Mining Sequential Patterns Across Multiple Sequence Databases

Zhung-Xun Liao Wen-Chih Peng* Xing-Yuan Hu

Department of Computer Science

National Chiao Tung University

Hsinchu, Taiwan, ROC

E-mail:{hyhu, g9113, wcpeng}@cs.nctu.edu.tw

Received June 15, 2007 ; Accepted June 30, 2007

Abstract. In reality, sequential patterns may exist in multiple sequence databases. In this paper, we explore a novel sequential pattern mining problem: mining multi-domain sequential patterns across multiple domain sequence databases. We propose two algorithms, IndividualMine and PropagatedMine, for efficiently mining multi-domain sequential patterns. In algorithm IndividualMine, sequential patterns in each domain should first be discovered and then by iteratively combining sequential patterns among domain sequence databases, multi-domain sequential patterns are generated. Algorithm PropagatedMine performs sequential pattern mining only in one domain sequence database and propagates sequential patterns mined to other domain to generate corresponding sequential patterns so as to reduce the cost of mining. A comprehensive performance study is conducted and experimental results show the scalability and the efficiency of our proposed algorithms.

Keywords:

1 Introduction

Sequential pattern mining has attracted a significant amount of research efforts recently. The problem of sequential pattern mining is discovering frequent sequences with their occurrence counts being larger than or equal to the user-specified number, *min_support*, among a set of sequences [1]. Sequential pattern mining can be applied on business and marketing analysis, web page browsing behavior, symptomatic pattern of a disease, DNA sequence, hacker invasion detecting and to name a few. Due to the importance of sequential pattern mining, many efficient sequential pattern mining algorithms have been proposed recently [1][2][3][4][5]. However, existing sequential pattern mining algorithms only discover sequential behavior (e.g., buying behavior) in one domain, which are not sufficient for behavior analysis. One would like to discover sequential patterns across multiple domains. Such a sequential pattern across multiple domain sequence databases is referred to a multi-domain sequential pattern in this paper. A multidomain sequential pattern consists of sequences across multiple domains and for each item of a sequence, the corresponding items having the same order in different domain sequences occur in the same time slot. Note that a multi-domain sequential pattern captures cross relationship among multiple domains which in turn provides more significant knowledge. Applications of multi-domain sequential patterns include but are not limited to the following two.

- User behavior analysis in a mobile computing environment. Consider a mobile computing environment in Fig. 1, where mobile users can access three services (i.e., location tracking service, data searching service, and credit payment service) via mobile devices and each service is referred to one domain in this paper. Given a log of movements of a user from the location tracking service, one would mine user moving patterns referred to those areas in which the user frequently travels. As such, in Fig. 1, for each domain, sequential patterns in each domain are discovered by existing algorithms [1][2][6]. Note that in order to reflect behavior of a user in such environment, one would like to find more complex sequential patterns across multiple domains. An example of a multi-domain sequential pattern is shown in Fig. 1, where a user stays at area $\{A\}$, searches data items $\{1, 2\}$, and buys goods $\{\alpha, \beta\}$; then moves to area $\{B, C\}$, searches data $\{3, 4, 5\}$, and buys goods $\{\gamma\}$; and finally moves to area $\{D\}$, searches data $\{6, 7\}$, and buys goods $\{\theta, \delta\}$. Such a sequential pattern consists of sequences across multiple domains and provides more information to analyze user behaviors. For example, the user is motivated by the scene at location A and then buys goods $\{\alpha, \beta\}$ after surfing some web pages of $\{1, 2\}$.

* Correspondence author

	$time_1$	$time_2$	$time_3$
Location tracking	<(A)	(B,C)	(D) >
Search service	<(1,2)	(3,4,5)	(6,7)>
Credit payment	<(α,β)	(Δ)	(θ,δ)>

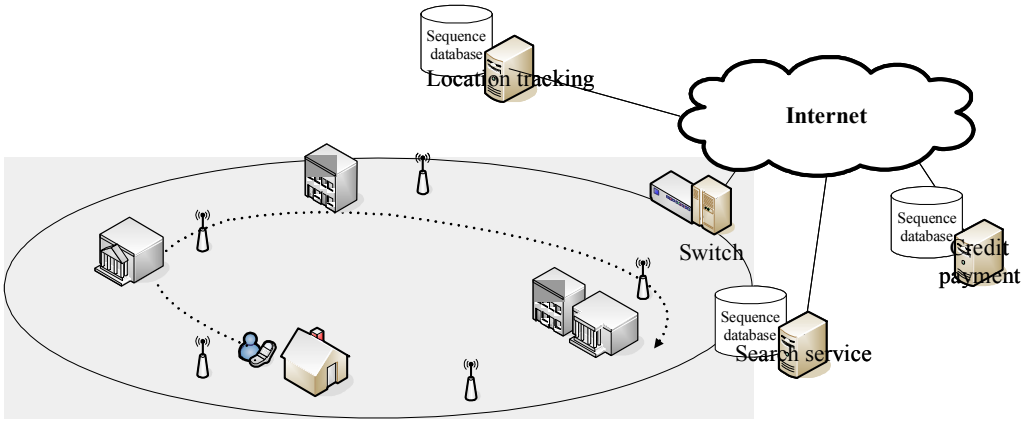

Fig. 1. An example of multi-domain sequential patterns in mobile computing environments.

- Behavior or event analysis in a sensor network. Imagine that a large amount of sensors are deployed in a smart home for behavior analysis. Sensors with different sensing capabilities (i.e., water, motion and vibration) are viewed as different domains. As such, mining a multidomain sequential patterns could be used to analyze behaviors of users. For example, to recognize one user behavior (i.e., cleaning behavior), one could mine multi-domain sequential patterns, which comprise of sequential patterns in water, motion and vibration domains, from those data logs generated by sensors with various sensing capabilities.

Though many sequential pattern mining algorithms are able to efficiently mine patterns in one domain, these algorithms cannot directly be applied in mining multi-domain sequential patterns. Existing sequential pattern mining algorithms suffer from poor performance when being applied in mining multi-domain sequential patterns across multiple domain sequence databases. Specifically, one could apply a sequential pattern mining algorithm in each individual domain and composite multi-domain sequential patterns by examining whether each element of sequential patterns occurs in the same time slot or not. For example, in Fig. 1, mining moving patterns, search patterns, and payment patterns in the corresponding sequence databases. Then, for each pattern mined in these three domains, we examine whether each element of these patterns occurs in the same time slot or not. However, the above method unavoidably leads to the poor performance in terms of efficiency and scalability. Note that mining all sequential patterns in each domain may not be necessary in forming multi-domain sequential patterns due to that the occurrence time slots of sequential patterns are not always the same. In addition, the extra-overhead is needed to integrate these sequential patterns across multiple domains into multi-domain sequential patterns.

In order to efficiently mine multi-domain sequential patterns, we propose algorithms IndividualMine and PropagatedMine. Specifically, algorithm IndividualMine consists of two phases: the mining phase and the checking phase. Specifically, in the mining phase, one could utilize one of existing sequential pattern mining algorithms to mine sequential patterns and derive the corresponding time instance set of each sequential pattern in each domain. In the checking phase, for each sequential pattern mined, we will enumerate all possible combinations to generate candidate multi-domain sequential patterns and then determine the support value of each candidate multi-domain sequential pattern. Note that mining sequential patterns in each domain is costly. Thus, algorithm PropagatedMine is proposed by only performing one sequential pattern mining in one domain sequence database. Then, those sequential patterns mined are organized as a lattice-like structure. Through the lattice-like structure, algorithm PropagatedMine is able to propagate time instance sets to other domain sequence databases and generate the corresponding sequential patterns. Our performance study shows that algorithm PropagatedMine outperforms algorithm IndividualMine. By propagating, algorithm PropagatedMine is able to

significantly reduce the execution time when it comes to mine multi-domain sequential patterns. Furthermore, through lattice-like structures in each domain sequence database, algorithm PropagatedMine is able to effectively mine sequential patterns across multiple sequence databases.

A significant amount of research efforts has been elaborated upon issues of mining sequential patterns [7][8]9][10][11][12]. We mentioned in passing that the authors in [1] formulated the problem of sequential pattern mining and proposed mining algorithms based on Apriori algorithm. By exploring a breadth first search and button-up algorithm, the authors in [13] developed algorithm GSP [13] for mining sequential patterns, whereas the authors in [14] devised algorithm SPADE, which is a depth first search and button-up algorithm with ID-list. The authors in [6][15] exploited the concept of projection in algorithms PrefixSpan and FreeSpan to reduce the volume of data for sequential pattern mining. To prevent the candidate generation, the authors in DISC-all [16] used a novel sequence comparison strategy. In addition, the authors in [4] developed several algorithms to mine multi-dimensional sequential patterns in which sequential patterns with some category attributes are discovered. To the best of our knowledge, prior works do not fully explore the mining capability for multi-domain sequential patterns, let alone proposing efficient algorithms tomine such sequential patterns. These features distinguish this paper from others. The contributions of this paper are twofold: (1) exploiting a novel and useful sequential patterns (i.e., multi-domain sequential patterns), and (2) devising algorithm PropagatedMine to efficiently mine multi-domain sequential patterns.

The remaining of the paper is organized as follows. In Section 2, some preliminaries are presented. Algorithms for mining multi-domain sequential patterns are developed in Section 3. Performance studies are conducted in Section 4. This paper concludes with Section 5.

2 Preliminary

In this section, we first describe some notations to facilitate the presentation of this paper. Then, the problem of mining sequential patterns across multiple domain sequence databases is defined.

Assume that each domain has its own set of items and a sequence in domain i is represented as $s_i =< X_{i1}, X_{i2},..., X_{il} >$, where X_{ij} is an itemset in the j th position of sequence s_i. Note that the length of time slots is depended on the log data collected. Therefore, a multi-domain sequence across k domain sequence databases is represented as $M = [s_1, s_2,..., s_k]^T$. A multi-domain sequence across k domains is further denoted as $\begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1b} \\ X_{21} & X_{22} & \cdots & X_{2b} \\ \vdots & \vdots & \vdots & \vdots \\ X_{k1} & X_{k2} & \cdots & X_{kb} \end{bmatrix}$, where each row $[X_{i1}, X_{i2},..., X_{ib}]$ is a sequence in domain i and each column $[X_{1j}, X_{2j},..., X_{aj}]^T$ for j = 1, 2, ..., b, is a vector of itemsets occurring in time slot j. Similar to the works in [1][6], the number of items in a sequence is called the length of the sequence. Since a multi-domain sequence consists of multiple sequences from various domains, we have the following definition for the length of a multi-domain sequence across k domains.

Definition 1: Length and Number of Elements: Let $M = [s_1, s_2,..., s_k]^T$ be a multi-domain sequence across k domains (abbreviated as k -*domain sequence*). The length of M, denoted as $|M|$, is formulated as $\max\left(|s_1|, |s_2|,..., |s_k|\right)$, meaning that the length of the longest sequence is viewed as the length of this multi-domain sequence. Furthermore, the number of elements of a multi-domain sequence, expressed by $e(M)$, is the number of itemsets appearing in each sequence of the multi-domain sequence.

For example, assume that $M = \begin{bmatrix} (a) & (b,c) & (b) \\ (1) & (2) & (1,2,3) \end{bmatrix}$. It can be verified that the length of M is $|M| =$ $\max\ (|(a)(b,c)(b)|, |(1)(2)(1,2,3)|) = \max(4,5) = 5$, and the number of element is 3 (i.e., $e(M)$= 3).

Table 1. An example of a multi-domain sequence database.

ID	Time instance sequences	Multi-domain sequences
S_1	$<(T_1)(T_2)(T_3)(T_4)>$	$\begin{bmatrix}(a) & (b,c) & (b,c,d) & (e)\\ (1,2) & (2,3) & (6) & (4,5)\end{bmatrix}$
S_2	$<(T_5)(T_7)(T_8)>$	$\begin{bmatrix}(a,b) & (b,c) & (c,e)\\ (1,3) & (2,4) & (8)\end{bmatrix}$
S_3	$<(T_{10})(T_{12})(T_{13})>$	$\begin{bmatrix}(a,e) & (h) & (g,j)\\ (1,6) & (5) & (9,10)\end{bmatrix}$
S_4	$<(T_{21})(T_{22})(T_{23})(T_{24})>$	$\begin{bmatrix}(a,b,f) & (d) & (b,c) & (e,f)\\ (1,2,5) & (7) & (2,3) & (4,5,6)\end{bmatrix}$

Definition 2: Containing Relation: Let $M=\begin{bmatrix}X_{11} & X_{12} & \cdots & X_{1b}\\ X_{21} & X_{22} & \cdots & X_{2b}\\ \vdots & \vdots & \vdots & \vdots\\ X_{a1} & X_{a2} & \cdots & X_{ab}\end{bmatrix}$ and $N=\begin{bmatrix}Y_{11} & Y_{12} & \cdots & Y_{1b'}\\ Y_{21} & Y_{22} & \cdots & Y_{2b'}\\ \vdots & \vdots & \vdots & \vdots\\ Y_{a1} & Y_{a2} & \cdots & Y_{ab'}\end{bmatrix}$ be two multi-domain sequences and $e(M)\le e(N)$. M is contained by N, denoted as $M\subseteq N$, if and only if there exists an integer list $1\le l_1<l_2<\ldots<l_b\le b'$, such that $X_{ij}\subseteq Y_{ilj}$, where $i=1,2,...,a$ and $j=1,2,...,b$.

Consider twomulti-domain sequences $M=\begin{bmatrix}(a) & (b,c)\\ (2) & (6)\end{bmatrix}$ and $N=\begin{bmatrix}(a) & (b,c) & (b,c,d) & (e)\\ (1,2) & (2,3) & (6) & (4,5)\end{bmatrix}$. It can be seen that N contains M since there exists an integer list 1 < 3, such that $(a)\subseteq(a)$, $(2)\subseteq(1,2)$, $(b,c)\subseteq(b,c,d)$, and $(6)\subseteq(6)$.

Based on the above descriptions of multi-domain sequences, a multi-domain sequence database is a set of multi-domain sequences. Consider an example of a multi-domain sequence database in Table 1, where there are four multi-domain sequences across two domains. Note that each multidomain sequence will have its own time instance sequence, an ordered list of time slots recording the occurrence time of the corresponding itemsets. For example, itemset $\begin{bmatrix}(a)\\ (1,2)\end{bmatrix}$ in multi-domain sequence S_1 occurs at time slot T_1.

Given a multi-domain sequence database MDB and a multi-domain sequence M, the support value of the multi-domain sequence M is the number of multi-domain sequences in MDB containing M.
Hence, we have $Support(M)=|\{N\mid N\in MDB \text{ and } M\subseteq N\}|$. Furthermore, we could extract the time-related information when counting the support of a multi-domain sequence (e.g., M). Therefore, we have the following definition:

Definition 3: Time Instance Set: The time instance set of *M* is defined as $TIS(M)=\{<\text{time instance sequences of sequence } N>:\text{the corresponding ordered integer list of sequence } N>\mid N\in MDB \text{ and } M\subseteq N\}|$. In addition, the size of time instance set of *M* is denoted as $|TIS(M)|=|\{N\mid N\in MDB \text{ and } M\subseteq N\}|$. Clearly, the value of $|TIS(M)|$ is equal to *Support*(*M*).

For example, assume that $M=\begin{bmatrix}(b)\\ (2)\end{bmatrix}$ is a multi-domain sequence. The support of *M* in multi-domain sequence database MDB shown in Table 1 is 3 since three multi-domain sequences (i.e., $S_1, S_2,$ and S_4) in *MDB* contain M. Also, we could have *TIS*(*M*) = {< $(T_1)(T_2)(T_3)(T_4)$: 2 >, < $(T_5)(T_7)(T_8)$: 2 >, < $(T_{21})(T_{22})(T_{23})(T_{24})$: 1 >, < $(T_{21})(T_{22})(T_{23})(T_{24})$: 3 >} and $|TIS(M)|=|\{S_1,S_2,S_4\}|=3$.

Table 2. An example of multiple domain sequence databases

Domain database D_1		
Id	Time instance sequences	Sequences
s_1	$<(T_1)(T_2)(T_3)(T_4)>$	$<(a)(b,c)(b,c,d)(e)>$
s_2	$<(T_5)(T_7)(T_8)>$	$<(a,b)(b,c)(c,e)>$
s_3	$<(T_{10})(T_{12})(T_{13})>$	$<(a,e)(h)(g,j)>$
s_4	$<(T_{21})(T_{22})(T_{23})(T_{24})>$	$<(a,b,f)(d)(b,c)(e,f)>$

Domain database D_2		
Id	Time instance sequences	Sequences
l_1	$<(T_{21})(T_{22})(T_{23})(T_{24})>$	<(1,2,5)(7)(2,3)(4,5,6)>
l_2	$<(T_{10})(T_{12})(T_{13})>$	<(1,6)(5)(9,10)>
l_3	$<(T_5)(T_7)(T_8)>$	<(1,3)(2,4)(8)>
l_4	$<(T_1)(T_2)(T_3)(T_4)>$	<(1,2)(2,3)(6)(4,5)>

Given a minimum support threshold δ, a multi-domain sequence database MDB, and a multi-domain sequence M, M is a frequent multi-domain sequence in MDB, if and only if $Support(M) \geq \delta$. For example, given a multi-domain sequence database MDB depicted in Table 1, and the minimum support δ = 3, the multi-domain sequential patterns are $\begin{bmatrix}(a)\\(1)\end{bmatrix}$, $\begin{bmatrix}(b)\\(2)\end{bmatrix}$, $\begin{bmatrix}(b)\\(3)\end{bmatrix}$, $\begin{bmatrix}(c)\\(2)\end{bmatrix}$, $\begin{bmatrix}(b,c)\\(2)\end{bmatrix}$, $\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}$, $\begin{bmatrix}(a) & (c)\\(1) & (2)\end{bmatrix}$, and $\begin{bmatrix}(a) & (b,c)\\(1) & (2)\end{bmatrix}$.

Problem of mining multi-domain sequential patterns: To facilitate the presentation of multi-domain sequential patterns, Table 1 is used to illustrate an example of a multi-domain sequence data-base and then we should determine multi-domain sequential patterns from a multi-domain sequence database given. In reality, however, each domain has its own sequence database in which each sequence is generated by sorting the occurrence time instances. Consider Table 1 as an example, where a multi-domain sequence database is shown in Table 2. To derive a multi-domain sequence database, one should join these sequences by time instances as joining keys. Since performing join operations across multiple sequence databases is costly, deriving a multi-domain sequence database is hard to achieve. As such, the problem of mining multi-domain sequential patterns is that give a set of sequence databases across multiple domains, we should mine multi-domain sequential patterns.

3 Multi-domain Sequential Pattern Mining

In this section, we develop two algorithms to mine sequential patterns across multiple domain sequence databases. Specifically, in Section 3.1, algorithm IndividualMine is devised. Then, by propagating, algorithm PropagatedMine is developed to efficiently mine sequential patterns across other sequence databases.

3.1 Algorithm IndividualMine

Algorithm IndividualMine consists of two phases: the mining phase and the checking phase. Specifically, in the mining phase, one could utilize one of existing sequential pattern mining algorithms to mine sequential patterns and derive the corresponding time instance set of each sequential pattern in each domain. In the checking phase, for each sequential pattern in each domain, we will enumerate all possible combinations to generate candidate multi-domain sequential patterns and then determine support values for each candidate multi-domain sequential pattern. If a candidate multi-domain sequential pattern has its support value larger than or equal to the threshold value defined (i.e., minimum support), this candidate multi-domain sequential pattern will become a multi-domain sequential pattern. The overview of algorithm IndividualMine is shown in Fig. 2.

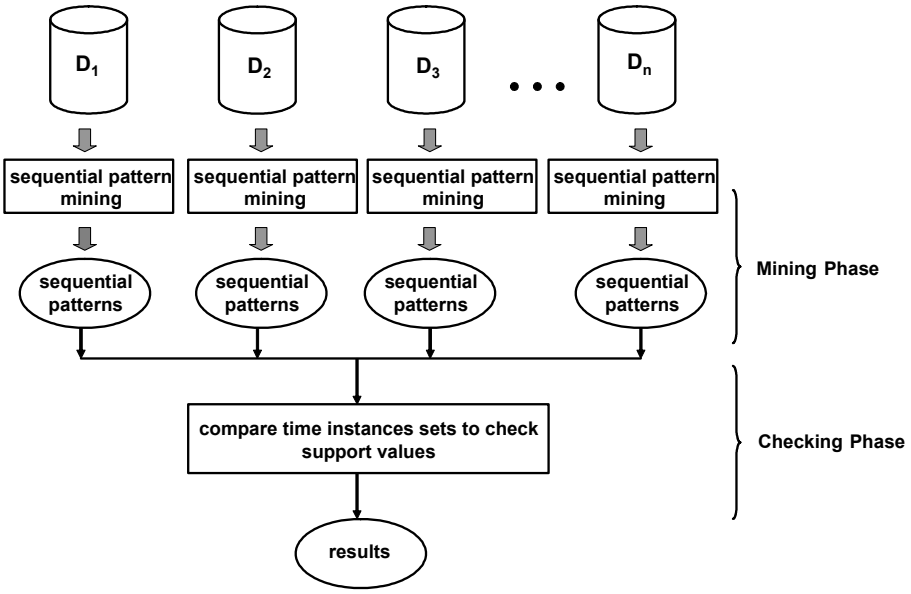

Fig. 2. Overview of algorithm IndividualMine.

As described before, in the mining phase of algorithm IndividualMine, existing sequential pattern mining algorithms are performed in each domain sequence database. By combining sequential patterns mined from domain sequence databases, we could generate candidate multi-domain sequential patterns. Then, in the checking phase, we will iteratively determine whether sequential patterns mined from domain sequence databases are able to be formed candidate multi-domain sequential patterns or not. Through counting the support values of these candidate multi-domain sequential patterns, one could derive multi-domain sequential patterns if their support values are larger than minimum support. Explicitly, assume that we have a multi-domain sequence database as $\{D_1, D_2, ...D_k\}$ and SP_i denotes the set of multi-domain sequential patterns across i domain sequence databases (i.e., $\{D_1, D_2, ...D_i\}$). In the beginning, we will obtain sequential patterns in a starting domain (i.e., domain D_1). Those sequential patterns in the starting domain are viewed as multi-domain sequential patterns across domain D_1 (referred to as SP_1). Then, for each patterns in SP_1, we will generate candidate multi-domain sequential patterns across two domains (i.e., D_1 and D_2) by combining sequential patterns mined in domain sequence database D_2. For example, we could have $\begin{bmatrix} p \\ q \end{bmatrix}$, where $p \in SP_1, q$ is a sequential pattern of D_2 and both patterns have the same number of elements(i.e., $e(p) = e(q)$). After generating candidate multi-domain sequential patterns across two sequence databases, support values of these multi-domain sequential patterns are evaluated by the number of intersections in their time instance sets. For example, suppose that we have a candidate multi-domain sequential pattern as $\begin{bmatrix} (a) & (b) \\ (1) & (2) \end{bmatrix}$ and the time instance sets of $< (a)(b) >$ and $< (1)(2) >$ are $\{< (T_1)(T_2)(T_3)(T_4) : 1, 2 >, < (T_1)(T_2)(T_3)(T_4) : 1, 3 >, < (T_5)(T_7)(T_8) : 1, 2 >, < (T_{21})(T_{22})(T_{23})(T_{24}) : 1, 3 >\}$ and $\{< (T_1)(T_2)(T_3)(T_4) : 1, 2 >, < (T_5)(T_7)(T_8) : 1, 2 >, < (T_{21})(T_{22})(T_{23})(T_{24}) : 1, 3 >\}$, respectively. It can be verified that $TIS(\begin{bmatrix} (a) & (b) \\ (1) & (2) \end{bmatrix}) = \{< (T_1)(T_2)(T_3)(T_4) : 1, 2 >, < (T_5)(T_7)(T_8) : 1, 2 >, < (T_{21})(T_{22})(T_{23})(T_{24}) : 1, 3 >\}$. Therefore, $Support(\begin{bmatrix} (a) & (b) \\ (1) & (2) \end{bmatrix}) = |TIS(\begin{bmatrix} (a) & (b) \\ (1) & (2) \end{bmatrix})| = 3$. Given a minimum support 2, we could have $\begin{bmatrix} (a) & (b) \\ (1) & (2) \end{bmatrix}$ as a multi-domain sequential pattern across two domain sequence databases since the support of this sequential pattern is larger than the minimum support. By combining patterns in SP_k and patterns in domain sequence database D_{k+1}, we could derive candidate multi-domain sequential patterns across $k+1$ domain sequence databases.

Algorithm IndividualMine:

Input: Multi-domain sequence database with n domains, $D_1, D_2, ...D_n$ and the minimum support δ.

Output: Multi-domain sequential patterns across n domains.

Begin

Apply sequential pattern mining on each domain $D_i, i = 1,2,...,n$.

Let SP_1 be the set of sequential patterns mined in D_1.

For each domain $D_i, i = 2,3,...,n$

For each $P \in SP_{i-1}$

For each sequential pattern n of D_i, and $e(q) = e(P)$

If $| TIS(P) \cap TIS(q) | \geq \delta$ **Then** append $\begin{bmatrix} P \\ q \end{bmatrix}$ to SP_i

Output= SP_n.

End

Algorithm IndividualMine performs sequential pattern mining algorithms in each domain sequence database. Then, these sequential patterns are merged together to form candidate multi-domain sequential patterns. However, those sequential patterns are not always able to generate multi-domain sequences due to that the occurrence times for each itemset in individual sequential patterns are not the same. Thus, the effort of generating candidate multi-domain sequential patterns could be reduced. Therefore, we develop algorithm PropagatedMine in which only one domain sequence database needs to perform sequential pattern mining. Furthermore, by propagating time instance sets to other domain sequence databases, algorithm PropagatedMine is able to efficiently mine sequential patterns that are likely to form multi-domain sequential patterns.

3.2 Algorithm PropagatedMine

Algorithm PropagatedMine is designed to reduce the cost of both mining sequential patterns in each domain and reducing the number of candidate multi-domain sequential patterns. Moreover, sequential patterns mined in each domain are not necessary to form multi-domain sequential patterns. Hence, algorithm PropagatedMine only performs sequential pattern mining in one domain (referred to as a starting domain) and then propagates time instance sets of sequential patterns mined to other domains. By propagating time instance sets to other domain sequence databases, we could only extract those sequences having the same time instances to generate candidate multi-domain sequential patterns. Algorithm PropagatedMine will iteratively propagate time instance sets of sequential patterns mined to next domain sequence databases until all domains have been mined. Specifically, algorithm PropagatedMine consists of two phases: the mining phase and the deriving phase. Fig. 3 shows the overview of algorithm PropagatedMine.

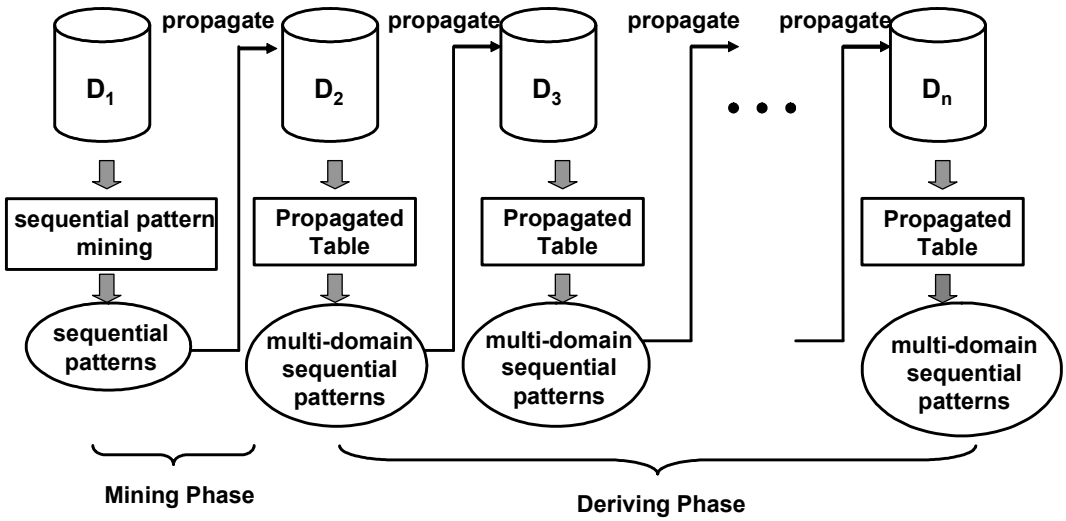

Fig. 3. Overview of algorithm PropagatedMine.

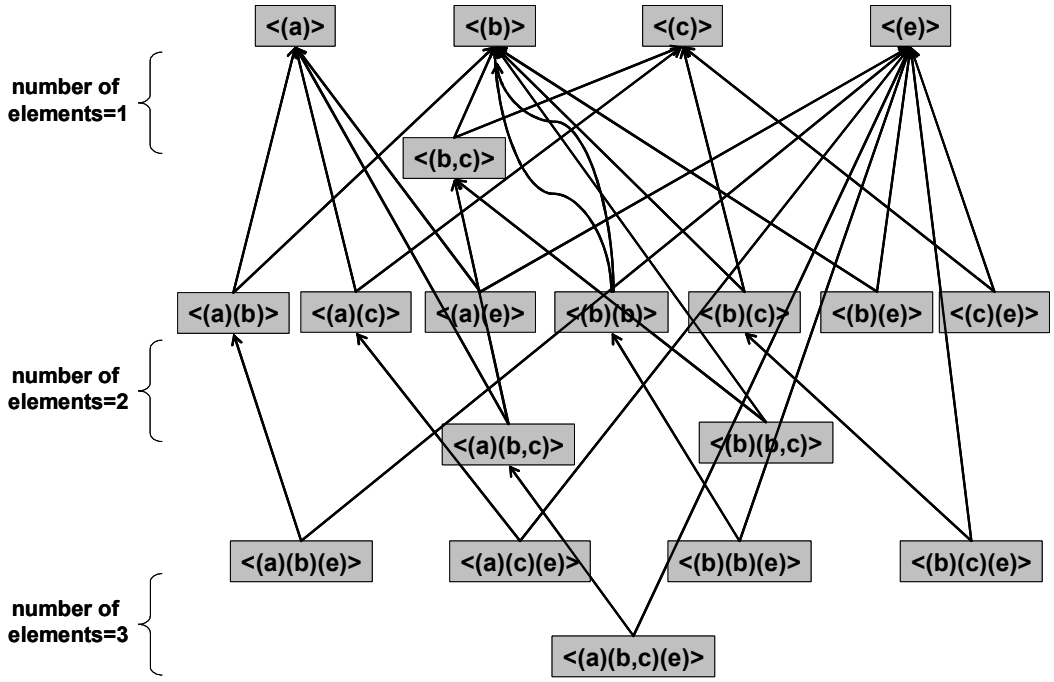

Fig. 4. A lattice-like structure for sequential patterns in a starting domain (i.e., D_1 in Table 2).

Same as in Algorithm IndividualMine, in the mining phase, algorithm PropagatedMine utilizes existing sequential pattern mining algorithms to discover sequential patterns in a starting domain (e.g., D_1) and then propagates time instance sets of sequential patterns mined to other domains. Note that once sequential patterns are mined in the starting domain sequence database, these sequential patterns in the starting domain provide guidelines for mining multi-domain sequential patterns across multiple domain sequence databases in that the number of elements and the length of multi-domain sequence databases are constrained by sequential patterns mined in the starting domain. Therefore, sequential patterns mined in the starting domain are represented as the lattice-like graph structure to facilitate the generation of candidate multi-domain sequential patterns. For example, assume that the starting domain sequence database is set to D_1 in Table 2 and then sequential patterns are found by existing sequential pattern algorithms. Those sequential patterns mined are represented as a lattice-like structure shown in Fig. 4, where each node represents a sequential pattern, the linkages of nodes (standing for *intra-domain links*) represent containing relationships and nodes are ordered by the number of their elements. In Fig. 4, those nodes having thesame number of elements are further arranged level-by-level according to their sequence length. Explicitly, it can be seen in Fig. 4 for the nodes with their number of elements is 1, these nodes are put level-by-level in increasing order of length of sequences. For example, $< (b,c) >$ is below the nodes whose length of sequence is 1 (e.g., $< (b) >$). As mentioned above, the lattice-like structure is used as a guideline for propagating time instance sets of sequential patterns to other domains. By propagating time instance sets, algorithm PropagatedMine in the deriving phase extract those sequences with their occurrence times the same as time instance sets propagated. Thus, for each time instance set propagated, we could build the corresponding propagated table defined as follows:

Definition 4 (Propagated table): Let *M* be a multi-domain sequential pattern across *k* domain sequence databases with $TIS(M) = \{< TS_1 : l_1 >, < TS_2 : l_2 >, ..., < TS_f : l_f >\}$, where TS_i is a time instance sequence and l_i is the corresponding integer list. Assume that domain $D_t = \{s_1, s_2, ..., s_m\}$, where $s_i =< X_1^i, X_2^i, ..., X_{e(t_i)}^i >$ and each sequence s_i has the corresponding time instance sequence, denoted as TS_{s_i}. When propagating time instance sets of *M* to domain D_t, we could have a propagated table defined as $D_t \|_M = \{X_{l_j}^i \mid \exists TS_{s_i}$ and $TS_j \ni (TS_{s_i} = TS_j)\}$.

For example, in Table 2, by propagating *TIS*(< (*b*) >) to sequence database D_2, we could have the

Table 3. An example of propagated table $D_2 \|_{<(b)>}$.

Time instance sequences Items	Items
<(T1)(T2)(T3)(T4)>	(2,3)
<(T1)(T2)(T3)(T4)>	(6)
<(T5)(T6)(T7)>	(1,3)
<(T5)(T6)(T7)>	(2,4)
<(T21)(T22)(T23)(T24)>	(1,2,5)
<(T21)(T22)(T23)(T24)>	(2,3)

propagated table $D_2 \|_{<(b)>}$ shown in Table 3. After obtaining propagated tables, we could mine frequent itemsets by association rule mining algorithms [17]. Then, those frequent itemsets could be combined by the corresponding patterns propagated to generate multi-domain sequential patterns. From the above example, given a minimum support 3, we can easily obtain $\begin{bmatrix}(b)\\(2)\end{bmatrix}$ and $\begin{bmatrix}(b)\\(3)\end{bmatrix}$ as multi-domain sequential patterns across 2 domain sequence databases, where (2) and (3) are the frequent items of $D_2 \|_{<(b)>}$.

Property of propagated table: Suppose that $P \in SP_k$ and β is a itemset in domain sequence database D_{k+1}. A multi-domain sequential pattern $\begin{bmatrix}P\\\beta\end{bmatrix}$ is a sequential pattern across (k + 1) domain sequence databases (i.e., $\{D_1, D_2, ..., D_{k+1}\}$) with the minimum support δ if and only if β is a frequent itemset in propagated table $D_{k+1} \|_P$ with the same minimum support δ. Clearly, we could have $TIS(\begin{bmatrix}P\\\beta\end{bmatrix}) = TIS(P) \cap TIS(\beta)$.

Property of anti-monotone: Given multi-domain sequence databases $MDB = \{D_1, D_2, ...D_k\}$ and a multi-domain sequence $M = [s_1 s_2 ... s_k]^T$, multi-domain sequences contained by M are frequent, if and only if M is a multi-domain sequential pattern of MDB. This property is also valid when it comes to mining multi-domain sequential patterns. Based on the property of anti-monotone, algorithm PropagatedMine generates candidate multi-domain sequential patterns in a level-by-level manner.

Note that through the lattice-like structure in the starting domain, algorithm PropagatedMine only needs to propagate time instance sets of sequential patterns with their length to be 1 to other domains. In other words, only time instance sets of the top level nodes (referred to as atomic patterns) are propagated. This is due to that sequential patterns in the propagated domain could use time instance sets of the upper level nodes to determine their support values. Therefore, in the propagated domain, sequential patterns are also generated level-by-level according to the number of elements of sequences. The detailed steps for deriving multi-domain sequential patterns are described as follows:

Step 1: Derive atomic patterns across (k + 1) **domains:**

Let SP_k be the set of multi-domain sequential patterns across k domain sequence databases. By propagating each atomic patterns in SP_k, we could derive the corresponding frequent itemsets from propagated tables. Then, those frequent itemsets mined from propagated tables are merged with atomic patterns in SP_k to derive atomic patterns across (k + 1) domains. Consider two domain sequence databases in Table 2 as an example. Since sequential patterns of domain D_1 are represented as a lattice-like structure, we should first derive atomic patterns in sequence database D_2. Specifically, in Fig. 5, time instance sets of atomic patterns in sequence database D_1 (i.e., the top-level nodes) are propagated to sequence database D_2 2. From the propagated tables of each atomic pattern, atomic patterns are easily obtained. For each atomic pattern in D_1, there is a *inter-domain link* representing that these two patterns are able to form multi-domain sequential patterns. Consequently, we have $\begin{bmatrix}(a)\\(1)\end{bmatrix}$, $\begin{bmatrix}(b)\\(2)\end{bmatrix}$, $\begin{bmatrix}(b)\\(3)\end{bmatrix}$, and $\begin{bmatrix}(c)\\(2)\end{bmatrix}$ in the above example.

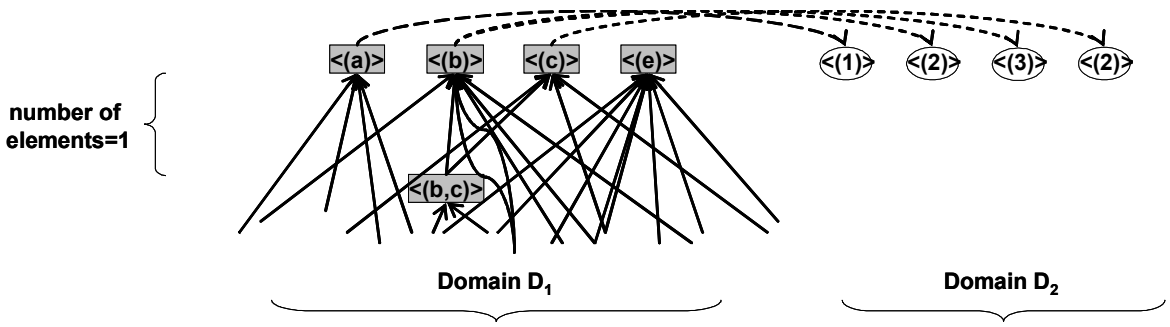

Fig. 5. An example of generating atomic patterns in domain D2.

Step 2: Derive (k + 1)**-domain sequential patterns with their number of elements being one:**

In this step, we will derive (k + 1)-domain sequential patterns with the number of elements to be one. Assume that k-domain sequential pattern Q across k domain sequence databases (i.e., $\{D_1, D_2, ...D_k\}$) and the number of elements in Q is 1. Through the lattice-like structure for each domain, one could follow intra-domain links to find those atomic patterns that are the components of k-domain sequential pattern Q. Thus, we could use the lattice-like structure in sequence database D_k and extract atomic patterns of a sequential pattern in sequence database D_{k+1} from k-domain sequential pattern Q. By travelling inter-domain links in the lattice-like structure in domain D_k, those corresponding atomic patterns in domain D_{k+1} are found. Hence, in light of these atomic patterns found in domain D_{k+1}, sequential patterns in sequence database D_{k+1} are generated. Suppose that we have a k-domain sequential pattern in Q as $X_1 \cup X_2 \cup ... \cup X_l$, where X_j is the jth atomic pattern in Q and l is the length of sequential pattern Q. It could be verified that a multi-domain sequential pattern P is the union of $\begin{bmatrix} X_i \\ y_i \end{bmatrix}$, where y_i is the corresponding atomic pattern in domain (k + 1) and $\begin{bmatrix} X_i \\ y_i \end{bmatrix}$ is the (k + 1)-domain atomic pattern mined in Step 1, for i = 1, 2, ..., l. For example, let Q =< (b, c) >, a sequential pattern with $e(Q)$ = 1 in domain D_1 of Table 2. Through the intradomain links, we can find atomic patterns that are components of Q (i.e., < (b) > and < (c) >). In Fig. 6 following inter-domain links of < (b) > and < (c) >, we could obtain the atomic patterns in domain D_2 (i.e., < (2) > and < (3) >). Consequently, two possible unions of P are generated (i.e., $\begin{bmatrix} (a) \\ (1) \end{bmatrix} \cup \begin{bmatrix} (c) \\ (2) \end{bmatrix} = \begin{bmatrix} (b,c) \\ (2) \end{bmatrix}$ and $\begin{bmatrix} (b) \\ (3) \end{bmatrix} \cup \begin{bmatrix} (c) \\ (2) \end{bmatrix} = \begin{bmatrix} (b,c) \\ (2,3) \end{bmatrix}$). Once we have the possible candidate multi-domain sequential patterns, support values of these patterns are examined by checking their time instance sets (i.e., $Support(P) = |\, TIS(P) \,| = |\, \bigcap_{i=1}^{l} TIS(\begin{bmatrix} X_i \\ y_i \end{bmatrix}) \,|$). Given a minimum support 3, since the

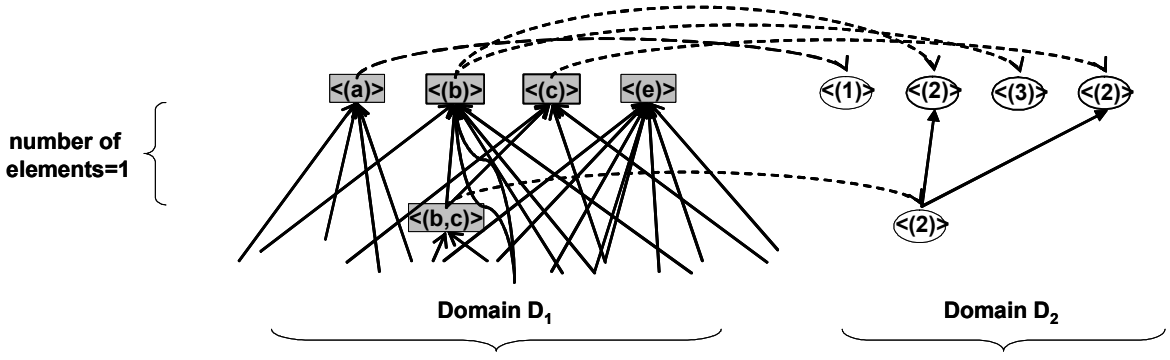

Fig. 6. An example of generating sequential patterns whose number of elements is 1 in domain D_2.

support values of $\begin{bmatrix}(b,c)\\(2)\end{bmatrix}$ and $\begin{bmatrix}(b,c)\\(2,3)\end{bmatrix}$ are 3 and 2, respectively, $\begin{bmatrix}(b,c)\\(2)\end{bmatrix}$ is a frequent multi-domain sequence. Thus, the lattice-like structure in domain D_2 contains < (2) > and inter-links are built between lattice-like structures in domain D_1 and that in domain D_2.

Step 3: Derive (k + 1)**-domain sequential patterns with their number of elements larger than one:**

After generating those (k + 1)-domain sequential patterns with their number of elements being one, algorithm PropagatedMine could further generate candidate (k+1)-domain sequential patterns with their number of elements larger than one in a level-by-level manner. In order to generate (k+1)-domain sequential patterns, algorithm PropagatedMine will refer the lattice-like structure in the last domain propagated (i.e., domain D_k in our example). In the lattice-like structure of domain D_k, algorithm PropagatedMine first identify those patterns with their numbers of elements to be 2. Through the intra-domain links in the lattice-like structure of D_k, those frequent patterns in the upper levels are found. Following inter-domain links of these upper level patterns, the corresponding upper level patterns in the lattice-like structure of domain D_{k+1} are identified. Before deriving (k + 1)-domain sequential patterns, those sequential patterns identified in the lattice-like structure should further be verified whether these patterns should be merge or not. The verification should be made by comparing their time instance sets. Thus, we have the following definition:

Definition 5 ($\cap_<$ operation of TIS): Let M and N be two multi-domain sequences, where $e(M) = e(n) = 1$, $TIS(M) = \{< S_i : l_i >\}$ and $TIS(N) = \{< T_i : m_i >\}$. $TIS(M) \cap_< TIS(N)$ is defined as $\{< S_i : m_i, l_i >\}$ such that $S_i = T_i$ and $l_i < m_i$, meaning that these two multi-domain sequences (i.e., M and N) could be merged together as a multi-domain sequence since their time instance sets obey a time sequential order.

For example, given $M = \begin{bmatrix}(a)\\(1)\end{bmatrix}$, $N = \begin{bmatrix}(b,c)\\(2)\end{bmatrix}$ and the multi-domain sequence database in Table 2 with a minimum support as 3, it can be verified that $TIS(M) = \{< (T_1)(T_2)(T_3)(T_4) : 1 >, < (T_5)(T_7)(T_8) : 1 >, < (T_{10})(T_{12})(T_{13}) : 1 >\}$, $(T_{21})(T_{22})(T_{23})(T_{24}) : 1 >\}$, $TIS(N) = \{< (T_1)(T_2)(T_3)(T_4) : 2 >, < (T_5)(T_7)(T_8) : 2$, $(T_{21})(T_{22})(T_{23})(T_{24}) : 3 >\}$, and $TIS(M) \cap_< TIS(N) = \{< (T_1)(T_2)(T_3)(T_4) : 1, 2 >, < (T_5)(T_7)(T_8) : 1, 2 >$, $< (T_{21})(T_{22})(T_{23})(T_{24}) : 1, 3 >\}$. Since $TIS(M) \cap_< TIS(N)$, we could further merge these sequential patterns into $\begin{bmatrix}(a)(b,c)\\(1)\ \ (2)\end{bmatrix}$. In light of Definition 5, we could determine whether two patterns should be merged or not. Assume that pattern $P \in SP_k, e(P) > 1$ and $P =< S_1, S_2, ..., S_{e(p)} >$, where $S_i \in SP_k$ and $e(S_i) = 1$. By traveling intra-domain links and inter-domain links among lattice-like structures across k domains, we could obtain subsets of SP_{k+1} as $\{\begin{bmatrix}(S_1)\\(T_1)\end{bmatrix}, \begin{bmatrix}(S_2)\\(T_2)\end{bmatrix}, \ldots, \begin{bmatrix}(S_{e(p)})\\(T_{e(p)})\end{bmatrix}\}$, where T_i is an itemset, for i = 1, 2, ..., $e(p)$. If those subsets have a relationship as $TIS(\begin{bmatrix}(S_1)\\(T_1)\end{bmatrix}) \cap_< TIS(\begin{bmatrix}(S_2)\\(T_2)\end{bmatrix}) \cap_< \ldots \cap_< TIS(\begin{bmatrix}(S_{e(p)})\\(T_{e(p)})\end{bmatrix}$, $P' = \begin{bmatrix}S_1 & S_2 & \ldots & S_{e(p)}\\T_1 & T_2 & \ldots & T_{e(p)}\end{bmatrix}$ is generated. The corresponding time instance set is derived by $TIS(P') = TIS(\begin{bmatrix}(S_1)\\(T_1)\end{bmatrix}) \cap TIS(\begin{bmatrix}(S_2)\\(T_2)\end{bmatrix}) \cap \ldots \cap TIS(\begin{bmatrix}(S_{e(p)})\\(T_{e(p)})\end{bmatrix})$. As such, we could verify whether P' is frequent or not by its support value (i.e., $|TIS(P')|$). Consider an example pattern $P =< (a)(b, c) >$ in Fig. 7. By intra-domain links and inter-domain links, we have $\begin{bmatrix}(a)\\(1)\end{bmatrix} \cap_< \begin{bmatrix}(b,c)\\(2)\end{bmatrix}$. Therefore, $P' = \begin{bmatrix}(a)(b,c)\\(1)(2)\end{bmatrix}$ is generated.

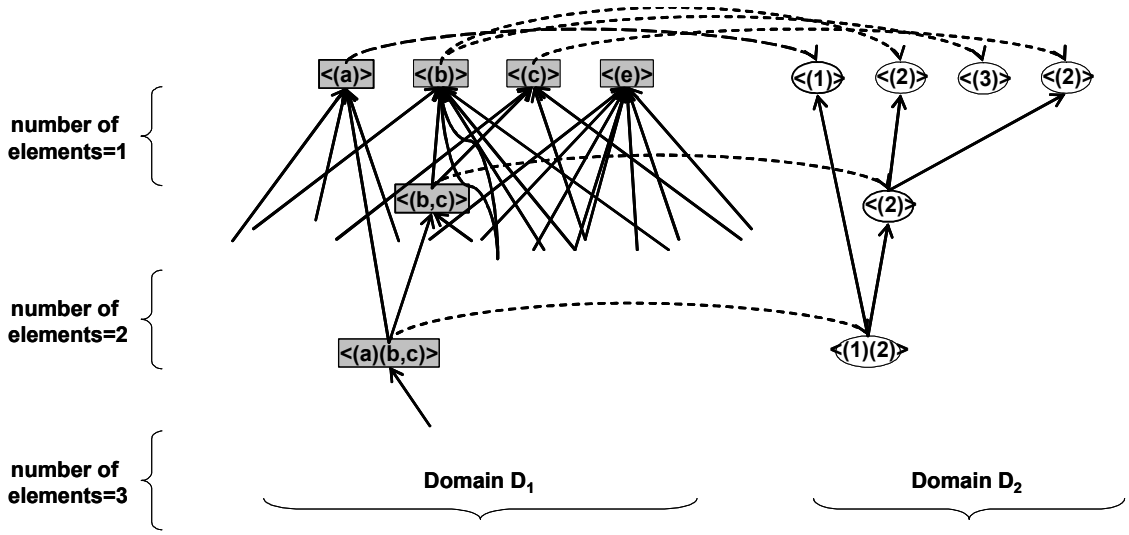

Fig. 7. An example of generating sequential patterns with their number of elements larger than 1 in domain D_2.

Through the above steps, we could derive multi-domain sequential patterns across $k + 1$ domain sequence databases from k-domain sequential patterns. Algorithm PropagatedMine iteratively repeats the above three steps until all domain sequence databases are propagated.

Algorithm PropagatedMine:

Input: Multi-domain sequence database with n domains, $D_1, D_2, ..., D_n$, and the minimum support δ.

Output: Multi-domain sequential patterns with n domains.

Begin

Apply sequential pattern mining on D_1.

Let SP_1 be the set of sequential patterns mined in D_1.

For each domain D_i, $i = 2, 3, ..., n$

For each $P \in SP_{i-1}$

If $|P| = 1$ **Then Begin**

Construct Propagation Table $D_i \|_P$.

Find frequent items in $D_i \|_P$ with minimum support δ.

Let FI be the set of frequent items in $D_i \|_P$.

For each $q \in FI$

Append $\begin{bmatrix} P \\ q \end{bmatrix}$ to SP_i.

Let $TIS(\begin{bmatrix} P \\ q \end{bmatrix}) = TIS(P) \cap TIS(q)$.

End

If $e(\underline{P}) = 1$ **Then Begin**

Compose $\begin{bmatrix} P \\ q \end{bmatrix}$ with $e(\begin{bmatrix} P \\ q \end{bmatrix}) = 1$.

If $Support(\begin{bmatrix} P \\ q \end{bmatrix}) \geq \delta$) **Then** append $\begin{bmatrix} P \\ q \end{bmatrix}$ to SP_i

End

If $e(P) > 1$ **Then Begin**

Compose $\begin{bmatrix} P \\ q \end{bmatrix}$ with $e(\begin{bmatrix} P \\ q \end{bmatrix}) > 1$.

If $Support(\begin{bmatrix} P \\ q \end{bmatrix}) \geq \delta$ **Then** append $\begin{bmatrix} P \\ q \end{bmatrix}$ to SP_i

End

Output= SP_n .

End

4 Performance Study

Our experiments run on a 1.8GHz Athlon PC with 1G main memory, and both algorithms IndividualMine and PropagatedMine are implemented in Java. For mining sequential patterns in one domain sequence database, we implement algorithm PrefixSpan [6]. The performance of algorithms IndividualMine and PropagatedMine is measured in terms of the execution time. The datasets were generated by data generator in [1] with slightly modifications that includes multiple domain sequence databases. Table 4 depicts the parameters used to represent the characteristic of dataset generated. For example, a dataset M5D10kC10T5S4 means that there are 5 domains, each domain sequence database consists of 10,000 sequences, where the average number of elements in a sequence is 10, the average number of items in an element is 5 and the average length of maximal sequential patterns is 4.

We first investigate the performance of algorithms IndividualMine and PropagatedMine with the value of the minimum support varied. The dataset is M5D10kC8T8S8 and the values of minimum support is ranged from 2.5% to 10%. The execution time of these two algorithms is shown in Fig. 8. With the smaller minimum support, the number of sequential patterns will be larger, thereby increasing the execution time of both algorithms. Since algorithm IndividualMine needs to perform sequential pattern mining in each domain sequence database, the execution time of algorithm IndividualMine is larger than that of algorithm PropagatedMine. Next, we conduct experiments with the number of domain varied. The number of domain is varied from 2 to 5. For each domain sequence database, the setting of datasets is D10kC8T8S8. The minimum support is set to 2.5%. The performance is shown in Fig. 9. Clearly, when the number of domains increases, the execution time of both algorithms IndividualMine and PropagatedMine increases. It is expected that with a larger number of domains, algorithm IndividualMine performs worse than algorithm PropagatedMine since sequential pattern mining algorithms are performed at each domain sequence database.

The experiments of varying the number of sequences is now evaluated. The numbers of sequences are set to 1000, 4000, 7000 and 10000, respectively. The setting of the other parameters is fixed to M5C8T8S8 and the minimum support is 2.5%. As can be seen in Fig. 10, the execution time of both algorithms increases with the number of sequences. Furthermore, algorithm PropagatedMine outperforms algorithm IndividualMine due to the same reason that algorithm IndividualMine individually executes sequential pattern mining algorithms in each domain sequence database. As the number of sequences increases, the performance of mining sequential patterns is worst.

Table 4. Parameters used for the data generator

Parameter	Descriptions
M	number of domains
D	number of sequences
C	average number of elements within a sequence
T	average number of items within an element
S	average length of maximal potentially sequential patterns

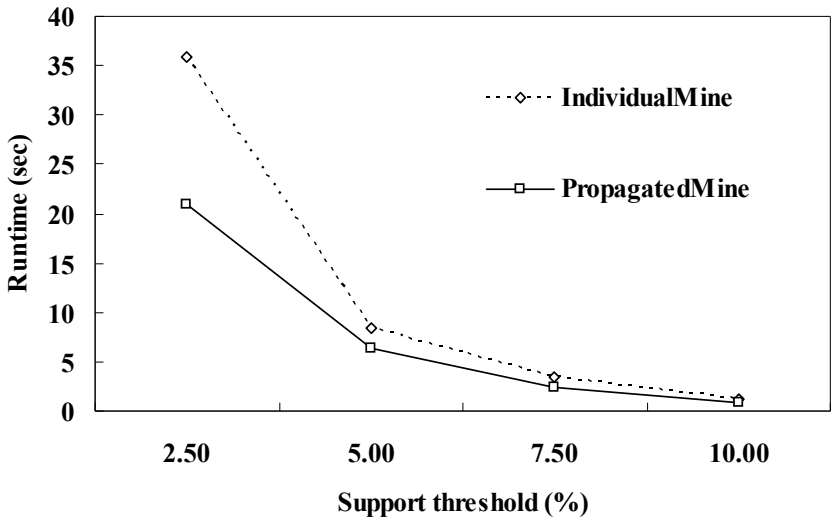

Fig. 8. The execution time of algorithms IndividualMine and PropagatedMine with various minimum support values.

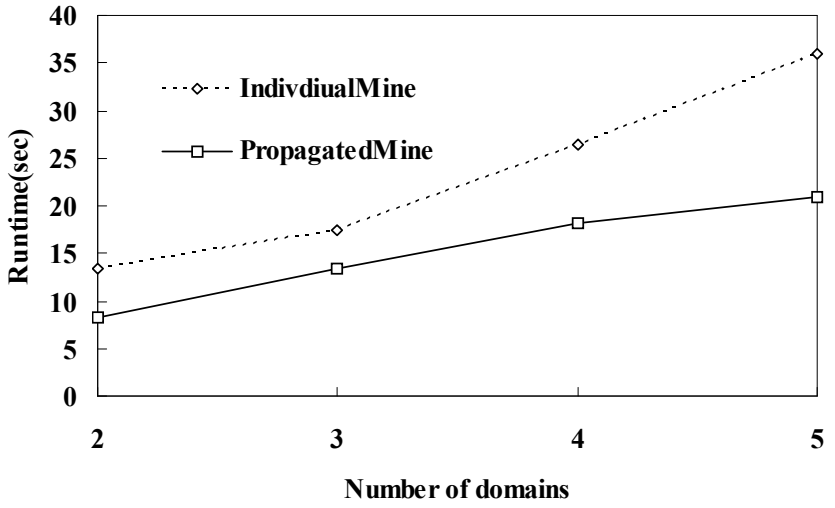

Fig. 9. The performance of algorithms IndividualMine and PropagatedMine with the number of domain varied.

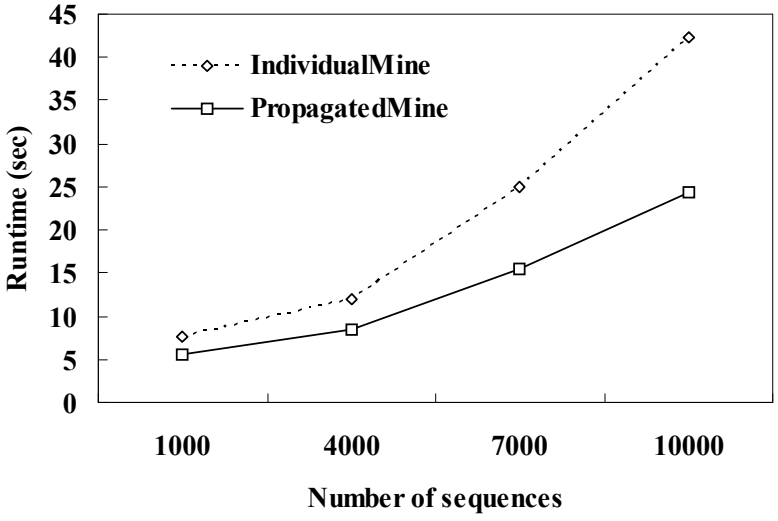

Fig. 10. The performance of algorithms IndividualMine and PropagatedMine with the number of sequences varied.

5 Conclusions

In this paper, we explored multi-domain sequential patterns across multiple domain sequence databases. Algorithms IndividualMine and PropagatedMine for mining multi-domain sequential patterns are developed. Specifically, in algorithm IndividualMine, each domain individually performs sequential pattern mining and then by checking the time instances, sequential patterns in each domain are merged as multi-domain sequential patterns. In order to reduce the mining cost in each domain sequence database, algorithm PropagatedMine first mines sequential patterns in a starting domain sequence database. Furthermore, algorithm PropagatedMine uses lattice-like structures to store these sequential patterns. In light of lattice-like structures, algorithm PropagatedMine is able to propagate time instance sets of sequential patterns mined to other domains and discovers multidomain sequential patterns in a level-by-level manner. A comprehensive performance study was conducted. Experimental results show that by propagating time instance sets of sequential patterns mined to other domains, algorithm PropagatedMine is able to more efficiently mine multi-domain sequential patterns than algorithm IndividualMine. In the future, we will devise an optimal propagation order to further improve the performance of algorithm PropagatedMine.

References

[1] R. Agrawal and R. Srikant, "Mining Sequential Patterns", *Proceedings of the 1995 IEEE International Conference on Data Engineering (ICDE)*, pp. 3–14, 1995.

[2] H. Cheng, X. Yan, and J. Han, "IncSpan: Incremental Mining of Sequential Patterns in Large Database," *Proceedings of the 2004 ACMInternational Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 527–532, 2004.

[3] N. Lesh, M. J. Zaki, and M. Ogihara "Mining Features for Sequence Classification," *Proceedings of the 1999 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 342–346, 1999.

[4] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, "Multi-Dimensional Sequential Pattern Mining," *Proceedings of the 2001 ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 81–88, 2001.

[5] P.-Y. Rolland, "FlExPat: Flexible Extraction of Sequential Patterns," *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM)*, pp. 481–488, 2001.

[6] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U.Dayal, and M. Hsu, "PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth," *Proceedings of the 2001 IEEE International Conference on Data Engineering (ICDE)*, pp. 215–224, 2001.

[7] G. Chen, X. Wu, and X. Zhu, "Sequential Pattern Mining in Multiple Streams," *Proceedings of the 2005 IEEE International Conference on Data Mining (ICDM)*, pp. 585–588, 2005.

[8] M. N. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints," *Proceedings of the 1999 International Conference on Very Large Data Bases (VLDB)*, pp. 223–234, 1999.

[9] P. Tzvetkov, X. Yan, and J. Han. TSP, "Mining Top-K Closed Sequential Patterns," *Knowledge Information System*, Vol.7, No.4, pp.438–457, 2005.

[10] J.Wang and J. Han. BIDE, " Efficient Mining of Frequent Closed Sequences," *Proceedings of the 2004 IEEE International Conference on Data Engineering (ICDE)*, pp. 79–90, 2004.

[11] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Databases," *Proceedings of the 2003 SIAM International Conference on Data Mining (SDM)*, 2003.

[12] J. Yang, W. Wang, P. S. Yu, and J. Han, "Mining Long Sequential Patterns in A Noisy Environment," *Proceedings of the 2002 ACM International Conference on Management of Data (SIGMOD)*, pp. 406–417, 2002.

[13] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proceedings of the 1996 International Conference on Extending Database Technology (EDBT)*, pp. 3–17, 1996.

[14] M. J. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences,” *Machine Learning*, Vol.42, No.1/2, pp. 31–60, 2001.

[15] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu, “FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining,” *Proceedings of the 2000 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 355–359, 2000.

[16] D.-Y. Chiu, Y.-H. Wu, and A. L. P. Chen, “An Efficient Algorithm for Mining Frequent Sequences by a New Strategy without Support Counting,” *Proceedings of the 2004 IEEE International Conference on Data Engineering (ICDE)*, pp. 375–386, 2004.

[17] R. Agrawal, T. Imielinski, and A. Swami, “Mining Associations between Sets of Items in Massive Databases,” *SIGMOD*, pp. 207–216, May 1993.