.2.1 The construction of MTS-Tree

The procedures of MTS-Tree construction are shown in Fig. 5. It can be completed after one scan of the original database. Without loss of generality, we give a formal definition for MTS-Tree first.

Definition 8. (*MTS-Tree*): In MTS-Tree, each node N includes $N_{.location}$, $N_{.itemset}$, $N_{.SID}$ and a path table. N is represented by the form <$N_{.location}[N_{.itemset1}] : N_{.SID1}$;

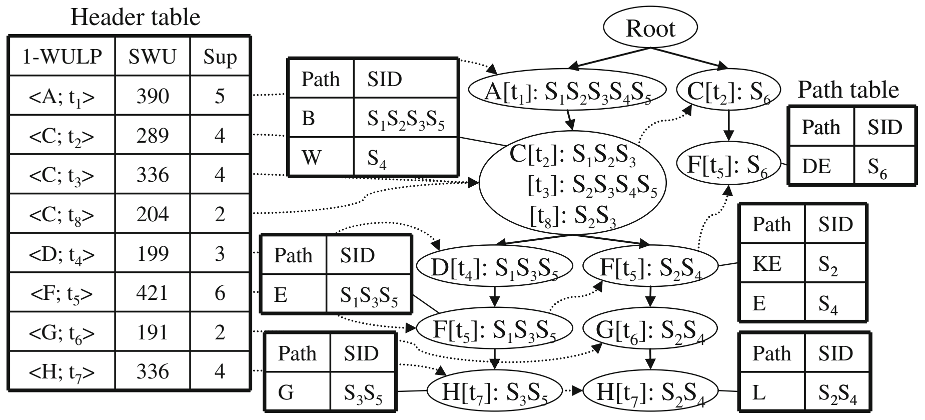


Fig. 6 An example of MTS-Tree

$[N._{itemset2}]$: $N._{SID2}$; $\ldots$>.$N._{location}$ records the node's location. Each node has several itemsets $N._{itemset}$ which represent the itemsets that happened in the same location. For each itemset in a node, it has a string of sequence identifiers, $N._{SID}$, which records the mobile transaction sequences including the itemset. A *path table* records the paths, which are a series of locations without purchasing item from $N's$ parent node to N, and the SIDs of the paths. Moreover, a *header table* is applied to efficiently traverse the nodes of a MTS-Tree. In a header table, each *entry* is composed of a 1-WULP, the SWU and support of the 1-WULP and a link which points to its first occurrence in MTS-Tree.

Now we introduce the processes of the MTS-Tree construction, i.e., the second step of UMSP$_{\text{T(DFG)}}$, by continuing the example. At the beginning, the first mobile transaction sequence S$_1$ is read. The first transaction in S$_1$ is (A; {[i$_1$, 2]}), so we find the corresponding loc-itemset of <A; i$_1$> in the mapping table in Table 5. Thus, <A; i$_1$> is converted into <A; t$_1$>. Then it is inserted into MTS-Tree. Since there is no corresponding node in the MTS-Tree, a new node <A[t$_1$] : S$_1$ > is created. The loc-itemset <A; t$_1$> is also inserted as an entry into the header table.

Subsequently, the second transaction (B; null) is evaluated. Since it has no purchased item, the location B is kept. Then the third transaction (C; {[i$_2$, 1]}) is checked in the mapping table and converted into <C; t$_2$>. Then the node <C [t$_2$] : S$_1$> is created and inserted as a child node of <A[t$_1$] : S$_1$>. Since there is a current kept path B, B and its SID S$_1$ are put into the path table of the node <C [t$_2$] : S$_1$>. The loc-itemset <C; t$_2$> and its relevant information are also inserted into the header table. The remaining transactions in S$_1$ are inserted into the MTS-Tree sequentially by the same way.

Subsequently, the second mobile transaction sequence S$_2$ is read. For the first transaction (A; {[i$_1$, 3]}), it is converted into <A; t$_1$>. Since there is already a node <A[t$_1$] : S$_1$> with the same location A in MTS-Tree, the SID S$_2$, SWU and support of the node are updated. Then the location B of the second transaction (B; null) is kept, since it has no purchased item. Next, the third transaction (C; {[i$_2$, 2], [i$_3$, 5]}) is evaluated. By the mapping table, it is converted into <C; t$_2$>, <C; t$_3$> and <C; t$_8$>. Since there is a node <C[t$_2$] : S$_1$> with location C and item t$_2$, the SWU and support of the node are updated. On the other hand, for <C; t$_3$> and <C; t$_8$>, they are stored as new itemsets in the same node <C>. After processing this transaction, the node becomes <C [t$_2$] : S$_1$S$_2$; [t$_3$]: S$_2$; [t$_8$] : S$_2$>. The remaining transactions in S$_2$ are inserted into the MTS-Tree sequentially by the same way. After all sequences in D are inserted, we can get the MTS-Tree as shown in Fig. 6. By converting the original loc-itemsets

```
Algorithm (Step 3 of UMSP_T(DFG): Generating WUMSPs)
Input: A MTS-Tree $T$, a header table $H$, a minimum utility threshold $\varepsilon$, and
a minimum support threshold $\delta$
Output: A WUMSP-Tree $T'$
1.  Let $T'$ be an WUMSP-Tree
2.  foreach WULI $\alpha$ in the bottom entry of $H$ do
3.    trace the link of WULI $\alpha$ in $H$ to get 1-WULP
4.    add 1-WULP $\alpha$ and sid to $T'$
5.    create a conditional MTS-Tree $CT_\alpha$ and a header table $H_\alpha$
6.    call WUMSP-Mine($CT_\alpha$, $H_\alpha$, $\alpha$)

Procedure WUMSP-Mine($CT_\alpha$, $H_\alpha$, $\alpha$)
1.  foreach WULI $\beta$ in $H_\alpha$ do
2.    if $sup(\beta) < \delta$ or $SWU(\beta) < \varepsilon$ then
3.      delete $\beta$ from $CT_\alpha$ and $H_\alpha$
4.      if there exists an empty node $X$ in $CT_\alpha$
5.        delete $X$
6.        append the $X'$s children nodes to $X'$s parent node
7.  foreach WULI $\beta$ of $HT_\alpha$ do
8.    add WULP $\beta\alpha$ and its sids to $T'$
9.    trace the paths of $\beta\alpha$ in $T$
10.   calculate the corresponding supports and SWUs
11.   add the paths to the path table of $\beta\alpha$ in the node of $\beta$ in $T'$
    /* line 12-14: Path pre-checking technique*/
12.   if there exists a path in $\beta\alpha$ to form a WUMSP $Y$, such that $sup(Y) \geqq$
      $\delta$ and $SWU(Y) \geqq \varepsilon$ then
13.     create a conditional MTS-Tree $CT_{\beta\alpha}$ and a header table $H_{\beta\alpha}$
14.     call WUMSP-Mine($CT_{\beta\alpha}$, $H_{\beta\alpha}$, $\beta\alpha$)
```

Fig. 7 The procedure of generating WULPs

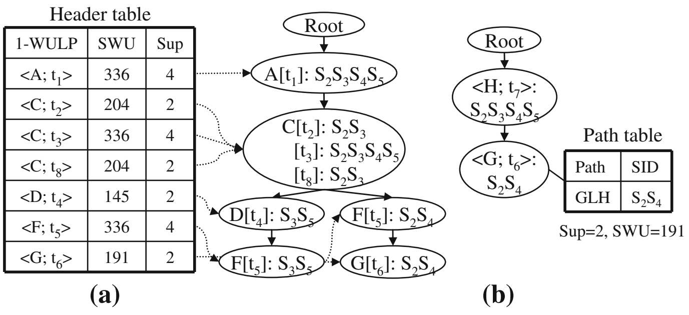
in the database to the ones in the mapping table, we do not need to store all combinations of the original loc-itemsets into the MTS-Tree. It can save much computational cost and memory space.

4.2.2 Generating WUMSPs from MTS-Tree

After constructing MTS-Tree, now we describe the step 3 of UMSP$_{T(DFG)}$. The purpose of this step is generating WUMSPs from MTS-Tree by the *depth first generation* strategy. The procedures are shown in Fig. 7. First, WULPs and their *conditional MTS-Trees* are generated by tracing the links from the entries in the header table of the MTS-Tree. After generating the WUMSPs in a conditional MTS-Tree, to efficiently trace the WUMSPs and save the memory space of storing them, the WULPs are stored into a *WUMSP-Tree* (*high sequence Weighted Utilization Mobile Sequential Pattern Tree*) in a compact form. The definition of WUMSP-Tree is given as follows:

Definition 9 (*WUMSP-Tree*:) In a WUMSP-Tree, each node is a WULI. For a node N, $N._{SID}$ and its path table are recorded. N is represented by the form <$N._{WULI}$: $N._{SID}$>. For the WULI in a node, it has a string of sequence identifiers, $N._{SID}$, which records the mobile ransaction sequences that the WULI occurs. A path table which records the paths from the node N to root and the SIDs of the paths is also kept.

Each node of the WUMSP-Tree represents a WUMSP. By tracing from a node to root of a WUMSP-Tree, a WULP can be derived. Take the WUMSP-Tree in Fig. 8 for example; we



Header table

1-WULP	SWU	Sup
<A; t_1>	336	4
<C; t_2>	204	2
<C; t_3>	336	4
<C; t_8>	204	2
<D; t_4>	145	2
<F; t_5>	336	4
<G; t_6>	191	2

Path table

Path	SID
GLH	S_2S_4

Sup=2, SWU=191

Fig. 8 Conditional MTS-Tree of <H; t_7> and the corresponding WUMSP-Tree. **a** Conditional MTS-Tree of <H; t_7>. **b** WUMSP-Tree

can get a WULP <G; t_6> <H; t_7> by tracing from the node to root. Moreover, we can get a WUMSP <{<G; t_6><H; t_7>}; GLH > by combining the path GLH in the node <G; t_6> with the WULP. After tracing all nodes in WUMSP-Tree, all WUMSPs can be obtained.

During the processes of generating WULPs, if the length of the WULPs is larger than 1, besides the processes of tracing path, a *path pre-check technique* will be performed to prune the moving patterns which cannot fit the user-specified thresholds.

Definition 10 (*Path pre-check*): If there exists no path for a WULP X to form a WUMP Y such that $sup(Y) \geq \delta$ and $SWU(Y) \geq \varepsilon$, X is pruned.

Path pre-check technique is used for trimming the search space. By using this technique, the number of conditional MTS-Tree can be reduced effectively and the mining performance can be improved more.

Now we introduce the processes of the step 3, i.e., generating WUMSPs from MTS-Tree, by continuing the example in the previous section. First, the bottom entry <H; t_7> in the header table of the MTS-Tree shown in Fig. 6 is checked and a WULP <H; t_7> is generated. Then <H; t_7> and its SIDs are inserted as the first child node of the root of the WUMSP-Tree. Since <H; t_7> is a 1-WULP, its path is not traced. Then the conditional MTS-Tree of <H; t_7> shown in Fig. 8a is constructed by tracing all nodes labeled <H; t_7> to the root of the MTS-Tree. The nodes labeled <H; t_7> are acquired by tracing the links from the entry of <H; t_7> in header table. Note that the conditional MTS-Trees do not need any path table.

Subsequently, in the header table of the conditional MTS-Tree of <H; t_7>, the last entry <G; t_6> is checked and a WULP <G; t_6><H; t_7> is generated and inserted into the WUMSP-Tree. Since there is already a node <H; t_7> in the WUMSP-Tree, we just insert <G; t_6> as a child node of <H; t_7>. At the same time, the path of the WULP <G; t_6><H; t_7> is traced in the original MTS-Tree. By the links from the entry <H; t_7> in header table, we can get the node <H[t_7]> with the SIDs S_2 and S_4. The node <H[t_7]> is traced up until the node <G; t_6> is reached to obtain the paths between the nodes. Then a path GLH is found. After tracing the path, a WUMSP <{<G; t_6><H; t_7>}; GLH> whose support is 2 and SWU is 191 is found. By the path pre-checking technique, since its support and SWU are both no less than the two thresholds, it is kept, and the path is added into the node <G; t_6> of the WUMSP-Tree. The WUMSP-Tree now is shown in Fig. 8b. Generating the patterns from the MTS-Tree by the above processes recursively, all WUMSPs can be generated.

After generating all WUMSPs, an additional database scan will be performed to find UMSPs from the set of WUMSPs. The WUMSPs whose utilities are larger than or equal to the minimum utility threshold will be regarded as UMSPs. Moreover, since the WUMSPs in

WUMSP-Tree include SIDs, instead of checking all mobile transaction sequences, they will just check the specified sequences. By applying this process, the performance will be more improved.

4.3 Improved method: $\text{UMSP}_{\text{T(BFG)}}$

In $\text{UMSP}_{\text{T(DFG)}}$, since the number of combinations of 2-WULPs is quite large, many conditional MTS-Trees will be generated. Dealing with these conditional MTS-Trees is time-consuming in the mining processes. Moreover, tracing the paths of WULPs in the processes of generating WUMSPs also spends much time. If we can decrease the number of WULPs requiring verification, especially the large number of 2-WULPs, the performance can be improved more. Therefore, how to speed up the processes about 2-WUMSPs is a crucial problem.

To conquer this problem, we propose an improved tree-based algorithm named $UMSP_{T(BFG)}$ (*mining high Utility Mobile Sequential Pattern by a tree-based method with a Breadth First Generation strategy*). The difference between the two algorithms is that $\text{UMSP}_{\text{T(BFG)}}$ uses a breadth first generation strategy for generating 2-WUMSPs. Within the strategy, a *possible succeeding node check technique* is applied. By this technique, the size of the conditional MTS-Trees will be smaller, and the 2-moving patterns which cannot be 2-WUMSPs will be pruned in advance.

Instead of generating a 2-WULP by combining the last entry with the 1-WULP of a conditional MTS-Tree, in the breadth first generation strategy, 2-WULPs are generated by combining all 1-WULPs in the header table with the 1-WULP of the conditional MTS-Tree. After generating the 2-WULPs, their paths, supports and SWUs are checked in advance. The valid paths will be stored in the corresponding nodes of WUMSP-Tree. While generating 2-WULPs, $\text{UMSP}_{\text{T(BFG)}}$ applies a *possible succeeding node check technique* for pruning useless 2-WULPs, which is addressed as follows:

Definition 11 (*Possible succeeding node check*) While generating 2-WULPs of a 1-WULP X in X's conditional MTS-Tree, all 1-WULPs in the header table are inserted as children nodes of X in the WUMSP-Tree in advance. If there exists no path in a WULP Y to form a WUMSP Z such that $sup(Z) \geq \delta$ and $SWU(Z) \geq \varepsilon$, Y is pruned.

Furthermore, only the nodes kept in the WUMSP-Tree are able to be succeeding nodes of the WUMSP-Tree in the later mining processes.

In the following paragraphs, we use the same example as the previous section. The MTS-Tree shown in Fig. 6 and the conditional MTS-Tree of <H; t_7> shown in Fig. 8 are constructed by the same processes in previous section. <H; t_7> is inserted into the WUMSP-Tree as the first node. Different from $\text{UMSP}_{\text{T(DFG)}}$, in the processes of generating 2-WULPs of $\text{UMSP}_{\text{T(BFG)}}$, all 1-WULPs in the header table of the conditional MTS-Tree of <H; t_7> are inserted as children nodes of the node <H; t_7> in the WUMSP-Tree, that is, all 2-WULPs of the conditional MTS-Tree of <H; t_7> are generated in advance. The paths of the 2-WULPs are then generated by tracing the original MTS-Tree. Combining the 2-WULPs and the paths, 2-moving patterns are generated. Also, their supports and SWUs are obtained. The results are shown in Fig. 9.

By Fig. 9, since the supports or SWUs of the 2-moving patterns <{<C; t_8><H; t_7>}; CFGLH>, <{<C;t_8><H; t_7>};CDFGH>, <{<C; t_3><H; t_7>};CKEFGLH>, <{<C; t_3><H; t_7>};CEFGLH>, <{<C; t_2><H; t_7>};CKEFGLH>, <{<C; t_2><H; t_7>};CDEFGH>, <{<A; t_1><H; t_7>};ABCKEFGLH> and <{<A; t_1><H; t_7>};AWCEFGLH are less than the thresholds, their relevant paths are pruned from the path tables of the WUMSP-Tree.

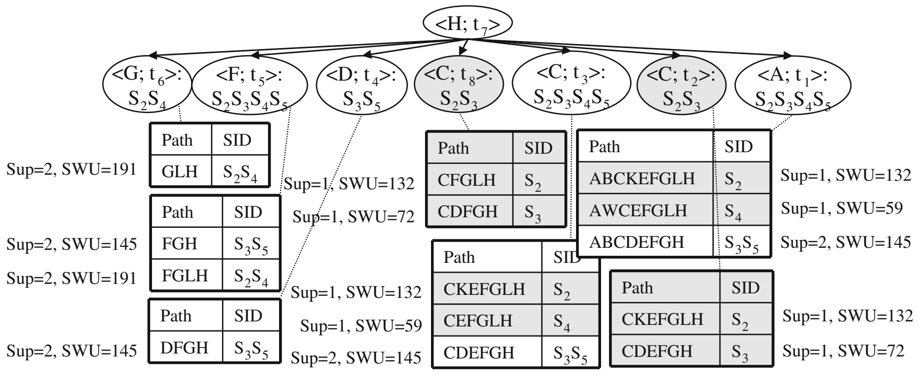

<H; t$_7$>
<G; t$_6$>: S$_2$S$_4$
<F; t$_5$>: S$_2$S$_3$S$_4$S$_5$
<D; t$_4$>: S$_3$S$_5$
<C; t$_8$>: S$_2$S$_3$
<C; t$_3$>: S$_2$S$_3$S$_4$S$_5$
<C; t$_2$>: S$_2$S$_3$
<A; t$_1$>: S$_2$S$_3$S$_4$S$_5$
Path | SID
GLH | S$_2$S$_4$ — Sup=2, SWU=191
Path | SID
FGH | S$_3$S$_5$ — Sup=2, SWU=145
FGLH | S$_2$S$_4$ — Sup=2, SWU=191
Path | SID
DFGH | S$_3$S$_5$ — Sup=2, SWU=145
Path | SID
CFGLH | S$_2$ — Sup=1, SWU=132
CDFGH | S$_3$ — Sup=1, SWU=72
Path | SID
CKEFGLH | S$_2$ — Sup=1, SWU=132
CEFGLH | S$_4$ — Sup=1, SWU=59
CDEFGH | S$_3$S$_5$ — Sup=2, SWU=145
Path | SID
ABCKEFGLH | S$_2$ — Sup=1, SWU=132
AWCEFGLH | S$_4$ — Sup=1, SWU=59
ABCDEFGH | S$_3$S$_5$ — Sup=2, SWU=145
Path | SID
CKEFGLH | S$_2$ — Sup=1, SWU=132
CDEFGH | S$_3$ — Sup=1, SWU=72

Fig. 9 An example of WUMSP-Tree generated by UMSP$_{T(BFG)}$

Table 7 Parameter settings

Parameter descriptions	Default
D: Number of mobile transaction sequences	50 k
P: Average length of mobile transaction sequences	20
T: Average number of items per transaction	2
N: Size of mesh network	8
n$_I$: The range of the number of items sold in each location	200
P$_b$: The probability that user makes the transaction in the location	0.5
w: Unit profit of each item	1–1,000
q: Number of purchased items in transactions	1–5

Moreover, since there is no valid path in the nodes <C; t$_8$> and <C; t$_2$>, the two nodes are also pruned. In Fig. 9, the pruned nodes and paths in the WUMSP-Tree are labeled with grey. By the WUMSP-Tree, we can know the possible succeeding nodes of <G; t$_6$> are <F; t$_5$>,<D; t$_4$>, <C; t$_3$> and <A; t$_1$>.

After ascertaining which 2-moving patterns need to be pruned, the relevant nodes and entries in the conditional MTS-Tree of <H; t7> are also pruned. After this step, the mining processes proceed without the pruned nodes in both the WUMSP-Tree and the conditional MTS-Tree of <H; t7>. The remaining conditional MTS-Tree is much smaller than the original one. Moreover, since the useless entries are pruned in the header table, they will never be checked in the following processes. Therefore, the search space can be further reduced and the mining performance is further improved.

5 Experimental evaluations

In this section, we evaluate the performance of the proposed algorithms. The experiments were performed on a 2.4 GHz Processor with 1.6 gigabyte memory, and the operating system is Microsoft Windows Server 2003. The algorithms are implemented in Java language. The default settings of the parameters are listed in Table 7. The settings of parameters related to mobile commerce environment and utility mining are similar to [13,26], respectively.

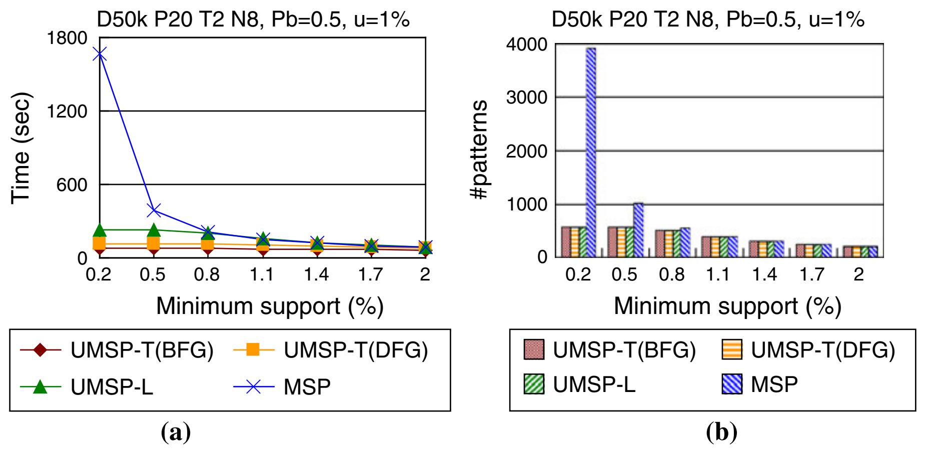


Fig. 10 The performance under varied minimum support thresholds. **a** Execution time. **b** Number of patterns after phase I

For comparing the performance of the proposed algorithms, we extend the algorithm TJ_{PF} in [26] to form a basic algorithm for mining high-utility mobile sequential patterns, called MSP. The processes of MSP are as follows: first, the mobile sequential patterns whose supports are no less than the minimum support threshold are generated by TJ_{PF}. Then an additional check of the actual utilities of the mobile sequential patterns is performed for finding high-utility mobile sequential patterns. Note that the main difference of MSP and $UMSP_L$ is that MSP does not consider utility in phase I. In the following experiments, the performance of MSP is compared with that of the three proposed algorithms.

5.1 Performance under varied thresholds

The first part of the experiments is the performance under various minimum support thresholds. In the experiments, the minimum utility threshold is set to 1 %. The results for the execution time and the number of patterns after phase I under varied minimum support thresholds are shown in Fig. 10. For the three proposed algorithms, the patterns after phase I are WUMSPs, on the other hand, for MSP, the patterns are mobile sequential patterns. In Fig. 10a, it can be seen that MSP requires much more execution time than the other algorithms. The reason is that since MSP does not consider utility in phase I, the number of generated patterns is much larger than that of other algorithms as shown in Fig. 10b. MSP spends much time on processing redundant patterns, so its performance is the worst. Besides, although the number of WUMSPs generated by the proposed three algorithms is the same, their execution time is different. Overall, the tree-based algorithms are better than the level-wise ones especially when the minimum support threshold is low.

Next, we show the performance under various minimum utility thresholds. In the experiments, the minimum support threshold is set to 0.5 %. The results are shown in Fig. 11. Overall, the tree-based algorithms are better than the level-wise ones. Besides, since MSP does not consider utility in phase I, its execution time and number of generated patterns remain the same. On the contrary, both of the two results of the proposed three algorithms decrease with the minimum utility threshold increasing. Besides, when the minimum utility threshold is lower, the execution time of $UMSP_L$ is closer to that of MSP. The reason is that

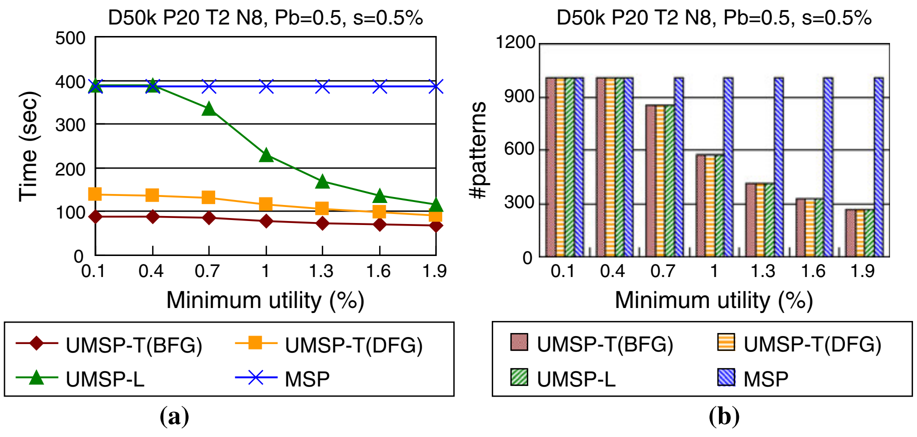


Fig. 11 The performance under varied minimum utility thresholds. **a** Execution time. **b** Number of patterns after phase I

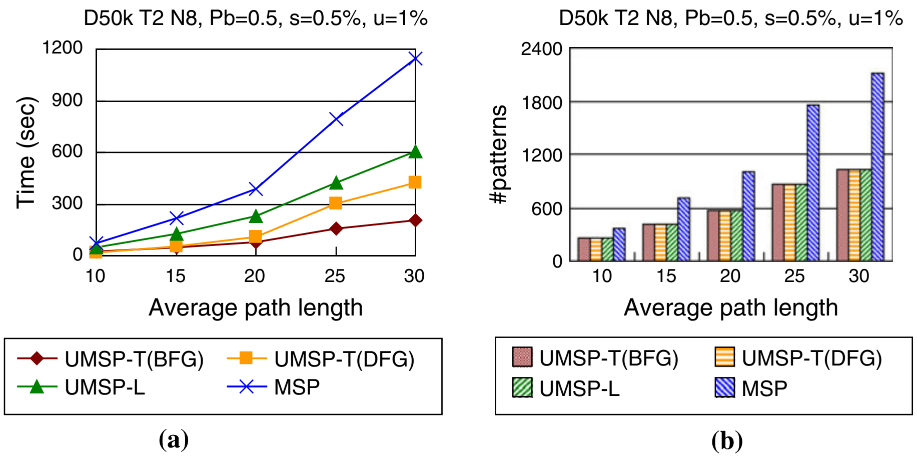


Fig. 12 The performance under varied average length of mobile transaction sequences. **a** Execution time. **b** Number of patterns after phase I

when the minimum utility threshold is lower, fewer candidates could be pruned. In Fig. 11a, when the minimum utility threshold is below 0.4 %, almost no candidate can be pruned by this threshold. Thus, the performance of the two algorithms is almost the same.

5.2 Performance under varied parameter settings

In this first part of the experiments, we show the performance under varied average length of mobile transaction sequences. The results are shown in Fig. 12. In the experiments, it can be seen that the execution time of all the four algorithms increases with the average length of mobile transaction sequences. The reasons are addressed as follows: When the mobile transaction sequences are longer, the length of generated patterns will become longer. To generate longer patterns, the level-wise methods must deal with more passes and the tree-based methods must generate more conditional MTS-Trees. On the other hand, since the

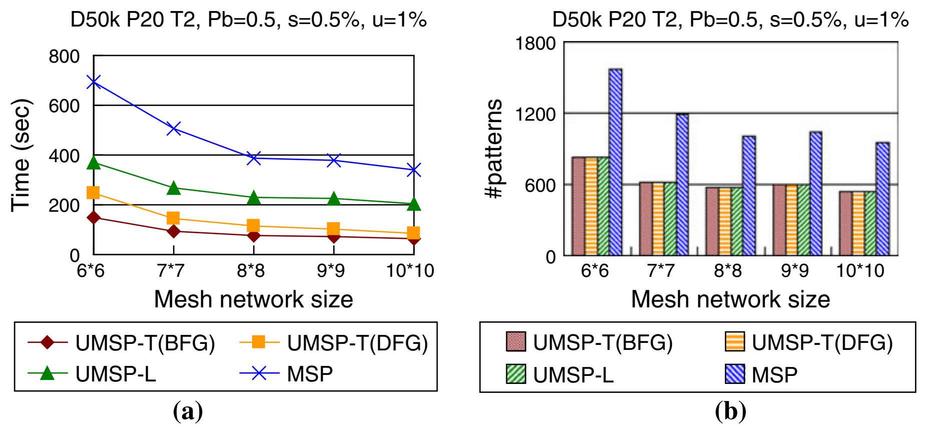


Fig. 13 The performance under varied mesh network size. **a** Execution time. **b** Number of patterns after phase I

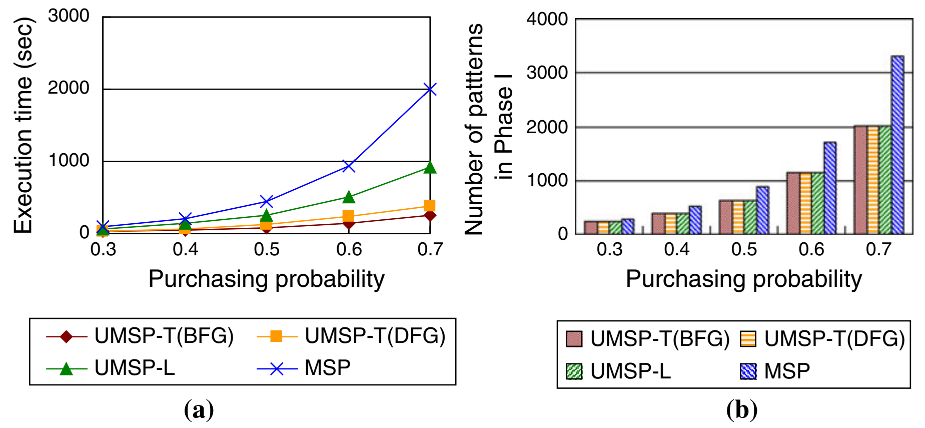


Fig. 14 The performance under varied purchasing probabilities. **a** Execution time. **b** Number of patterns after phase I

longer patterns contain more sub-patterns, more patterns must be generated when the length of mobile sequential patterns is longer. Thus, the execution time of the four algorithms also becomes longer. Overall, the tree-based algorithms are still better than the level-wise ones.

Subsequently, we show the performance under varied mesh network size. The results are shown in Fig. 13. By Fig. 13a, it can be seen that the execution time of all the four algorithms decreases with the size of mesh network. The reason is that when the size of mesh network is larger, the database will be sparser; therefore, the patterns generated in phase I will become fewer and the time cost of mining processes will be reduced.

Third, we show the performance under varied purchasing probabilities (P_b) in each transaction. The results are shown in Fig. 14. We can see that in Fig. 14a the best performer is delivered by $UMSP_{T(BFG)}$, followed by $UMSP_{T(DFG)}$, $UMSP_L$, and finally MSP. In Fig. 14b, it can be observed that with the increasing of purchasing probability, the number of WUMSPs increases in a slightly rising trend. The reason is the combinations of items rise due to the

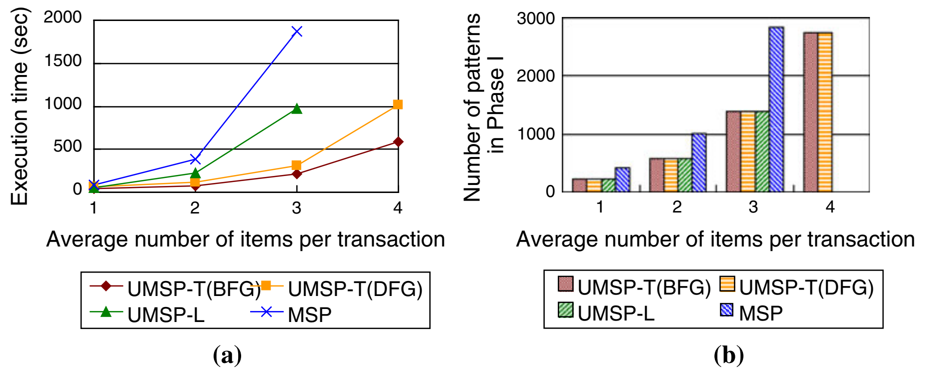


Fig. 15 The performance under varied average number of items per transaction. **a** Execution time. **b** Number of patterns after phase I

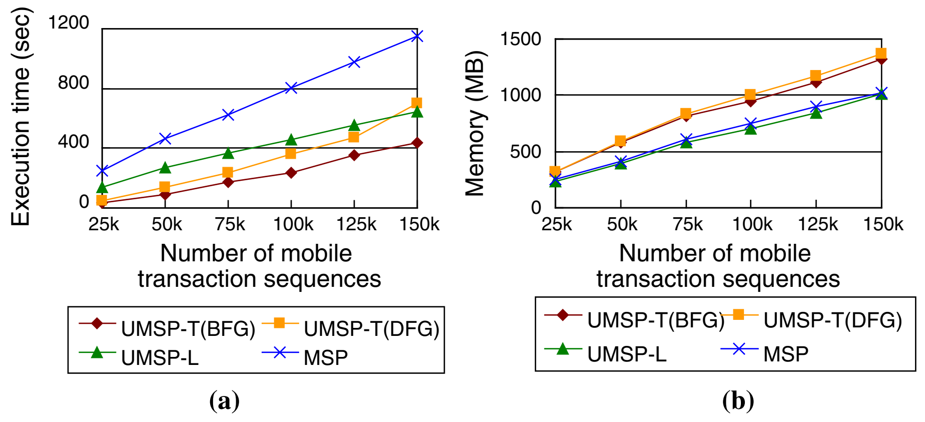


Fig. 16 The performance under varied number of mobile transaction sequences. **a** Execution time. **b** Memory usage

increase of purchased items in the transactions. Hence, the execution time of the compared algorithms increases with the same trend.

In the last of this sub-session, we show the performance under varied average number of items per transaction (T). The results are shown in Fig. 15. When T is larger than 3, the runtime of MSP and UMSP$_L$ is too long to be executed. In Fig. 15b, we can know that the combination of itemsets increases exponentially with the increasing number of items per transaction. Thus, the execution time of compared algorithms in Fig. 15a also increases in an exponential trend. In this figure, it can also be observed that the performance of tree-based algorithms outperforms the level-wise ones.

5.3 Scalability of the compared algorithms

The last part of the experiments is the scalability of the compared algorithms. The experimental results about execution time and memory usage are shown in Fig. 16a, b, respectively. Due to the constraint of main memory, we can only show the results with the maximum sequence size of 150 k. In Fig. 16a, we can see that when the number of mobile transaction sequences is larger, the execution time of $\text{UMSP}_{\text{T(BFG)}}$, UMSP_{L} and MSP increases linearly. When the number of mobile transaction sequences of $\text{UMSP}_{\text{T(DFG)}}$ is larger than 125k, its performance becomes worse than that of UMSP_{L}. This is because that there are more combinations of length 2 patterns with the increase of mobile transaction sequences. For $\text{UMSP}_{\text{T(DFG)}}$, it needs to generate more local MTS-Trees without skipping infrequent paths. However, $\text{UMSP}_{\text{T(BFG)}}$ keeps a better scalability than $\text{UMSP}_{\text{T(DFG)}}$ since the former deals with not only smaller but also fewer conditional MTS-Trees than the latter. Figure 16b shows that all the compared algorithms have good scalability in memory usage. Besides, we can observe that the tree-based algorithms take more memory space because many kinds of information need to be recorded in the proposed MSP-Trees. The two figures mentioned above show the trade-off between runtime and memory for the tree-based and level-wise algorithms.

5.4 Discussions

Further discussions about the experiments are addressed as follows: First, the results about the number of patterns after phase I show that many mobile sequential patterns can be pruned by the minimum utility threshold during the mining processes. The performance of MSP is the worst among the four algorithms since MSP considers only support in phase I. Second, the experimental results show that the tree-based algorithms outperform the level-wise ones. The reasons are as follows: (1) In the processes of phase I, the tree-based algorithms do not need to generate candidate patterns and perform an additional database scan to check them. (2) While tracing path, the level-wise methods need to find the sub-paths by scanning all complete paths in D_{T}. On the contrary, the tree-based algorithms just need to trace the corresponding nodes in the MTS-Tree directly. (3) While generating the UMSPs in phase II, tree-based methods only check the mobile transaction sequences with the specified SIDs which are recorded in the WUMSPs. Third, the experimental results show that $\text{UMSP}_{\text{T(BFG)}}$ outperforms $\text{UMSP}_{\text{T(DFG)}}$. It is because that $\text{UMSP}_{\text{T(BFG)}}$ checks the 2-WUMSPs by the breadth first strategy. The strategy effectively reduces much search space in the mining processes. Thus, the performance of $\text{UMSP}_{\text{T(BFG)}}$ is better than that of $\text{UMSP}_{\text{T(DFG)}}$.

By the above experiments, the proposed algorithms are shown to outperform the state-of-the-art mobile sequential pattern algorithm MSP. Among the three algorithms, the performance of $\text{UMSP}_{\text{T(BFG)}}$ is the best since the MTS-tree is an efficient tree structure and the breadth first strategy effectively enhances the mining performance. In addition, although the performance of the tree-based algorithms is better than that of the level-wise ones, the memory usage of the latter is slightly better. As a consequence, these experiments show a trade-off between the performance and memory usage for the two frameworks.

6 Conclusions

In this research, we explored a novel data mining issue about mining high-utility mobile sequential patterns in mobile commerce environments. To our best knowledge, this paper is the first work on the combination of mobility pattern mining and utility mining. Three algorithms, namely UMSP_{L}, $\text{UMSP}_{\text{T(DFG)}}$ and $\text{UMSP}_{\text{T(BFG)}}$, were developed under different frameworks, i.e., level-wise and tree-based ones, for efficiently mining high-utility mobile sequential patterns. The experimental results show that the proposed algorithms outperform the state-of-the-art mobile sequential pattern algorithm. Moreover, the performance of the tree-based algorithms outperforms level-wise ones in most conditions. For future work, new algorithms which improve the mining performance will also be explored.

Acknowledgments This research was partially supported by National Science Council, Taiwan, under grant no. NSC100-2631-H-006-002 and NSC100-2218-E-006-001 as well as supported by Top University project of Ministry of Education, Taiwan.

References

1. Achar A, Laxman S, Sastry PS (2011) A unified view of the apriori-based algorithms for frequent episode discovery. Knowl Inf Syst. doi:10.1007/s10115-011-0408-2
2. Agrawal R, Srikant R (1994) Fast Algorithms for Mining Association Rules. In: Proceedings of the 20th interantional conference on very large data bases, pp 487–499
3. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of 11th international conference on data mining, pp 3–14
4. Ahmed CF, Tanbeer SK, Jeong B-S, Lee Y-K (2009) Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans Knowl Data Eng 21(12):1708–1721
5. Cao L (2010) In-depth behavior understanding and use: the behavior informatics approach. Inf Sci 180(17):3067–3085
6. Chan R, Yang Q, Shen Y (2003) Mining high utility itemsets. In: Proceedings of third IEEE international conference on data mining, pp 19–26
7. Chen M-S, Park J-S, Yu PS (1998) Efficient data mining for path traversal patterns. IEEE Trans Knowl Data Eng 10(2):209–221
8. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of the ACM-SIGMOD international conference on management of data, pp 1–12
9. Hung C-C, Peng W-C (2011) Clustering fragmented trajectories for mining movement behaviors. In: Proceedngs of 2011 workshop on behavior informatics
10. Kim H, Park J-H (2011) Evaluating the regularity of human behavior from mobile phone usage logs. In: Proceedings of 2011 workshop on behavior informatics
11. Lee SC, Paik J, Ok J, Song I, Kim UM (2007) Efficient mining of user behaviors by temporal mobile access patterns. Int J Comput Sci Secur 7(2):285–291
12. Li Y-C, Yeh J-S, Chang C-C (2008) Isolated items discarding strategy for discovering high utility itemsets. Data Knowl Eng 64(1):198–217
13. Liu Y, Liao W-K, Choudhary A (2005) A fast high utility itemsets mining algorithm. In: Proceedings of utility-based data mining
14. Lu EH-C, Tseng VS (2009) Mining cluster-based mobile sequential patterns in location-based service environments. In: Proceedings of IEEE international conference on mobile data management
15. Lu EH-C, Huang C-W, Tseng VS (2009) Continuous fastest path planning in road networks by mining real-time traffic event information. In: Proceedings of the 2nd international symposium on intelligent informatics
16. Lu EH-C, Lee W-C, Tseng VS (2010) Mining fastest path from trajectories with multiple destinations in road networks. Knowl Inf Syst 29(1):25–53
17. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC (2004) Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Trans Knowl Data Eng 16(10):1–17
18. Senkul P, Salin S (2011) Improving pattern quality in web usage mining by using semantic information. Knowl Inf Syst 30(3):527–541
19. Shie B-E, Tseng VS, Yu PS (2010) Online mining of temporal maximal utility itemsets from data streams. In: Proceedings of the 25th annual ACM symposium on applied computing (SAC 2010), pp 1622–1626
20. Tseng VS, Lu EH-C, Huang C-H (2007) Mining temporal mobile sequential patterns in location-based service environments. In: Proceedings of the 13th IEEE international conference on parallel and distributed systems
21. Tseng VS, Lin WC (2005) Mining sequential mobile access patterns efficiently in mobile web systems. In: Proceedings of the 19th international conference on advanced information networking and applications, pp 867–871
22. Tseng VS, Wu C-W, Shie B-E, Yu PS (2010) UP-growth: an efficient algorithm for high utility itemsets mining. In: Proceedings of the 16th ACM SIGKDD conference on knowledge discovery and data mining (KDD'10), pp 253–262

23. Yao H, Hamilton HJ (2006) Mining itemset utilities from transaction databases. Data Knowl Eng 59: 603–626
24. Yen S-J, Chen CC, Lee Y-S (2011) A fast algorithm for mining high utility itemsets. In: Proceedings of 2011 workshop on behavior informatics
25. Yun C-H, Chen M-S (2000) Using pattern-join and purchase-combination for mining web transaction patterns in an electronic commerce environment. In: Proceedings of 24th IEEE annual international computer software and application conference, pp 99–104
26. Yun C-H, Chen M-S (2007) Mining mobile sequential patterns in a mobile commerce environment. IEEE Trans Syst Man Cybern Part C: Appl Rev 37(2):278–295

Author Biographies

Bai-En Shie received his Master's Degree in Computer Science and Information Engineering from Ming Chuan University, Taiwan, ROC, in 2006. He is currently pursuing his PhD Degree in the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, ROC. His research interests include data mining algorithm, utility mining, healthcare system, mobility pattern mining and temporal data mining.

Hui-Fang Hsiao received his B.S. degree in computer science and information engineering from National Taiwan Normal University Taiwan, R.O.C., in 2008 and M.S degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, R.O.C., in 2010. Her research interests include data mining, utility mining, mobility pattern mining, information retrieval and web mining.

Vincent S. Tseng is currently a professor at Department of Computer Science and Information Engineering/Institute of Medical Informatics at National Cheng Kung University (NCKU), Taiwan. He is also the president of Taiwanese Association for Artificial Intelligence and had acted as the director for Institute of Medical Informatics of NCKU during 2008 and 2011. Dr. Tseng has a wide variety of research interests covering data mining, biomedical informatics, multimedia databases and mobile Web technologies. He has published more than 200 research papers in referred journals and international conferences and has held/filed more than 15 patents in USA and R.O.C. He is on the editor board of several international journals and has also served as chair/program committee for a number of premier conferences related to data mining and database systems.