Data & Knowledge Engineering 68 (2009) 1014–1033

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Mining sequential patterns across multiple sequence databases

Wen-Chih Peng *, Zhung-Xun Liao

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC

ARTICLE INFO

Article history:
Received 26 April 2008
Received in revised form 11 April 2009
Accepted 13 April 2009
Available online 7 May 2009

Keywords:
Data mining
Sequential pattern mining
Multi-domain sequential patterns

ABSTRACT

In this paper, given a set of sequence databases across multiple domains, we aim at mining multi-domain sequential patterns, where a multi-domain sequential pattern is a sequence of events whose occurrence time is within a pre-defined time window. We first propose algorithm Naive in which multiple sequence databases are joined as one sequence database for utilizing traditional sequential pattern mining algorithms (e.g., PrefixSpan). Due to the nature of join operations, algorithm Naive is costly and is developed for comparison purposes. Thus, we propose two algorithms without any join operations for mining multi-domain sequential patterns. Explicitly, algorithm IndividualMine derives sequential patterns in each domain and then iteratively combines sequential patterns among sequence databases of multiple domains to derive candidate multi-domain sequential patterns. However, not all sequential patterns mined in the sequence database of each domain are able to form multi-domain sequential patterns. To avoid the mining cost incurred in algorithm IndividualMine, algorithm PropagatedMine is developed. Algorithm PropagatedMine first performs one sequential pattern mining from one sequence database. In light of sequential patterns mined, algorithm PropagatedMine propagates sequential patterns mined to other sequence databases. Furthermore, sequential patterns mined are represented as a lattice structure for further reducing the number of sequential patterns to be propagated. In addition, we develop some mechanisms to allow some empty sets in multi-domain sequential patterns. Performance of the proposed algorithms is comparatively analyzed and sensitivity analysis is conducted. Experimental results show that by exploring propagation and lattice structures, algorithm PropagatedMine outperforms algorithm IndividualMine in terms of efficiency (i.e., the execution time).

1. Introduction

Sequential pattern mining has attracted a considerable amount of research effort recently [3,4,9,23,24]. Given a sequence database that contains a set of sequences and a user-specified threshold (the minimum support), the main task of sequential pattern mining is to discover frequent subsequences that appear in a sufficient number of sequences. Since sequential pattern mining is able to discover temporal relationship (i.e., order of events), a significant amount of research works has elaborated on developing novel approaches to discover sequential patterns for a variety of applications [7,10,19,22,25,26].

Note that prior works only mine sequential patterns in one sequence database. This sequence database consists of sequences of events in one domain. For example, given a sequence database of purchasing in a supermarket, frequent purchasing behavior is discovered. In many real world applications, we may have events in multiple domains. Consider payment lists of credit cards, where a user uses a credit card for a variety of services, such as payments in restaurants, food, books and movies. These payments are referred to as events in different domains. For each domain, we could extract these events

* Corresponding author.
E-mail addresses: wcpeng@cs.nctu.edu.tw (W.-C. Peng), zxliao.cs96g@cs.nctu.edu.tw (Z.-X. Liao).

0169-023X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2009.04.009

and build the corresponding sequence database. Then, one could utilize sequential pattern mining to discover frequent sequences. For example, in the movie domain, one sequential pattern is that users watch a series of movies related to Harry Potter. On the other hand, in the book domain, one sequential pattern is that users buy a series of novels related to Harry Potter. From these two sequential patterns of two domains, one could derive a composite sequential pattern across two domains (referred to as a *multi-domain sequential pattern*) if events in these two sequential patterns closely occur together (i.e., events occur within the same time window). For the above example, Fig. 1 shows that these two sequences are sequential patterns in the movie domain and the book domain, respectively. Moreover, the corresponding time of events in these two sequences are within the same time window. In this paper, we claim that discovering sequential patterns from multiple domains will provide a unique way to reveal complex relationships across multiple domains. In Fig. 1, one could infer that users are likely to buy novels of Harry Potter, which is motivated by movies. Furthermore, this multi-domain sequential pattern also implies that users who go to watch movies are likely to be triggered by books bought. To reveal more information from sequence databases across multiple domains, multi-domain sequential patterns are very useful. Depending on requirements of applications, one could decide which domains should be involved in mining multi-domain sequential patterns. In our above example, if a user wants to know the cross-relationship between book and movie domains, sequence databases of these two domains are given. Consequently, such a multi-domain sequential pattern captures the cross-relationship among multiple domains, which in turn can yield significant information and reveal more knowledge.

Given a set of sequence databases across multiple domains, we aim at mining multi-domain sequential patterns, where a multi-domain sequential pattern is a sequence of events whose occurrence time is within a pre-defined time window. With a set of sequence databases, these sequence databases could be joined into one sequence database according to time information of sequences. Then, by exploring traditional sequential pattern mining algorithms, we could obtain multi-domain sequential patterns as well. This method is referred to as algorithm *Naive* in our paper and the details of this algorithm are presented later. However, there are three drawbacks in this algorithm: (1) integrating sequence databases of multiple domains into a single sequence database incurs a considerable cost due to the nature of joining operations, (2) the length of each sequence becomes longer and the number of items becomes huge after joining operations, and (3) sequential patterns mined should be further verified whether these sequential patterns satisfy multi-domain sequential patterns or not. Hence, the above algorithm unavoidably exhibits poor efficiency and scalability performance, which calls for the design of efficient mining algorithms for multi-domain sequential patterns.

To avoid the above poor performance issues, in this paper, we first propose algorithm IndividualMine in which sequential patterns in each sequence database should be mined first and then the sequential patterns in each domain are integrated as candidate multi-domain sequential patterns. Clearly, algorithm IndividualMine is able to avoid join operations among sequence databases. In Fig. 1, it can be seen that in the movie domain (respectively, book domain), we could have one sequential pattern, a series of movies (respectively, books) related to Harry Potter. By checking the corresponding time of events in these two sequential patterns, we could combine these two sequential patterns into one multi-domain sequential pattern since each event in these two sequential patterns has close occurrence time. It is possible that sequential patterns from each sequence database cannot be formed as multi-domain sequential patterns since events' occurrence time is not close. Though avoiding join operations, algorithm IndividualMine is likely to suffer from mining cost since sequential patterns in each sequence database should be discovered. Consequently, we propose algorithm PropagatedMine to further reduce the mining cost in each sequence database. Algorithm PropagatedMine first performs one sequential pattern mining from one sequence database. In light of sequential patterns mined, algorithm PropagatedMine propagates time information (referred to as the time-instance set) of sequential patterns mined to other sequence databases. By utilizing time-instance sets, we are able to extract a subset of sequences from sequence databases, where the subset of sequences has the same time information. As such, only a limited number of sequences that are likely to form multi-domain sequential patterns are extracted. Furthermore, sequential patterns mined are represented as a lattice structure for further reducing the number of time-instance sets to other sequence databases. In addition, we develop some mechanisms to allow some empty sets in multi-domain sequential patterns. Performance of the proposed algorithms is comparatively analyzed and sensitivity analysis is conducted. It is shown by our simulation results that both algorithms IndividualMine and PropagatedMine perform better than algorithm Naive. By exploring propagation and lattice structures, algorithm PropagatedMine outperforms algorithms IndividualMine and Naive in terms of efficiency (i.e., the execution time).

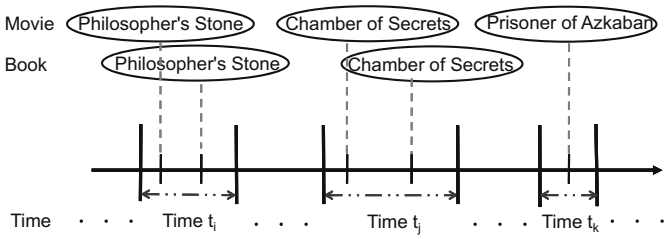

Fig. 1. An example of multi-domain sequential pattern.

The remainder of the paper is organized as follows. In Section 2, we present existing research works of mining sequential patterns. In Section 3, some notations and the problem definition are given. Our proposed algorithms are described in Section 4. Performance study and experimental results are shown in Section 5. Section 6 concludes with this paper.

2. Related works

A significant amount of research efforts has been devoted to sequential pattern mining [6,11,16,28–31]. The problem of sequential pattern mining is first formulated in [3] and the authors in [3] proposed mining algorithms based on the Apriori algorithm. Algorithm GSP [27] was developed for mining sequential patterns using a breadth first search and button-up method, whereas algorithm SPADE [33] employed a depth first search and button-up method with an ID-list. The authors in [13,23,24] exploited the projection concept to reduce the amount of data for sequential pattern mining. To prevent candidate generation, DISC-all [8] used a novel sequence comparison strategy. A progressive concept has been explored in mining sequential patterns to capture the dynamic nature of data addition and deletion [15]. The above research works are focused on improving the performance of traditional sequential pattern mining.

Some variations and applications on sequential patterns are proposed recently. We mention in passing that the authors in [25] developed to mine multi-dimensional sequential patterns, in which sequential patterns indicate not only frequent sequences but also some attributes in the category dimensions. In [25], the sequence database consists of category attributes and sequence attributes, and Table 1 shows an example of a multi-dimensional sequence database. Clearly, the problem of mining multi-domain sequential patterns is very different from the problem in [25] in terms of the input and output of problem definitions. In this paper, the input is the set of sequence databases and the output is the set of multi-domain sequential patterns that consist sequences of co-occurred events across sequence databases. The authors in [32] explore the concept of mining sequential patterns from multi-dimensional sequences. However, the output of their proposed method is a set of high-dimensional sequential patterns, which consist of sequential patterns from multiple sequence datasets. However, in their proposed algorithm, time information is not associated with each event, and thus each event from multiple dimensions of a high-dimensional sequential pattern is not co-occurred. As such, a high-dimensional sequential pattern is intrinsically different from our proposed multi-domain sequential pattern. Furthermore, the problem in [5] is to discover events that are occurred together. In contrast, our problem is that given a set of sequence databases, we intend to discover sequences consisting of co-occurred events. Moreover, the authors in [12] proposed the problem of distributed sequential pattern mining, where each set of co-occurred events is complete and sequences are separated into different databases. Similarly, the problem in [17] is indeed a distributed sequential pattern mining problem and the authors in [17] exploited the concepts of approximate patterns and local clustering to avoid noise and a large number of local patterns. As pointed early, given a payment list of credit cards, we could divide payments into several domains according to payment services. Thus, our problem of mining multi-domain sequential patterns is not the same as distributed mining of sequential patterns.

To the best of our knowledge, previous studies have not adequately explored multi-domain sequential patterns, let alone proposing efficient algorithms for mining such sequential patterns. The contributions of this paper are twofold: (1) exploiting novel and useful sequential patterns (i.e., multi-domain sequential patterns), and (2) devising algorithms IndividualMine

Table 1
An example of multi-dimensional sequence database in [25].

Cid	Cust-grp	City	Age-grp	Sequence
10	Business	Boston	Middle	$\langle(bd)(c)(b)(a)\rangle$
20	Professional	Chicago	Young	$\langle(bf)(ce)(fg)\rangle$
30	Business	Chicago	Middle	$\langle(ah)(a)(b)(f)\rangle$
40	Education	New York	Retired	$\langle(be)(ce)\rangle$

Table 2
An example of sequence databases in two domains.

ID	Time sequences	Sequences
Domain database D_1		
s_1	$\langle(T_1)(T_2)(T_3)(T_4)\rangle$	$\langle(a)(b,c)(b,c,d)(e)\rangle$
s_2	$\langle(T_5)(T_6)(T_7)\rangle$	$\langle(a,b)(b,c)(c,e)\rangle$
s_3	$\langle(T_{10})(T_{12})(T_{13})\rangle$	$\langle(a,e)(h)(g,j)\rangle$
s_4	$\langle(T_{21})(T_{22})(T_{23})(T_{24})\rangle$	$\langle(a,b,f)(d)(b,c)(e,f)\rangle$
Domain Database D_2		
l_1	$\langle(T_{21})(T_{22})(T_{23})(T_{24})\rangle$	$\langle(1,2,5)(7)(2,3)(4,5,6)\rangle$
l_2	$\langle(T_{10})(T_{12})(T_{13})\rangle$	$\langle(1,6)(5)(9,10)\rangle$
l_3	$\langle(T_5)(T_6)(T_7)\rangle$	$\langle(1,3)(2,4)(8)\rangle$
l_4	$\langle(T_1)(T_2)(T_3)(T_4)\rangle$	$\langle(1,2)(2,3)(6)(4,5)\rangle$

and PropagatedMine to efficiently mine multi-domain sequential patterns. Our preliminary works were presented in [20,21]. In this paper, more detailed complexity and theoretical analysis are conducted. Also, we develop some mechanisms in each proposed algorithm to allow multi-domain sequential patterns with some empty sets in some domains. In particular, by exploring lattice structures, algorithm PropagatedMine is able to further reduce the number of candidate multi-domain sequential patterns. Furthermore, an extensive performance study is conducted and sensitivity analysis is investigated on several parameters for each algorithm.

3. Preliminaries

Assume that each domain has its own set of items and a sequence database. The problem of mining multi-domain sequential patterns is that given a set of sequence databases, we aim at discovering sequential patterns that consist of co-occurred events across multiple domains. Table 2 shows two domains with its own sequence database, where in each sequence of sequence databases, the corresponding time sequence indicates the occurrence time of events. For example, in sequence s_1 in D_1, it can be seen that event a occurs at T_1 and both b and c occur at T_2. By joining these two sequence databases via their time sequences, we could have one multi-domain sequence database (referred to as MDB). As such, Table 3 is an example of multi-domain sequence database.

To facilitate the presentation of multi-domain sequences, one sequence s_i in domain D_i is expressed by $\langle X_{i1}, X_{i2}, \ldots, X_{il}\rangle$, where X_{ij} is the jth element of sequence s_i, and l is the number of elements of s_i. Therefore, a multi-domain sequence across k domains (abbreviated as *k-domain sequence*) is represented as $M = [s_1, s_2, \ldots, s_k]^T$ and is further denoted as $M = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1l} \\ X_{21} & X_{22} & \ldots & X_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ X_{k1} & X_{k2} & \ldots & X_{kl} \end{bmatrix}$, where each column is a set of itemsets that occur within the same time window, denoted as T_i. A time sequence $TS(M)$ is represented as $TS(M) = \langle T_1, T_2, \ldots, T_l\rangle$ to indicate the occurrence time of M. Actually, the time window, a time interval, is determined in accordance with application requirements.

With the above representation of multi-domain sequences, we further define the length and the number of elements for multi-domain sequential patterns. Since a multi-domain sequence consists of multiple sequences from various domains, the length of a multi-domain sequence across k domains can be defined as follows:

Definition 1 (*Length and number of elements*). Let $M = [s_1, s_2, \ldots, s_k]^T$ be a k-domain sequence. The length of M, denoted as $|M|$, is the length of the longest sequence in multi-domain sequence M. Furthermore, the number of elements in a multi-domain sequence, expressed by $e(M)$, is the number of itemsets in the multi-domain sequence.

For example, given a 2-domain sequence $M = \begin{bmatrix} (a) & (b,c) & (b) \\ (1) & (2) & (1,2,3) \end{bmatrix}$, the length of M is 5 due to that the longest sequence $\langle(1)(2)(1,2,3)\rangle$ in M, and the number of elements is 3 (i.e., $e(M) = 3$).

Once we have the definition of the length and the number of elements for a multi-domain sequences, the containing relation among multi-domain sequences is thus defined as follows:

Definition 2 (*Containing relation*). Suppose that we have two multi-domain sequences $M = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1b} \\ X_{21} & X_{22} & \ldots & X_{2b} \\ \vdots & \vdots & \ddots & \vdots \\ X_{a1} & X_{a2} & \ldots & X_{ab} \end{bmatrix}$ and $N = \begin{bmatrix} Y_{11} & Y_{12} & \ldots & Y_{1b'} \\ Y_{21} & Y_{22} & \ldots & Y_{2b'} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{a1} & Y_{a2} & \ldots & Y_{ab'} \end{bmatrix}$, where $e(M) \leqslant e(N)$. M is contained by N, denoted as $M \sqsubseteq N$, if and only if there exists an integer list $L(M,N)$, denoted as $\langle l_1, l_2, \ldots, l_b\rangle$, such that $1 \leqslant l_1 < l_2 < \cdots < l_b \leqslant b'$ and $X_{ij} \subseteq Y_{il_j}$, where $1 \leqslant i \leqslant a$ and $1 \leqslant j \leqslant b$.

Table 3
An example of a multi-domain sequence database.

ID	Time sequences	Multi-domain sequences
S_1	$\langle(T_1)(T_2)(T_3)(T_4)\rangle$	$\begin{bmatrix} (a) & (b,c) & (b,c,d) & (e) \\ (1,2) & (2,3) & (6) & (4,5) \end{bmatrix}$
S_2	$\langle(T_5)(T_6)(T_7)\rangle$	$\begin{bmatrix} (a,b) & (b,c) & (c,e) \\ (1,3) & (2,4) & (8) \end{bmatrix}$
S_3	$\langle(T_{10})(T_{12})(T_{13})\rangle$	$\begin{bmatrix} (a,e) & (h) & (g,j) \\ (1,6) & (5) & (9,10) \end{bmatrix}$
S_4	$\langle(T_{21})(T_{22})(T_{23})(T_{24})\rangle$	$\begin{bmatrix} (a,b,f) & (d) & (b,c) & (e,f) \\ (1,2,5) & (7) & (2,3) & (4,5,6) \end{bmatrix}$

For example, assume that $M = \begin{bmatrix} (a) & (b,c) \\ (2) & (6) \end{bmatrix}$ and $N = \begin{bmatrix} (a) & (b,c) & (b,c,d) & (e) \\ (1,2) & (2,3) & (6) & (4,5) \end{bmatrix}$. It can be verified that M is contained by N since there exist integer list $L(M,N) = \langle 1,3 \rangle$ such that $1 \leqslant 1 < 3 \leqslant 4$, and $(a) \subseteq (a)$, $(2) \subseteq (1,2)$, $(b,c) \subseteq (b,c,d)$ and $(6) \subseteq (6)$.

Based on the above definitions, a multi-domain sequence database is a set of multi-domain sequences. Consider an example of a multi-domain sequence database in Table 3, where the number of 2-domain sequences is 4. Given a multi-domain sequence database *MDB*, the support value of a multi-domain sequence M is the number of multi-domain sequences in *MDB* that contain the multi-domain sequence M.

Multi-domain sequential pattern mining: Given a set of sequence databases across multiple domains, one could join these sequence databases as one multi-domain sequence database. Then, the task of mining multi-domain sequential patterns is to derive multi-domain sequences with their supports larger than a user-specified minimum support threshold δ in MDB. For example, for the multi-domain sequence database *MDB* in Table 3 and a minimum support $\delta = 3$, 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 $\begin{bmatrix} (a) & (b,c) \\ (1) & (2) \end{bmatrix}$.

Notice that joining these multiple sequence databases is costly due to the nature of join operations. It can be verified that multi-domain sequential patterns contain sequential patterns in each domain. For example, $\begin{bmatrix} (a) & (b,c) \\ (1) & (2) \end{bmatrix}$ is a multi-domain sequential pattern, where $(a)(b,c)$ (respectively, $(1)\,(2)$) is a sequential pattern in domain D_1 (respectively, D_2) and events in $(a)(b,c)$ and $(1)\,(2)$ has the same time sequences. Thus, in this paper, we propose algorithms to discover multi-domain sequential patterns without joining.

4. Algorithms of mining multi-domain sequential patterns

In this section, we first describe one *Naive method* in which multiple sequence databases are joined as one sequence database, and multi-domain sequential patterns are derived by using traditional sequential pattern mining algorithms (e.g., PrefixSpan [23,24]). As pointed out early, to avoid the overheads of joining multiple sequence databases, we then propose algorithm IndividualMine in which sequential patterns in each sequence database should be mined and further merged for possible multi-domain sequential patterns. Furthermore, to further reduce the cost of mining sequential patterns in each sequence database, algorithm PropagatedMine is proposed. By propagation of sequential patterns to other sequence databases, the number of sequences in other sequence databases is reduced. In addition, the above three algorithms could be extended to discover multi-domain sequential patterns with some empty sets in some domains.

4.1. Naive algorithm with one multi-domain sequence database

As mentioned early, to mine multi-domain sequential patterns, one Naive method is joining sequence databases into one multi-domain sequence database. Then, this multi-domain sequence database is transformed such that the Naive algorithm could utilize existing sequential pattern mining algorithms. Consequently, in the Naive algorithm, there are two steps: the joining step and the mining step. In the joining step, multiple sequence databases are first joined together by the time sequences and then the multiple sequence databases are thus transformed into a sequence database. In the mining step, one could utilize existing sequential pattern mining algorithms to derive sequential patterns. In light of sequential patterns mined, we have to separate the items from different domains and derive multi-domain sequential patterns. The detailed steps are described as follows:

Step 1: joining step: In the beginning, sequence databases are joined by their time sequences to form one multi-domain sequence database. For example, Table 3 is derived by performing the join process among two sequence databases in Table 2. It can be verified that s_1 in D_1 sequence database and l_4 in D_2 sequence database are joined as one sequence S_1 in Table 3. With the multi-domain sequence database derived, one should transform this multi-domain sequence database into one sequence database. Explicitly, in Table 3, for each sequence, time sequences are deleted and multi-domain sequences could be viewed as one sequence. Table 4 is an example of a sequence database transformed from Table 3. It can be seen that in sequence S_1 in Table 4, co-occurred events from multiple domains are viewed as one event. For example, (a, 1, 2) comes from $\begin{bmatrix} (a) \\ (1,2) \end{bmatrix}$ in sequence S_1 of Table 3.

Table 4
An example of a transformed sequence database.

ID	Sequences
S_1	$\langle (a,1,2)(b,c,2,3)(b,c,d,6)(e,4,5) \rangle$
S_2	$\langle (a,b,1,3)(b,c,2,4)(c,e,8) \rangle$
S_3	$\langle (a,e,1,6)(h,5)(g,j,9,10) \rangle$
S_4	$\langle (a,b,f,1,2,5)(d,7)(b,c,2,3)(e,f,4,5,6) \rangle$

Table 5
An example of a transformed sequence database.

Pattern ID	Sequential patterns	multi-domain sequential patterns
P_1	$\langle(1)(b,2)(e)\rangle$	$\begin{bmatrix} & (b) & (e) \\ (1) & (2) & \end{bmatrix}$
P_2	$\langle(a,1)(5)\rangle$	$\begin{bmatrix} (a) & \\ (1) & (5) \end{bmatrix}$
P_3	$\langle(a,1)(c,2)\rangle$	$\begin{bmatrix} (a) & (c) \\ (1) & (2) \end{bmatrix}$
P_4	$\langle(b,3)\rangle$	$\begin{bmatrix} (b) \\ (3) \end{bmatrix}$
P_5	$\langle(b,c,2)\rangle$	$\begin{bmatrix} (b,c) \\ (2) \end{bmatrix}$
P_6	$\langle(b,c,2)(e)\rangle$	$\begin{bmatrix} (b,c) & (e) \\ (2) & \end{bmatrix}$

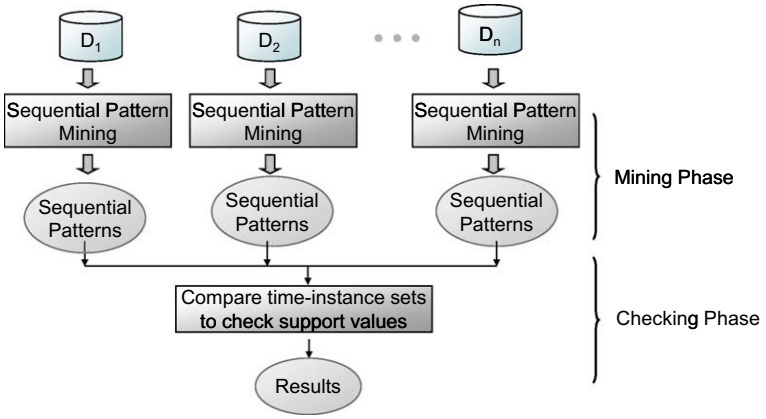

Fig. 2. An overview of algorithm IndividualMine.

Step 2: mining step: According to the sequence database derived in Step 1, by exploiting traditional sequential pattern mining algorithms, we could derive sequential patterns. The second column of Table 5 shows some examples of sequential patterns mined from the sequence database in Table 4 with the minimum support as 3. However, even if a sequence database is obtained, traditional sequential pattern mining algorithms are not directly able to mine multi-domain sequential patterns. This is due to that several sequential patterns mined do not contain events from all domains. Thus, each sequential pattern should be represented as multi-domain sequential patterns. Then, we could first verify whether multi-domain sequential patterns consists of events from all domains or not. For example, the third column of Table 5 shows multi-domain sequential patterns from the second column of Table 5. Since we have all events of all domains, it is very straightforward to represent sequential patterns as multi-domain sequential patterns. It can be seen in Table 5, P_1, P_2 and P_6 have some empty sets and these patterns are referred to as multi-domain sequential patterns with empty sets (abbreviated as *relaxed multi-domain sequential patterns*). On the other hands, P_3, P_4 and P_5 are called *strong multi-domain sequential patterns* since all co-occurred events are from all domains required.

Algorithm Naive needs to perform join operations among multiple sequence databases. Due to join operations, the performance of algorithm Naive is not efficient. Furthermore, in order to utilize traditional sequential pattern mining algorithms, one sequence database is derived by transforming from one multi-domain sequence database joined from sequence databases. Clearly, with events from all domains, the sequence database contains long sequences, which is not efficient in mining sequential patterns. With the above two drawbacks of algorithm Naive, we develop two efficient algorithms for mining multi-domain sequential patterns without joining sequence databases.

4.2. Algorithm IndividualMine: mining patterns in each domain

In this section, we develop algorithm IndividualMine. Fig. 2 shows the overview of algorithm IndividualMine, where algorithm IndividualMine consists of two phases: the mining phase and the checking phase. In the mining phase, sequential patterns in each sequence database are first mined by utilizing sequential pattern mining algorithms (e.g., PrefixSpan [23,24]).

In the checking phase, sequential patterns from all domains are combined to generate candidate multi-domain sequential patterns. If a candidate multi-domain sequential pattern has its support value larger than the minimum support threshold, this candidate multi-domain sequential pattern is a multi-domain sequential pattern. The support counts of candidate multi-domain sequential patterns will be described later.

Without loss of generality, given k sequence databases, we intend to derive multi-domain sequential patterns across k domains. Furthermore, we denote the set of k sequence databases as $\{D_1, D_2, \ldots, D_k\}$, and SP_i as the set of i-domain sequential patterns across a set of i sequence databases (i.e., $\{D_1, D_2, \ldots, D_i\}$). To derive k-domain sequential patterns, we should start with one sequential patterns from one domain and progressively composite sequential patterns from other domains until the number of domains is k. Hence, sequential patterns mined in D_1 is first in the set of SP_1. Then, for each pattern in SP_1, candidate 2-domain sequential patterns (across two domains $\{D_1$ and $D_2\}$) are generated by combining sequential patterns in domain D_2. For example, given a minimum support as 3, in our above example in Table 2, $\langle(a)(b)\rangle$ is a sequential pattern and is put in the set of SP_1. Also, $\langle(1),(2)\rangle$ is one sequential pattern in D_2. Consequently, we could have a candidate 2-domain sequential pattern $\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}$.

After generating candidate multi-domain sequential patterns, their support values should be determined. As can be seen in Table 2, each sequence is associated with its own time sequence. Thus, one could use time sequences to derive support values. Explicitly, the time-instance set of sequence M is defined as follows:

Definition 3 (*Time-instance set*). Let *MDB* be a k-domain sequence database[1] and M be a k-domain sequence. The time-instance set of M is defined as $TIS(M) = \{\langle TS(N) : L(M,N)\rangle | N \in MDB \text{ and } M \sqsubseteq N\}$.

Based on the above definition, for a candidate multi-domain sequential pattern, we could determine its support value by evaluating the intersections in time-instance sets of each sequential pattern. For example, to determine the support of $\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}$, we should check both time-instance set of $\langle(a)(b)\rangle$ and $\langle(1)(2)\rangle$ in Table 2. It can be seen that in Table 2, the time-instance set of $\langle(a)(b)\rangle$ is $\{\langle(T_1)(T_2)(T_3)(T_4) : 1,2\rangle, \langle(T_1)(T_2)(T_3)(T_4) : 1,3\rangle, \langle(T_5)(T_6)(T_7) : 1,2\rangle, \langle(T_{21})(T_{22})(T_{23})(T_{24}) : 1,3\rangle\}$. Moreover, we could have $TIS(\langle(1)(2)\rangle)$ as $\{\langle(T_1)(T_2)(T_3)(T_4) : 1,2\rangle, \langle(T_5)(T_6)(T_7) : 1,2\rangle, \langle(T_{21})(T_{22})(T_{23})(T_{24}) : 1,3\rangle\}$. Thus, the support of a candidate 2-domain sequential pattern $\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}$ is represented as $TIS\left(\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}\right) = \{\langle(T_1)(T_2)(T_3)(T_4) : 1,2\langle, \langle(T_5)(T_6)(T_7) : 1,2\rangle, \langle(T_{21})(T_{22})(T_{23})(T_{24}) : 1,3\rangle\}$. Therefore, $Support\left(\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}\right) = \left|TIS\left(\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}\right)\right| = 3$. Given a minimum support threshold of 3, $\begin{bmatrix}(a) & (b)\\(1) & (2)\end{bmatrix}$ is a 2-domain sequential pattern, since its support value is not less than the minimum support. Consequently, through the time-instance sets, support values for candidate multi-domain sequence patterns are derived.

Once we have 2-domain sequential patterns, these 2-domain sequential patterns are in the set of SP_2. Then, for each pattern in SP_2, candidate 3-domain sequential patterns and their corresponding supports will be generated by the above same procedure. Given sequential patterns in k domains, k-domain sequential patterns are derived by iteratively expanding one domain in each round until the number of rounds is k.

Algorithm: IndividualMine
Input: Sequence databases across n domains $D_1, D_2, \ldots, D_n$, and minimum support δ.
Output: Multi-domain sequential patterns across n domains.
Begin
  Let C_k be the set of candidate patterns across k domains, where $k = 1, 2, \ldots, n$.
  Apply sequential pattern mining on each domain $D_i, i = 1, 2, \ldots, n$.
  Let SP_1 be the set of sequential patterns mined in D_1.
  For each domain D_{i+1}, $i = 1, 2, \ldots, n-1$
    For each $P \in SP_i$
      For each sequential pattern Q of D_{i+1}
        If $e(Q) = e(P)$ **Then** append $\begin{bmatrix}P\\Q\end{bmatrix}$ to C_{i+1}.
    For each candidate $c \in C_{i+1}$
      If $Support(c) \geqslant \delta$ **Then** append c to SP_{i+1}.
  Output $= SP_n$.
End

[1] To facilitate our presentation, one could image that *MDB* are virtually joined by multiple sequence databases.

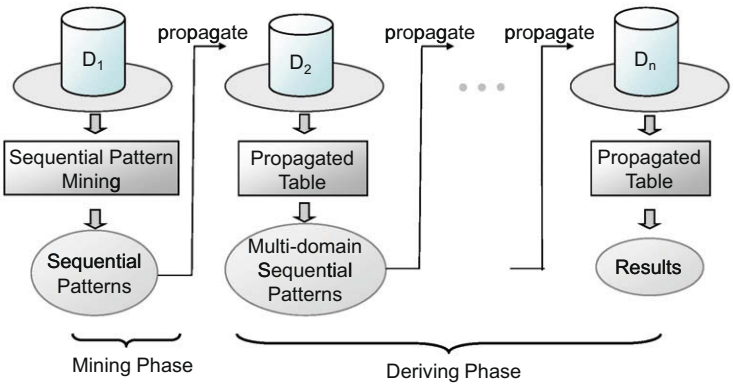

Fig. 3. An overview of algorithm PropagatedMine.

Without joining, algorithm IndividualMine could still discover multi-domain sequential patterns. It can be seen that in algorithm IndividualMine, each domain should individually perform sequential pattern mining algorithms, which incurs a considerable amount of mining cost. Furthermore, those sequential patterns mined from each domain are not necessarily able to become multi-domain sequential patterns. Thus, to further reduce the cost of mining sequential patterns in each domain and the number of candidate multi-domain sequential patterns, we develop algorithm PropagatedMine in which those sequences that are likely to form multi-domain sequential patterns are extracted from their sequence databases.

4.3. Algorithm PropagatedMine: propagating sequential patterns among domains

Algorithm PropagatedMine is designed to reduce the mining cost in each sequence database. Explicitly, algorithm PropagatedMine first performs sequential pattern mining in one domain (referred to as the starting domain) and then propagates time-instance sets of the mined sequential patterns to other domains. By propagating time-instance sets, only those sequences that have the same time sequences with the time-instance sets are extracted, thereby reducing the mining space in each sequence database. Algorithm PropagatedMine iteratively propagates time-instance sets of multi-domain sequential patterns to the next domain until all domains have been mined. Fig. 3 shows an overview of algorithm PropagatedMine, where there are two phases in algorithm PropagatedMine: the mining phase and the deriving phase.

In the mining phase, PropagatedMine utilizes existing sequential pattern mining algorithms to discover sequential patterns in a starting domain (i.e., D_1) and then propagates these patterns to other domains. Note that the mined sequential patterns in the starting domain provide a guideline to extract multi-domain sequential patterns from other domains, and hence for mining multi-domain sequential patterns in sequence databases across multiple domains, the length and the number of elements of multi-domain sequences are constrained by sequential patterns mined in the starting domain. Consequently, sequential patterns mined in the starting domain could be represented as a lattice structure to facilitate the generation of candidate multi-domain sequential patterns across other domains.

For example, assume that the starting domain is set to D_1 in Table 2 and that sequential patterns are then found using existing sequential pattern mining algorithms with the same minimum support 3. The mined sequential patterns are represented as a lattice structure in Fig. 4, where each node represents a sequential pattern, the linkages of nodes (or *intradomain links*) represent containing relation, and nodes are ordered by the number of elements. In Fig. 4, those nodes having the same number of elements are further arranged level by level according to their sequence lengths and nodes with one element are placed level by level in increasing order of sequence length. For example, $\langle(b,c)\rangle$ in Fig. 4 is below the nodes whose sequence

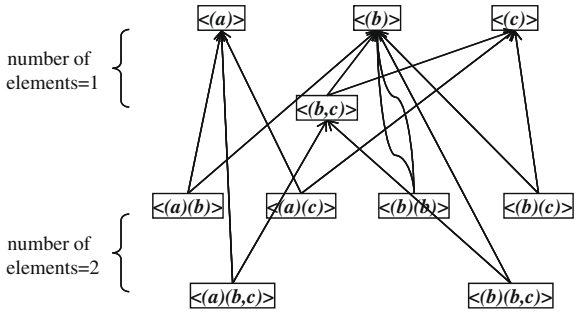

Fig. 4. An example of lattice structures for sequential patterns in a starting domain (i.e., D_1 in Table 2).

Table 6
An example of propagated table $D_2||_{\langle(a)(c)\rangle}$.

Time sequences	Sequences
$\langle(T_1)(T_2)(T_3)(T_4)\rangle$	(1,2)(2,3)
$\langle(T_1)(T_2)(T_3)(T_4)\rangle$	(1,2)(6)
$\langle(T_5)(T_6)(T_7)\rangle$	(1,3)(2,4)
$\langle(T_5)(T_6)(T_7)\rangle$	(1,3)(8)
$\langle(T_{21})(T_{22})(T_{23})(T_{24})\rangle$	(1,2,5)(2,3)

length is 1 (e.g., $\langle(b)\rangle$). As mentioned above, the lattice structure is used as a guideline for propagating time-instance sets of sequential patterns to other domains. In the deriving phase, algorithm PropagatedMine extracts those sequences with occurrence times equal to those of the time-instance sets propagated. Thus, for each propagated time-instance set, we can build the corresponding propagated table as defined in Definition 4.

Definition 4 (*Propagated table*). Let M be a k-domain sequential pattern. The propagated table of M in sequence database D_{k+1} is denoted as $D_{k+1}||_M = \{\langle S_i[l_1], S_i[l_2], \ldots, S_i[l_b]\rangle | \langle TS(S_i) : l_1, l_2, \ldots, l_b\rangle \in TIS(M), \text{ where } S_i \in D_{k+1}\}$ which is consisted of sequences that co-occurred with M. Furthermore, $D_{k+1}||_M$ is also a sequence database, and $\begin{bmatrix} M \\ S \end{bmatrix}$ is a $(k+1)$-domain sequential pattern if and only if S is a sequential pattern of $D_{k+1}||_M$ and $e(S) = e(M)$ with the same minimum support threshold.

For example, in domain D_1 of Table 2, we have $TIS(\langle(a)(c)\rangle) = \{\langle(T_1)(T_2)(T_3)(T_4) : 1,2\rangle, \langle(T_1)(T_2)(T_3)(T_4) : 1,3\rangle, \langle(T_5)(T_6)(T_7) : 1,2\rangle, \langle(T_5)(T_6)(T_7) : 1,3\rangle, \langle(T_{21})(T_{22})(T_{23})(T_{24}) : 1,3\rangle\}$, and propagating $TIS(\langle(a)(c)\rangle)$ to domain D_2 yields propagated table $D_2||_{\langle(a)(c)\rangle}$. Table 6 is the propagated table $D_2||_{\langle(a)(c)\rangle}$, where each sequence is very likely to form multi-domain sequential patterns with $\langle(a)(c)\rangle$ mined from domain D_1. From propagated tables, one could mine sequential patterns having the same number of elements as the propagated sequential pattern and these sequential patterns could be formed as multi-domain sequential patterns. Consider the above example, where the minimum support is set to 3. We can easily find that $\langle(1)(2)\rangle$ is the sequential pattern of $D_2||_{\langle(a)(c)\rangle}$ and thus $\begin{bmatrix} (a) & (c) \\ (1) & (2) \end{bmatrix}$ is a 2-domain sequential pattern by compositing $\langle(a)(c)\rangle$ and $\langle(1)(2)\rangle$.

Note that even though PropagatedMine successfully prevents mining sequential patterns in each domain, however, the cost of some redundant mining of propagated tables can be further reduced. For example, some patterns mined in propagated tables $D_2||_{\langle(a)\rangle}$ and $D_2||_{\langle(c)\rangle}$ are the same as patterns mined in propagated table $D_2||_{\langle(a)(c)\rangle}$. This is due to that the time-instance set of $\langle(a)(c)\rangle$ is contained in both time-instance sets of $\langle(a)\rangle$ and $\langle(c)\rangle$. Consequently, sequences in propagated table $D_2||_{\langle(a)(c)\rangle}$ also include some sequences in propagated table $D_2||_{\langle(a)\rangle}$ and $D_2||_{\langle(c)\rangle}$. Therefore, only sequential patterns with their length being one should be propagated to other domains. In other words, only time-instance sets of the top-level nodes (referred to as *atomic patterns*) in lattice structures are propagated. After obtained, propagated tables are viewed as transaction databases. Consequently, given a propagated table, by utilizing frequent itemset algorithms in [1,2,34,14], we could generate the corresponding multi-domain sequential patterns. We now analyze some important properties of the propagated table. With these properties of propagated tables, the lattice structure in the starting domain is used to determine multi-domain sequential patterns whose length is larger than one. The details of generating multi-domain sequential patterns are described later.

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

Property of antimonotone with multiple domains: If M is a k-domain sequential pattern (i.e., across $D_1, D_2, \ldots$, and D_k), k-domain sequences contained by M are also k-domain sequential patterns.

Based on the antimonotone property, algorithm PropagatedMine generates candidate multi-domain sequential patterns in a level-by-level manner. However, in the propagated domain, sequential patterns are also generated level by level according to the number of sequence elements. The detailed steps for deriving multi-domain sequential patterns are described below.

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

Let SP_k be the set of multi-domain sequential patterns across k domains. When deriving atomic patterns across $(k+1)$ domains, the corresponding frequent itemsets can be derived from the propagated tables of each atomic pattern in SP_k. Through the property of propagated table of atomic patterns, those frequent items mined from propagated tables are merged with atomic patterns in SP_k to derive atomic patterns across $(k+1)$ domains. Consider the sequence databases across two domains in Table 2 as an example, where sequential patterns of domain D_1 are represented as a lattice structure. We could derive atomic patterns in domain D_2 and thus generate their corresponding multi-domain sequential patterns by propagating the time-instance sets of atomic patterns in domain D_1 (i.e., the top-level nodes) to domain D_2. Specifically, in Fig. 5, for

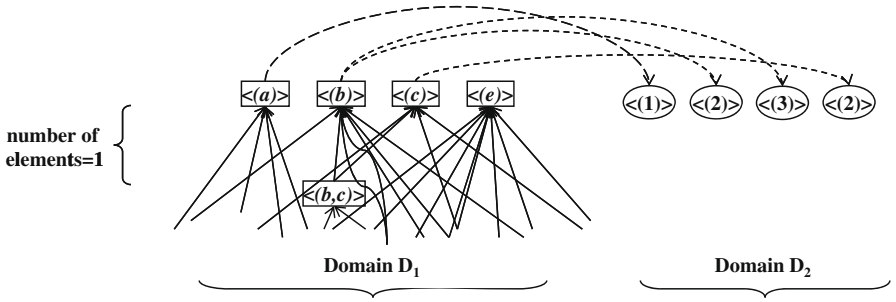

Fig. 5. An example of generating atomic patterns in domain D_2.

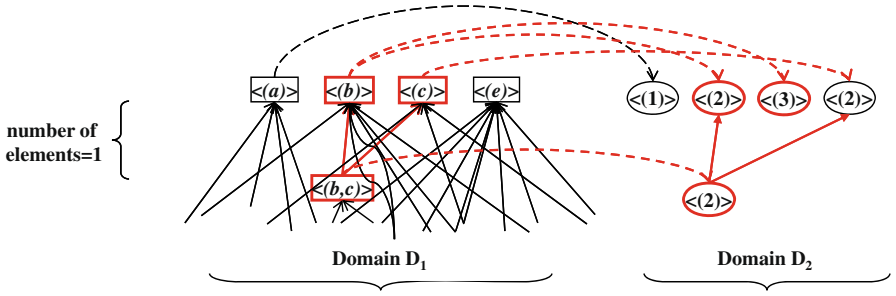

Fig. 6. An example of generating sequential patterns with one element in domain D_2.

each atomic pattern in D_1, there are *interdomain links* representing that these two patterns are able to form multi-domain sequential patterns. Consequently, we have $\left[\begin{smallmatrix}(a)\\(1)\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}(b)\\(2)\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}(b)\\(3)\end{smallmatrix}\right]$, and $\left[\begin{smallmatrix}(c)\\(2)\end{smallmatrix}\right]$ in the above example, and they are obviously also atomic patterns.

Step 2: Derive $(k+1)$*-domain* sequential patterns with one element

This step involves deriving $(k+1)$-domain sequential patterns with one element. Assume that k-domain sequential pattern P across k-domain sequence databases (i.e., $D_1, D_2, \ldots$, and D_k) and that there is only one element in P (i.e., $e(P)=1$). The intradomain links in the lattice structure for domain k can be followed to find two multi-domain sequential patterns (e.g., X and Y, which are the components of P). The corresponding multi-domain sequential patterns in domain $k+1$ are found by traversing interdomain links of X and Y. According to the antimonotone property, if there exists any corresponding sequential patterns of X or Y in domain $k+1$, they must have been discovered due to $X \sqsubseteq P$ and $Y \sqsubseteq P$. Hence, the corresponding sequential patterns of P in domain $k+1$ are generated from the union of all the multi-domain sequential patterns found in domain $k+1$. For example, let $P=\langle(b,c)\rangle$ be a sequential pattern with $e(P)=1$ in D_1 of Table 2. The components of P (i.e., $\langle(b)\rangle$ and $\langle(c)\rangle$) can be found from the intradomain links. Following interdomain links of $\langle(b)\rangle$ and $\langle(c)\rangle$ in Fig. 6, yields the multi-domain sequential patterns in domain D_2 (i.e., $\left[\begin{smallmatrix}(b)\\(2)\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}(b)\\(3)\end{smallmatrix}\right]$ for $\langle(b)\rangle$, and $\left[\begin{smallmatrix}(c)\\(2)\end{smallmatrix}\right]$ for $\langle(c)\rangle$). Consequently, two candidates are generated by union operation: $\left[\begin{smallmatrix}(b)\\(2)\end{smallmatrix}\right] \cup \left[\begin{smallmatrix}(c)\\(2)\end{smallmatrix}\right] = \left[\begin{smallmatrix}(b,c)\\(2)\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}(b)\\(3)\end{smallmatrix}\right] \cup \left[\begin{smallmatrix}(c)\\(2)\end{smallmatrix}\right] = \left[\begin{smallmatrix}(b,c)\\(2,3)\end{smallmatrix}\right]$.

Once the candidate multi-domain sequential patterns are obtained, support values of these patterns are examined by checking their time-instance sets (i.e., $Support\left(\left[\begin{smallmatrix}(\alpha)\\(\beta)\end{smallmatrix}\right]\right) = \left|TIS\left(\left[\begin{smallmatrix}(\alpha)\\(\beta)\end{smallmatrix}\right]\right)\right| = |TIS(\langle(\alpha)\rangle) \cap TIS(\langle(\beta)\rangle)|$). Given a minimum support of 3, since the support values of $\left[\begin{smallmatrix}(b,c)\\(2)\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}(b,c)\\(2,3)\end{smallmatrix}\right]$ are 3 and 2, respectively, only $\left[\begin{smallmatrix}(b,c)\\(2)\end{smallmatrix}\right]$ is frequent. Thus, the lattice structure in domain D_2 contains node $\langle(2)\rangle$, and interdomain links are built between lattice structures in domains D_1 and D_2.

Step 3: Derive $(k+1)$*-domain* sequential patterns with more than one element

After generating atomic patterns and the $(k+1)$-domain sequential patterns with one element in step1 and step 2 respectively, algorithm PropagatedMine can further generate remaining $(k+1)$-domain sequential patterns in a level-by-level manner by referring to the lattice structure in the last domain propagated (i.e., domain D_k). In this step, PropagatedMine starts deriving from those patterns with two elements due to the antimonotone property. The frequent patterns in the upper levels are found from the intradomain links in the lattice structure of D_k, and the corresponding upper level patterns in the

lattice structure of domain D_{k+1} are identified from their interdomain links. Now, the interdomain links of upper level patterns must been established due to the antimonotone property. Before deriving $(k+1)$-domain sequential patterns, it should be determined whether or not to merge the sequential patterns identified in the lattice structure based on their time order. This leads to Definition 5.

Algorithm: PropagatedMine
Input: Sequence databases across n domains $D_1, D_2, \ldots, D_n$, and minimum support δ.
Output: Multi-domain sequential patterns across 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, $i = 2, 3, \ldots, n$
    For each $P \in SP_{i-1}$
      //Step 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 $TIS\left(\begin{bmatrix} P \\ Q \end{bmatrix}\right) = TIS(P) \cap TIS(Q)$.
      End
      //Step 2
      If $e(P) = 1$ **Then Begin**
        Let X and Y be two patterns pointed to by intradomain links of P.
        For each pattern α pointed to by interdomain links of X
          For each pattern β pointed to by interdomain links of Y
            If $Support\left(\begin{bmatrix} \alpha \\ \beta \end{bmatrix}\right) \geqslant \delta$ **Then Begin**
              Construct interdomain links from P to $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$.
              Construct intradomain links from $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ to α and β.
              Append $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ to SP_i.
      End
      //Step 3
      If $e(P) > 1$ **Then Begin**
        Let X and Y be two patterns pointed to by intradomain links of P.
        For each pattern α pointed to by interdomain links of X
          For each pattern β pointed to by interdomain links of Y
            If $Support([(\alpha)(\beta)]) \geqslant \delta$ **Then Begin**
              Construct interdomain links from P to $[(\alpha)(\beta)]$.
              Construct intradomain links from $[(\alpha)(\beta)]$ to α and β.
              Append $[(\alpha)(\beta)]$ to SP_i.
      End
  Output $= SP_n$.
End

Definition 5 (*Concatenate operation of TIS*). Let M and N be two multi-domain sequences, where $TIS(M) = \{\langle TS_1 : l_{11}, l_{12}, \ldots, l_{1e(M)}\rangle, \langle TS_2 : l_{21}, l_{22}, \ldots, l_{2e(M)}\rangle, \ldots, \langle TS_m : l_{m1}, l_{m2}, \ldots, l_{me(M)}\rangle\}$, $TIS(N) = \{\langle TT_1 : k_{11}, k_{12}, \ldots, k_{1e(N)}\rangle, \langle TT_2 : k_{21}, k_{22}, \ldots, k_{2e(N)}\rangle, \ldots, \langle TT_n : k_{n1}, k_{n2}, \ldots, k_{ne(N)}\rangle\}$, and TS_i is the time sequence for $i = 1, 2, \ldots, m$ while TT_j is also time sequence for $j = 1, 2, \ldots, n$. The concatenation of $TIS(M)$ and $TIS(N)$ is denoted as $TIS(M) \cap_< TIS(N) = \{\langle TS_i : l_{i1}, l_{i2}, \ldots, l_{ie(M)}, k_{j1}, k_{j2}, \ldots, k_{je(N)}\rangle\}$, such that $TS_i = TT_j$ and $l_{ie(M)} < k_{j1}$. In other words, $TIS(M) \cap_< TIS(N)$ is the time-instance set of the multi-domain sequence $[M, N]$, $TIS([M, N])$.

For example, given $M = \begin{bmatrix}(a)\\(1)\end{bmatrix}, N = \begin{bmatrix}(b,c)\\(2)\end{bmatrix}$, and the sequence database across two domains in Table 2, where $TIS(M) = \{\langle(T_1)(T_2)(T_3)(T_4) : 1\rangle, \langle(T_5)(T_6)(T_7) : 1\rangle, \langle(T_{10})(T_{12})(T_{13}) : 1\rangle, \langle(T_{21})(T_{22})(T_{23})(T_{24}) : 1\rangle\}$, and $TIS(N) = \{\langle(T_1)(T_2)(T_3)(T_4) : 2\rangle, \langle(T_5)(T_6)(T_7) : 2\rangle, \langle(T_{21})(T_{22})(T_{23})(T_{24}) : 3\rangle\}$. It can be verified that $TIS(\begin{bmatrix}(a) & (b,c)\\(1) & (2)\end{bmatrix}) = TIS(M) \cap_< TIS(N) = \{\langle(T_1)(T_2)(T_3)(T_4) : 1,2\rangle, \langle(T_5)(T_6)(T_7) : 1,2\rangle, \langle(T_{21})(T_{22})(T_{23})(T_{24}) : 1,3\rangle\}$.

Assume that pattern $P \in SP_k$ and $e(P) > 1$. Similar to Step 2, we can obtain the components of P, X and Y, by traversing intradomain links among lattice structures across k domains, and the multi-domain sequential patterns pointed to by their interdomain links can be determined. In light of Definition 5, a concatenate operation is considered rather than generating their union as in Step 2. For example, assume pattern $P = \langle(a)(b,c)\rangle$ in Fig. 7. The intradomain and interdomain links yield $\begin{bmatrix}(a)\\(1)\end{bmatrix}$ and $\begin{bmatrix}(b,c)\\(2)\end{bmatrix}$. Therefore, candidate multi-domain sequential pattern $\begin{bmatrix}(a) & (b,c)\\(1) & (2)\end{bmatrix}$ is generated, as its support value, $Support\left(\begin{bmatrix}(a) & (b,c)\\(1) & (2)\end{bmatrix}\right) = \left|TIS\left(\begin{bmatrix}(a)\\(1)\end{bmatrix}\right) \cap_< TIS\left(\begin{bmatrix}(b,c)\\(2)\end{bmatrix}\right)\right| = 3$.

The above steps allow multi-domain sequential patterns across $(k+1)$-domain sequence databases to be derived from k-domain sequential patterns. Algorithm PropagatedMine iteratively repeats the above three steps until all sequence databases are propagated.

Theorem 1. *Algorithm PropagatedMine is able to mine all multi-domain sequential patterns via lattice structures.*

Proof. Mining frequent itemsets in propagated tables reveals multi-domain atomic patterns across other sequence databases. To prove the correctness of Steps 2 and 3, first let P be a k-domain sequential pattern and P' be a $(k+1)$-domain sequential pattern derived from P, where $e(P') = e(P) = 1$ and $|P'| \geqslant |P| > 1$. In other words, $P' = \begin{bmatrix}P\\Z\end{bmatrix}$, where Z is a frequent itemset in the propagated table $D_{k+1}||_{\langle(P)\rangle}$. Assume that X and Y are parts of P, and $X \cup Y = P$. Hence, in the lattice structure, we have intradomain links from P to X and Y. In addition, there are interdomain links from X and Y to Z', where Z' is the power set of Z and $Z' \neq \emptyset$. Due to the antimonotone property, all multi-domain sequences contained by P' must also be frequent. In other words, both $\begin{bmatrix}X\\Z'\end{bmatrix}$ and $\begin{bmatrix}Y\\Z'\end{bmatrix}$ are frequent. Therefore, the lattice structures can be used to derive all pairs of P and P' while $e(P') = e(P) = 1$. Similarly, when $e(P') = e(P) > 1$, X and Y are parts of P and $TIS(X) \cap_< TIS(Y) = TIS(P)$. Moreover, assume that Z is a frequent itemset in propagated table $D_{k+1}||_{\langle(P)\rangle}$. Clearly, interdomain links exist from X and Y to Z' in domain D_{k+1}, where Z' is the power set of Z and $Z' \neq \emptyset$. The antimonotone property means that all multi-domain sequences contained by P' must also be frequent. This results in both $[(X,Z')]$ and $[(Y,Z')]$ being frequent. This proof indicates that algorithm PropagatedMine is able to mine all multi-domain sequential patterns. □

4.4. Mining relaxed multi-domain sequential patterns

The above three algorithms are utilized in mining strong multi-domain sequential patterns, where all co-occurred events are from all domains required. Strong multi-domain sequential patterns are very restricted since users may have their minds on analyzing the behavior across domains interested by users. In this paper, we further develop some mechanisms for min-

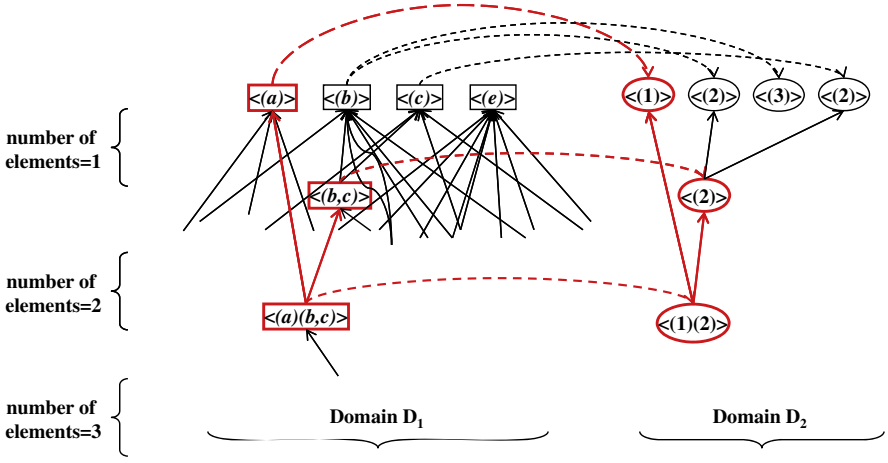

Fig. 7. An example of generating sequential patterns with more than one element in domain D_2.

ing relaxed multi-domain sequential patterns in which in some time slots, some empty sets are allowed. Note that both the Naive algorithm and algorithm IndividualMine could be extended for mining relaxed multi-domain sequential patterns. However, due to the feature of propagation, algorithm PropagatedMine is not able to discover relaxed patterns. In the following, we will discuss how to mine relaxed multi-domain sequential patterns.

Naive algorithm

As pointed out early, given a set of sequence databases, algorithm Naive will join these sequence databases into one multi-domain sequence database. With the proper transformed of multi-domain sequence databases, one could generate a sequence database whose events are from all domains. Thus, existing sequential pattern mining algorithms could be utilized to discover sequential patterns. Note that sequential patterns mined are then represented as the form of multi-domain sequential patterns. Hence, those multi-domain sequential patterns that have some empty sets are directly viewed as relaxed patterns.

Algorithm IndividualMine

Algorithm IndividualMine performs sequential pattern mining algorithms in each sequence database. After generating all sequential patterns in all domains, in the checking phase, algorithm IndividualMine will check and composite candidate multi-domain sequential patterns with the same number of elements. In order to mine relaxed patterns, all possible compositions of multi-domain sequential patterns from sequential patterns of each domain should be enumerated. For example, assume that one i-domain sequential pattern $P = \langle P_1, P_2, \ldots, P_l \rangle$, is selected SP_i and $Q = \langle q_1, q_2, \ldots, q_r \rangle$ is a sequential pattern of domain D_{i+1}. Candidate $(i+1)$-domain sequential patterns generated from P and Q are $\begin{bmatrix} P_1 & P_2 & \ldots & P_l & \\ & & q_1 & q_2 & \ldots & q_r \end{bmatrix}, \ldots, \begin{bmatrix} P_1 & P_2 & \ldots & & P_l & \\ & & & q_1 & q_2 & \ldots & q_r \end{bmatrix}$ and so on. Note that the number of candidate patterns is denoted as $f(r,l)$ which is formulated as follows:

$$f(r,l) = \begin{cases} 1, & \text{if } r = 0 \\ f(l, r-1) + 2\sum_{i=0}^{r-1} f(i, l-1), & \text{otherwise.} \end{cases} \quad (1)$$

Obviously, it could be very large when r and l increase. As expected, we could have a large number of candidate multi-domain sequential patterns, degrading the performance of algorithm IndividualMine.

Algorithm PropagatedMine

By exploring propagation and lattice structures, algorithm PropagatedMine is able to reduce the mining cost. However, algorithm PropagatedMine cannot mine relaxed patterns since propagation needs to obtain time-instance sets of sequential patterns. Empty sets mean that events don't occur and thus there are no any available time information for the empty sets. Thus, it is impossible to derive time-instance sets of empty sets. Consequently, for mining relaxed patterns, algorithm Naive and algorithm IndividualMine should be used.

5. Performance evaluation

To evaluate the performance of our proposed algorithms, we implement a simulation model and conduct extensive experiments. In Section 5.1, the simulation model and synthetic datasets are described. Section 5.2 is devoted to experimental results.

5.1. Simulation model

We modify a well-known data generator in [3] to generate datasets that include multiple domains and the data generator is broadly used in many studies to evaluate mining algorithms proposed [18]. The detailed generation process could be referred to [18]. Some parameters are summarized in Table 7. Explicitly, M denotes the number of domains, D is the number of sequences, C is the average number of elements in a sequence, T is the average number of events in an element and I is the total number of distinct events. The modeling of these parameters are almost the same in [3]. For example, dataset M5D10kC10T5I100 represents that there are 5 domains, each of which contains 10k of sequences, where the average number of elements in a sequence is 10, the average number of items in an element is 5, and the total number of distinct items is 100.

Table 7
Parameters used for the data generator.

Parameter	Description
M	Number of domains
D	Number of sequences
C	Average number of elements within a sequence
T	Average number of items within an element
I	Total number of different items

For the traditional sequential pattern mining, we use algorithm PrefixSpan which is obtained from the IlliMine project (http://illimine.cs.uiuc.edu/). Algorithm PrefixSpan is used in algorithm Naive and the mining phases of both algorithms IndividualMine and PropagatedMine. Our programs are executed in the platform with the hardware as an Intel 2.4-GHz XEON CPU and 3.5 GB of RAM, and the software as FreeBSD 5.0 and GCC 3.2. We use three performance metrics: the execution time, memory consumption and the number of mined patterns to compare the proposed algorithms.

5.2. Experimental results

Several experiments were conducted to evaluate the performance and memory consumption of the three algorithms. Sensitivity analysis on some important parameters, such as the minimum support, the number of sequences, and the number of domains, is conducted.

5.2.1. Impact of the minimum support threshold

We first investigated the performances of three algorithms with the minimum support varied. For the dataset M2D2kC3T4I200, Fig. 8 shows the execution time and the memory consumption of three algorithms. It can be seen in Fig. 8 that the execution time of algorithm IndividualMine and PropagatedMine is reduced as the minimum support increases. This is due to that with a larger minimum support, the number of sequential patterns in sequence databases is smaller. Furthermore, algorithm PropagatedMine significantly outperforms the other two algorithms in terms of execution time, which demonstrates the advantage of exploring propagation and lattice structures in mining multi-domain sequential patterns. On the other hand, when the minimum support was smaller than 1.5%, algorithm IndividualMine was worse than algorithm Naive. The reason is that with a smaller minimum support, a larger number of sequential patterns are mined in each domain. Thus, algorithm IndividualMine needs more time to composite candidate multi-domain sequential patterns and determine their supports. In Naive algorithm, joining operations among sequence databases are costly, which dominates the execution time. As for the memory consumption, algorithm Naive use less memory than algorithms IndividualMine and PropagatedMine. This is due to that both algorithms IndividualMine and PropagatedMine use more memory spaces for storing sequential patterns mined. Algorithm PropagatedMine also needs to store lattice structures, which incurs more memory space than algorithm IndividualMine. On the other hand, algorithm IndividualMine does not need any more memory space for storing sequential patterns. Though algorithm PropagatedMine needs more memory spaces, algorithm PropagatedMine is able to quickly derive multi-domain sequential patterns, which strikes a compromise between memory space and the execution time.

5.2.2. Impact of the number of domains

We next examine the impact of domains on the performance of three proposed algorithms. The experiments were conducted on D1kC2T3I100 (referred to as a smaller dataset) and D1kC3T4I200 (referred to as a larger dataset). With the minimum support as 0.3%, the execution time with its unit as seconds for these proposed algorithms is shown in Table 8 and Table 9. From both tables, it can be seen that all three algorithms have a larger execution time when the number of domains increases. In particular, the execution of algorithm Naive drastically increases the execution time. Both algorithms IndividualMine and PropagatedMine have smaller execution time than algorithm Naive. Furthermore, algorithm PropagatedMine outperforms other algorithms in terms of the execution time, showing the advantage of utilizing propagation to reduce the mining cost. In addition, given a larger dataset with more number of events and larger sequence lengths, the execution

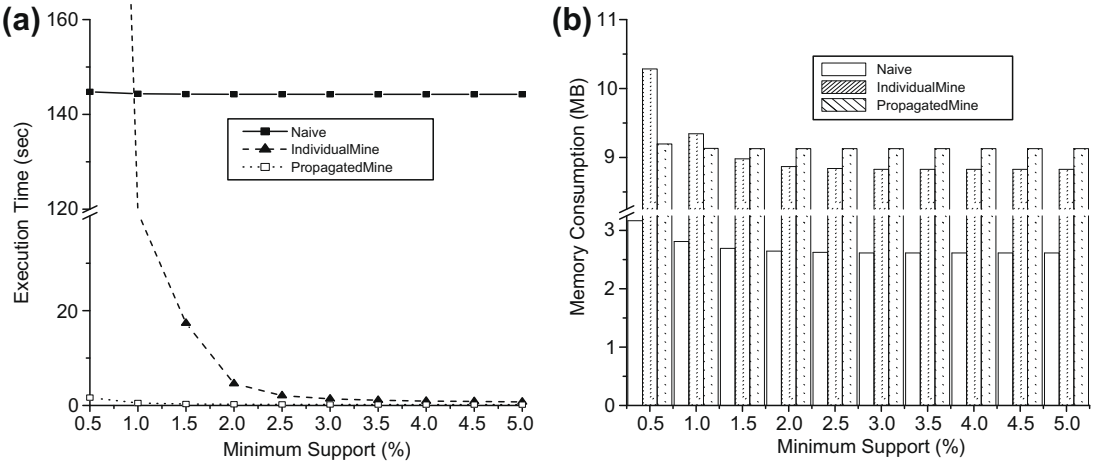

Fig. 8. Execution time of the three algorithms with various minimum support thresholds.

Table 8
Execution time of algorithms Naive, IndividualMine, and PropagatedMine with the number of domains varied on D1kC2T3I100.

Number of domains	2	3	4	5
Naive	5.3	206.7	2513.9	21769.7
IndividualMine	126.3	163.9	180.2	181.1
PropagatedMine	0.4	0.6	0.7	0.7

Table 9
Execution time of algorithms Naive, IndividualMine, and PropagatedMine with the number of domains varied on D1kC2T4I200.

Number of domains	2	3	4	5
Naive	57.1	3065.3	53164.9	379118.5
IndividualMine	1052.1	1192.9	1213.9	1214.4
PropagatedMine	2.1	2.4	2.5	2.5

time of algorithm Naive is worse. On the other hands, algorithm PropagatedMine incurs a smaller execution time than algorithms Naive and IndividualMine, showing the good scalability of algorithm PropagatedMine.

5.2.3. Impact of the number of sequences

Experiments with the number of sequences varied are examined, where the number of sequences is from 1000 to 6000 and other parameters are M2C3T3I200. With a given minimum support was 1%, Fig. 9 shows the execution time of all algorithms. As can be seen in Fig. 9, the execution of all three algorithms increases as the number of sequences increases. Notice that the execution time of algorithm Naive is significantly increasing when the number of sequences is lager than 2000. Thus, to compare algorithms IndividualMine and PropagatedMine, we only put the execution time of algorithms IndividualMine and PropagatedMine. By exploring lattice structures, PropagatedMine should mine only atomic patterns, from which other patterns are derived accordingly. As a result, the execution time of PropagatedMine slightly increases with the number of sequences. Note that the execution time of algorithm PropagatedMine is very smaller compared with algorithms IndividualMine and Naive. However, both algorithms IndividualMine and PropagatedMine need more memory space for storing sequential patterns mined. Thus, it can be seen in Fig. 9 that both algorithms IndividualMine and PropagatedMine have a larger memory consumption than algorithm Naive. This also agrees that algorithm Naive is bounded by execution time, and algorithms IndividualMine and PropagatedMine are bounded by memory spaces.

5.2.4. Impact of the average number of elements within a sequence

In this section, we investigate the performance of Naive, IndividualMine, and PropagatedMine with the average number of elements within a sequence varied. Without loss of generality, the minimum support threshold is set to 1% and the other parameters in the dataset are M2D1kT3I200. Fig. 10 shows experimental results of Naive, IndividualMine, and PropagatedMine. Clearly, the execution time of mining multi-domain sequential patterns increases with the average number of elements within a sequence. Note that algorithm IndividualMine even performs worse than algorithm Naive when the

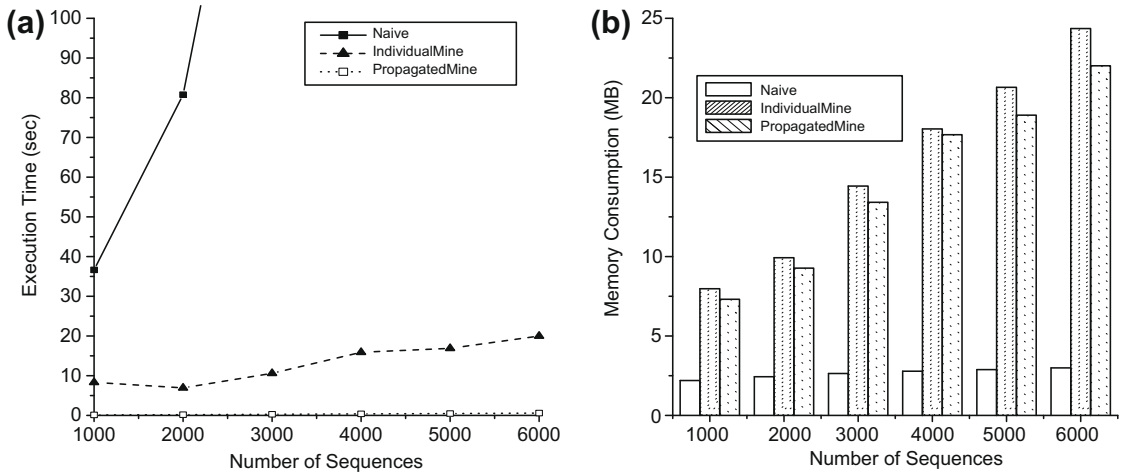

Fig. 9. Performance of Naive, IndividualMine, and PropagatedMine with the number of sequences varied.

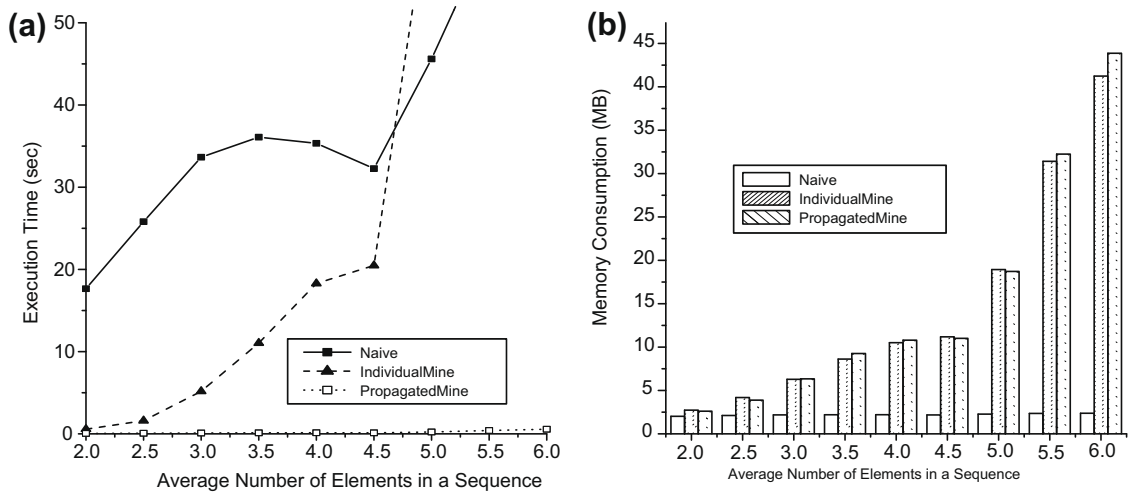

Fig. 10. Performance of Naive, IndividualMine, and PropagatedMine with the average number of elements within a sequence varied.

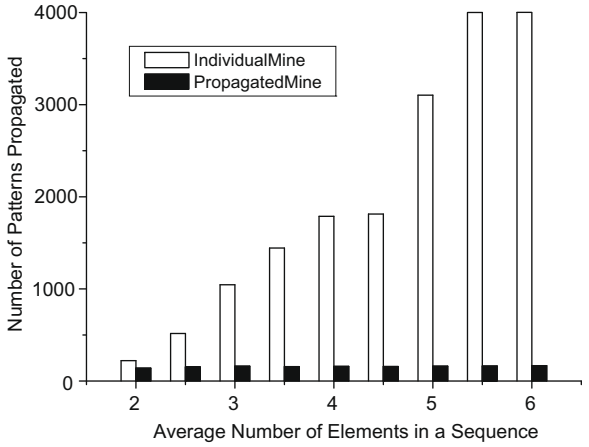

Fig. 11. Number of patterns propagated in IndividualMine and PropagatedMine with the average number of elements within a sequence varied.

average number of elements in a sequence is larger than 4.7. The reason is that IndividualMine mines a large number of sequential patterns in each domain and spends more costs to composite candidate multi-domain sequential patterns. The above observation is also proved in Fig. 11, where algorithm IndividualMine generates a larger number of sequential patterns propagated than algorithm PropagatedMine. Note that, the number of patterns propagated in algorithm IndividualMine is the number of patterns discovered in the starting domain. Fig. 10b also indicates that though algorithm PropagatedMine has a smaller execution time, algorithm PropagatedMine needs more memory spaces to store lattice structure.

5.2.5. Impact of the average number of items within an itemset

The average number of items within an itemset generally impacts on the performance of sequential pattern mining. Thus, we investigate the effect of varying the average number of items within an itemset. The minimum support was set to 1% and we used the dataset M2D1kC3I200. The execution time and memory consumption with the average number of items in an itemset varied are shown in Fig. 12. As can be seen that in Fig. 12, PropagatedMine performs the best in terms of the execution time. When the average number of items in an itemset is smaller, the execution time of IndividualMine is smaller than that of Naive. However, if there is a large number of items within an itemset, IndividualMine performs worse than Native since algorithm IndividualMine has a larger number of patterns mined, which incurs a considerable cost in the checking phase. Fig. 13 demonstrates that PropagatedMine is better than IndividualMine because sequential patterns mined in the starting domain are much smaller than that of algorithm IndividualMine. In algorithm PropagatedMine, only atomic patterns are mined and thus the number of patterns mined in the starting domain is equal to the number of atomic patterns. Consequently, by exploring lattice structures, algorithm PropagatedMine outperforms the other algorithms in terms of the execution time.

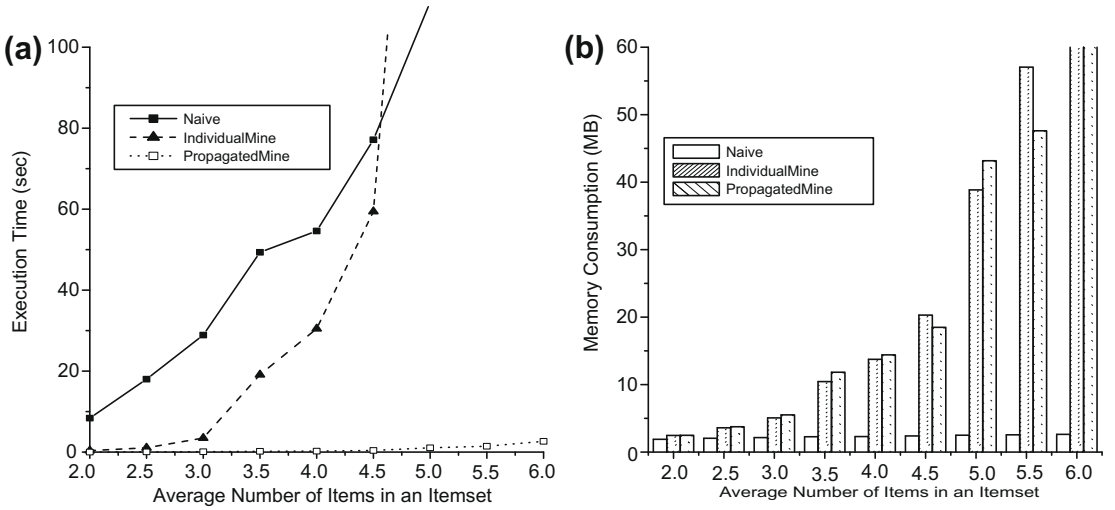

Fig. 12. Performance of Naive, IndividualMine, and PropagatedMine with the average number of items within an itemset varied.

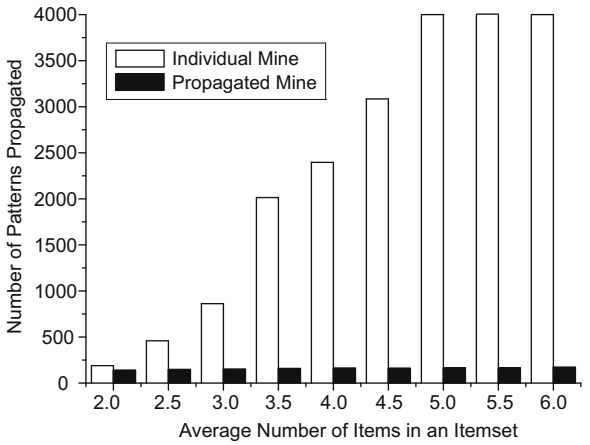

Fig. 13. Number of patterns propagated in IndividualMine and PropagatedMine with the average number of items within an itemset varied.

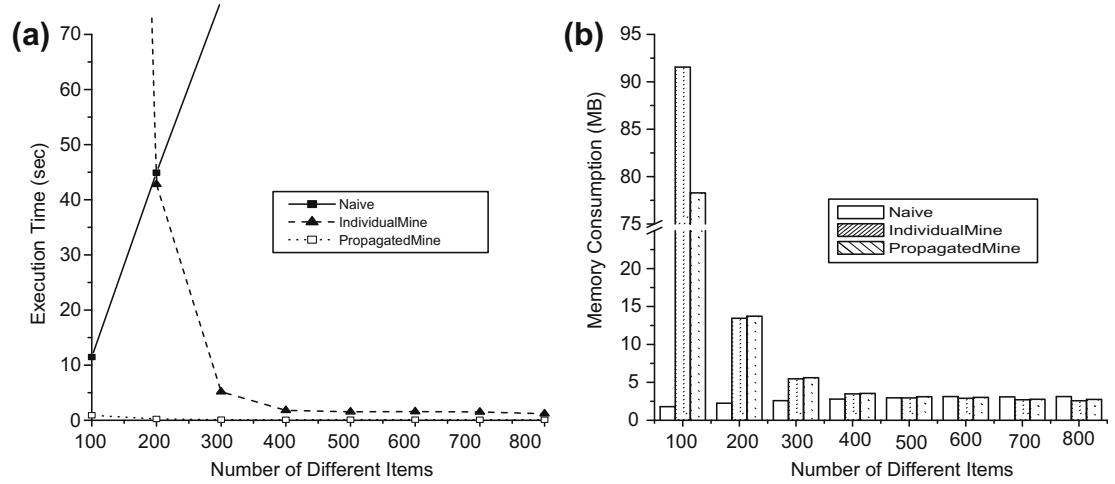

Fig. 14. Performance of Naive, IndividualMine, and PropagatedMine with the number of different items varied.

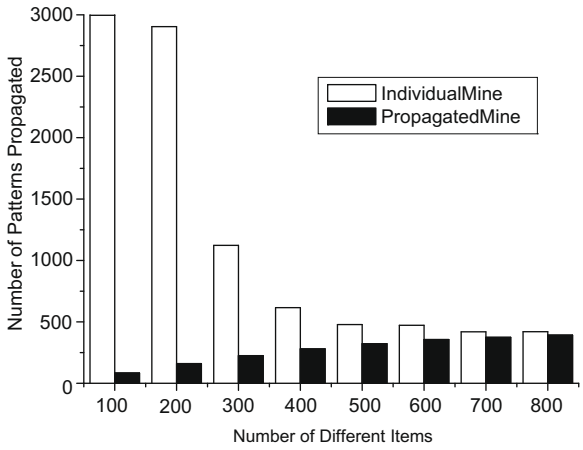

Fig. 15. Number of patterns propagated in IndividualMine and PropagatedMine with the number of different items varied.

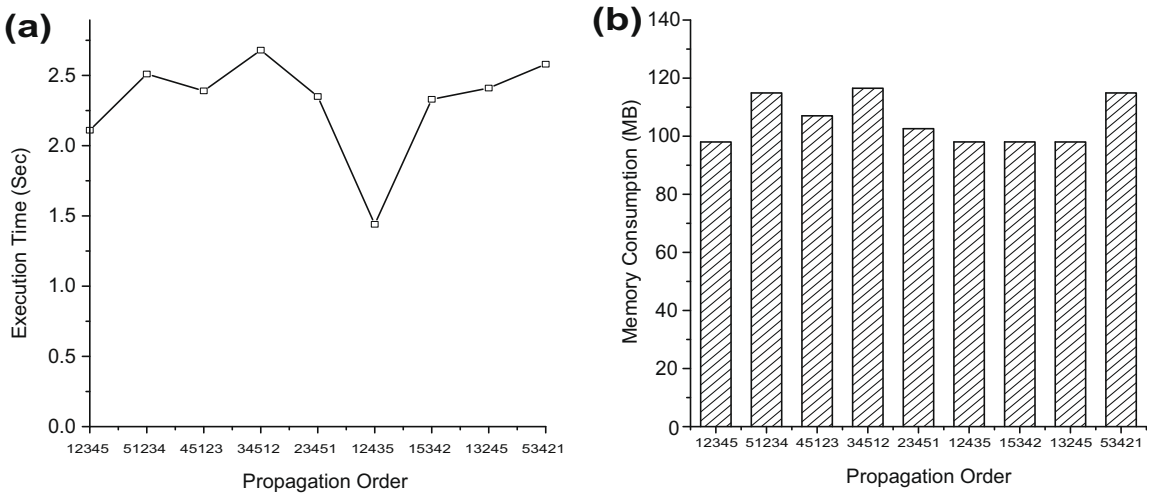

Fig. 16. Performance of PropagatedMine with varied propagation order.

5.2.6. Impact of the number of items

We next investigate the impact of the total number of items, where a minimum support is set to 1% and other parameters are set as M2D1kC3T4. Fig. 14 shows the execution times and memory consumption of Naive, IndividualMine, and PropagatedMine. It can be seen in Fig. 14 that both IndividualMine and PropagatedMine have a smaller execution time than Naive as the number of items increases. When the number of items is larger, the probability of being frequent for each item is smaller with the same setting in D1kC3T4. Fig. 15 depicts the number of patterns with the number of items varied. As can be seen in Fig. 15, PropagatedMine has a smaller number of patterns derived, which demonstrates the advantage of using lattice structures for discovering multi-domain sequential patterns.

5.2.7. Impact of the propagation order for PropagatedMine

Since algorithm PropagatedMine explores propagation on mining multi-domain sequential patterns, we now get insight into the impact of propagation orders on performance of algorithm PropagatedMine. As pointed out early, algorithm PropagatedMine first selects a starting domain and then performs sequential pattern mining. Based on the mining results, a lattice structure is built. Clearly, one should judiciously determine the starting domain in algorithm PropagatedMine. Intuitively, selecting a domain with a smaller number of sequential patterns is good to reduce the size of lattice structures, thereby improving the performance of algorithm PropagatedMine. In this experiment, we conduct experiments on different propagation orders. Fig. 16 shows the execution time of algorithm PropagatedMine with various propagation orders, where the value in the x-axle is the propagation order used. For example, 12435 indicates that the algorithm PropagatedMine starts with D_1, and then propagates to D_2, D_4, D_3 and D_5. As can be seen in Fig. 16, selecting domain D_1 as a starting domain is better

Table 10
Number of sequential patterns mined in each domain.

Domains	D_1	D_2	D_3	D_4	D_5
Number of sequential patterns	24982	25507	28204	27654	28560

since algorithm PropagatedMine has a smaller execution time and memory consumption. This implies that sequential patterns in D_1 has the minimal number of sequential patterns. Table 10 depicts the number of sequential patterns in each domain and the number of sequential patterns in D_1 is the smallest among other domains. Furthermore, in Fig. 16, propagation order 12435 incurs the smallest execution time of algorithm PropagatedMine. This observation gives a guideline in which a good propagation order is determined as an ascending order of the number of sequential patterns in sequence databases. Note that there are many ways (e.g., sampling) to approximate the number of sequential patterns in each domain. Thus, according to the guideline above, one could determine a good propagation order for algorithm PropagatedMine.

6. Conclusions

This paper addresses a novel mining task: the multi-domain sequential pattern mining problem. Multi-domain sequential patterns are of practical interest and use since they clearly reflect the relations of domains hidden in user's behavior. We designed algorithm Naive as a baseline algorithm and two efficient algorithms, IndividualMine and PropagatedMine, to solve this problem. Specifically, in algorithm IndividualMine, each domain individually performs sequential pattern mining and then candidate multi-domain sequential patterns are generated by combining all mined sequential patterns in each domain. Finally, by checking the time-instance sets of candidate multi-domain sequential patterns, the multi-domain sequential patterns are discovered without scanning databases. In order to reduce the mining cost of discovering sequential patterns in each domain, algorithm PropagatedMine first mines sequential patterns in a starting domain. Propagated tables are then constructed to discover the candidate multi-domain sequential patterns. Note that by using propagated tables, only sequential patterns that are likely to form multi-domain sequential patterns are extracted. Algorithm PropagatedMine further explores lattice structures to reduce the number of patterns propagated. A comprehensive experimental study is conducted and experimental results show that both algorithms IndividualMine and PropagatedMine are able to quickly mine multi-domain sequential patterns compared with algorithm Naive. By exploring propagation and lattice structures, algorithm PropagatedMine outperforms other algorithms in terms of execution times.

References

[1] Rakesh Agrawal, Tomasz Imielinski, Arun N. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the 1993 ACM International Conference on Management of Data (SIGMOD), 1993, pp. 207–216.
[2] Rakesh Agrawal, Ramakrishnan Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 1994 International Conference on Very Large Data Bases (VLDB), 1994, pp. 487–499.
[3] Rakesh Agrawal, Ramakrishnan Srikant, Mining sequential patterns, in: Proceedings of the 1995 IEEE International Conference on Data Engineering (ICDE), 1995, pp. 3–14.
[4] Jay Ayres, Jason Flannick, Johannes Gehrke, Tomi Yiu, Sequential pattern mining using a bitmap representation, in: Proceedings of the 2002 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2002, pp. 429–435.
[5] Mete Celik, Shashi Shekhar, James P. Rogers, James A. Shine, Jin Soung Yoo, Mixed-drove spatio-temporal co-occurence pattern mining: a summary of results, in: Proceedings of the 2006 IEEE International Conference on Data Mining (ICDM), 2006, pp. 119–128.
[6] Gong Chen, Xindong Wu, Xingquan Zhu, Sequential pattern mining in multiple streams, in: Proceedings of the 2005 IEEE International Conference on Data Mining (ICDM), 2005, pp. 585–588.
[7] Hong Cheng, Xifeng Yan, Jiawei Han, IncSpan: incremental mining of sequential patterns in large database, in: Proceedings of the 2004 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2004, pp. 527–532.
[8] Ding-Ying Chiu, Yi-Hung Wu, Arbee L.P. Chen, An efficient algorithm for mining frequent sequences by a new strategy without support counting, in: Proceedings of the 2004 IEEE International Conference on Data Engineering (ICDE), 2004, pp. 375–386.
[9] Chung-Wen Cho, Yi-Hung Wu, Arbee L.P. Chen, Effective database transformation and efficient support computation for mining sequential patterns, in: Proceedings of the 2005 International Conference Database Systems for Advanced Applications (DASFAA), 2005, pp. 163–174.
[10] Themis P. Exarchos, Markos G. Tsipouras, Costas Papaloukas, Dimitrios I. Fotiadis, A two-stage methodology for sequence classification based on sequential pattern mining and optimization, Data and Knowledge Engineering 66 (3) (2008) 467–487.
[11] Minos N. Garofalakis, Rajeev Rastogi, Kyuseok Shim, SPIRIT: sequential pattern mining with regular expression constraints, in: Proceedings of the 1999 International Conference on Very Large Data Bases (VLDB), 1999, pp. 223–234.
[12] Valerie Guralnik, George Karypis, Parallel tree-projection-based sequence mining algorithms, Parallel Computing 30 (4) (2004) 443–472.
[13] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, Meichun Hsu, FreeSpan: frequent pattern-projected sequential pattern mining, in: Proceedings of the 2000 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2000, pp. 355–359.
[14] Jiawei Han, Jian Pei, Yiwen Yin, Mining frequent patterns without candidate generation, in: Proceedings of the 2000 ACM International Conference on Management of Data (SIGMOD), 2000, pp. 1–12.
[15] Jen-Wei Huang, Chi-Yao Tseng, Jian-Chih Ou, Ming-Syan Chen, Pisa: progressive mining of sequential patterns, in: Proceedings of the ACM 2006 International Conference on Information and Knowledge Management (CIKM), 2006, pp. 850–851.
[16] Kuo-Yu Huang, Chia-Hui Chang, Jiun-Hung Tung, Cheng-Tao Ho, COBRA: closed sequential pattern mining using bi-phase reduction approach, in: Proceedings of the 2006 International Conference on Data Warehousing and Knowledge Discovery (DaWaK), 2006, pp. 280–291.
[17] Hye-Chung Kum, Joong Hyuk Chang, Wei Wang, Sequential pattern mining in multi-databases via multiple alignment, Data Mining and Knowledge Discovery 12 (2–3) (2006) 151–180.
[18] Hye-Chung Kum, Joong Hyuk Chang, Wei Wang, Benchmarking the Effectiveness of Sequential Pattern Mining Methods, Data and Knowledge Engineering 60 (1) (2007) 30–50.

[19] Neal Lesh, Mohammed Javeed Zaki, Mitsunori Ogihara, Mining features for sequence classification, in: Proceedings of the 1999 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 1999, pp. 342–346.
[20] Zhung-Xun Liao, Wen-Chih Peng, Exploring lattice structures in mining multi-domain sequential patterns, in: Proceedings of the 2007 International Conference on Scalable Information Systems (InfoScale), 2007, pp. 334–339.
[21] Zhung-Xun Liao, Wen-Chih Peng, Xing-Yuan Hu, Mining multi-domain sequential patterns, in: Workshop on Software Engineering, Databases, and Knowledge Discovery, International Computer Symposium (ICS), 2006, pp. 334–339.
[22] Florent Masseglia, Pascal Poncelet, Maguelonne Teisseire, Incremental mining of sequential patterns in large databases, Data and Knowledge Engineering 46 (1) (2003) 97–121.
[23] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, Meichun Hsu, PrefixSpan: mining sequential patterns by prefix-projected growth, in: Proceedings of the 2001 IEEE International Conference on Data Engineering (ICDE), 2001, pp. 215–224.
[24] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Meichun Hsu, Mining sequential patterns by pattern-growth: the PrefixSpan approach, IEEE Transactions on Knowledge and Data Engineering 16 (11) (2004) 1424–1440.
[25] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, Umeshwar Dayal, Multi-dimensional sequential pattern mining, in: Proceedings of the 2001 ACM International Conference on Information and Knowledge Management (CIKM), 2001, pp. 81–88.
[26] Pierre-Yves Rolland, FlExPat: flexible extraction of sequential patterns, in: Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM), 2001, pp. 481–488.
[27] Ramakrishnan Srikant, Rakesh Agrawal, Mining sequential patterns: generalizations and performance improvements, in: Proceedings of the 1996 International Conference on Extending Database Technology (EDBT), 1996, pp. 3–17.
[28] Petre Tzvetkov, Xifeng Yan, Jiawei Han, TSP: mining Top-K closed sequential patterns, Knowledge Information System 7 (4) (2005) 438–457.
[29] Jianyong Wang, Jiawei Han, BIDE: efficient mining of frequent closed sequences, in: Proceedings of the 2004 IEEE International Conference on Data Engineering (ICDE), 2004, pp. 79–90.
[30] Xifeng Yan, Jiawei Han, Ramin Afshar, CloSpan: mining closed sequential patterns in large databases, in: Proceedings of the 2003 SIAM International Conference on Data Mining (SDM), 2003.
[31] Jiong Yang, Wei Wang, Philip S. Yu, Jiawei Han, Mining long sequential patterns in a noisy environment, in: Proceedings of the 2002 ACM International Conference on Management of Data (SIGMOD), 2002, pp. 406–417.
[32] Chung-Ching Yu, Yen-Liang Chen, Mining sequential patterns from multidimensional sequence data, IEEE Transactions on Knowledge and Data Engineering 17 (1) (2005) 136–140.
[33] Mohammed Javeed Zaki, SPADE: an efficient algorithm for mining frequent sequences, Machine Learning 42 (1/2) (2001) 31–60.
[34] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, New algorithms for fast discovery of association rules, in: Proceedings of the 1997 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 1997, pp. 283–286.

Wen-Chih Peng was born in Hsinchu, Taiwan, R.O.C in 1973. He received the BS and MS degrees from the National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in Electrical Engineering from the National Taiwan University, Taiwan, R.O.C in 2001. Currently, he is an assistant professor at the department of Computer Science, National Chiao Tung University, Taiwan. Prior to joining the department of Computer Science and Information Engineering, National Chiao Tung University, he was mainly involved in the projects related to mobile computing, data broadcasting and network data management. Dr. Peng serves as PC members in several prestigious conferences, such as IEEE International Conference on Data Engineering (ICDE), Pacific Asia Knowledge Discovering and Mining (PAKDD) and Mobile Data Management (MDM). His research interests include mobile computing, network data management and data mining. He is a member of IEEE.

Zhung-Xun Liao received the BS degree from the National Taiwan Normal University, in 2002, and the MS degree in Computer Science from the National Chengchi University, Taiwan, in 2004. Currently, he is a Ph.D. candidate at the department of Computer Science, National Chiao Tung University, Taiwan. His research interests include data mining and databases.