Applied Intelligence (2018) 48:1327–1343
https://doi.org/10.1007/s10489-017-1123-9

Mining constrained inter-sequence patterns: a novel approach to cope with item constraints

Tuong Le[1,2] · Anh Nguyen[3] · Bao Huynh[4,5] · Bay Vo[6] · Witold Pedrycz[7,8,9]

Published online: 9 February 2018

Abstract
Data mining has become increasingly important in the Internet era. The problem of mining inter-sequence pattern is a sub-task in data mining with several algorithms in the recent years. However, these algorithms only focus on the transitional problem of mining frequent inter-sequence patterns and most frequent inter-sequence patterns are either redundant or insignificant. As such, it can confuse end users during decision-making and can require too much system resources. This led to the problem of mining inter-sequence patterns with item constraints, which addressed the problem when end-users only concerned the patterns contained a number of specific items. In this paper, we propose two novel algorithms for it. First is the ISP-IC (Inter-Sequence Pattern with Item Constraint mining) algorithm based on a theorem that quickly determines whether an inter-sequence pattern satisfies the constraints. Then, we propose a way to improve the strategy of ISP-IC, which is then applied to the *i*ISP-IC algorithm to enhance the performance of the process. Finally, *pi*ISP-IC, a parallel version of *i*ISP-IC, will be presented. Experimental results show that *pi*ISP-IC algorithm outperforms the post-processing of the-state-of-the-art method for mining inter-sequence patterns (EISP-Miner), ISP-IC, and *i*ISP-IC algorithms in most of the cases.

Keywords Data mining · Pattern mining · Inter-sequence pattern mining · Constraint mining · Parallel mining

1 Introduction

With the rapid growth of the Internet, the volume of data has become massive. Analyzing data to reveal knowledge that can be applied to intelligent systems constitutes a challenging problem [3]. Therefore, data mining, the crucial step for extracting knowledge, has been extensively studied. There are many types of systems with numerous types of data such as transaction databases, sequence databases, and streaming databases. Then, many problems have been

✉ Anh Nguyen
nguyenanhvn9@gmail.com

Tuong Le
lecungtuong@tdt.edu.vn

Bao Huynh
huynhquocbao@tdt.edu.vn

Bay Vo
bayvodinh@gmail.com

Witold Pedrycz
wpedrycz@ualberta.ca

[1] Division of Data Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam

[2] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

[3] Institute of Research and Development, Duy Tan University, Da Nang, Vietnam

[4] Center for Applied Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

[5] Faculty of Electrical Engineering and Computer Science, VŠB-Technical University of Ostrava, Ostrava-Poruba, Czech Republic

[6] Faculty of Information Technology, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam

[7] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada

[8] Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah, Saudi Arabia

[9] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

proposed such as pattern mining Yun and Ryu [35]; Salehi et al. [21]; Saif-Ur-Rehman et al. [20]; Yun and Kim [37]; Ryang and Yun [38]; Kim and Yun [39]; Ryang et al. [40]; Kieu et al. [41]; Vo et al. [27] and its applications Liao and Chen [13]; Scalmato et al. [22]; Wright et al. [30]; Lin et al. [14], association rule mining, and classification. Mining frequent sequential patterns, the sub-sequential patterns in a sequence database that satisfy the given minimum support threshold (*minsup*), which are used in recommendation systems Salehi et al. [21], deep order-preserving sub-matrix problem solutions Xue et al. [31], biological sequence mining Liao and Chen [13] and smart health services Jung and Chung [9]. Recently, Jeyabharathi andShanthi [8] proposed a new technique for protein sequence database mining with hybrid frequent pattern mining algorithm. The proposed algorithm ensures the effective extraction of frequent patterns with the optimization of resource constraints. In the recent years, many algorithms for mining frequent sequential patterns Ayres et al. [1]; Gouda et al. [6]; Kaneiwa and Kudo [7]; Yen and Lee [32]; Lin et al. [43, 44]; Pei et al. [19], sequential closed patterns Tran, Le and Vo [23]; Zhang et al. [45], sequential rules Pham et al. [18], closed weighted sequential patterns Yun et al. [33], weighted approximate sequential pattern Yun et al. [34], important sequential patterns Yun and Ryu [35], and high utility-probability sequential patterns Zhang et al. [42] from sequence databases have been proposed.

Besides, the problem of mining inter-transaction (closed) patterns has been attracted a lot of attention with many algorithms such as EH-Apriori and EA-Apriori Lu et al. [16], FITI Tung et al. [24], ITP-Miner Lee andWang [12], and ICMiner Lee et al. [11]. Although sequential patterns and inter-transaction patterns include several itemsets across several transactions, the ordered relationships among items in a transaction are not considered. However, sequence database usually exhibits an ordered relationship among items (or itemsets) in a transaction. Therefore, inter-sequence pattern Wang and Lee [28]; Vo et al. [25] and closed inter-sequence pattern Wang et al. [29]; Le et al. [10] mining are then proposed which are more general models than those of sequential and inter-transaction pattern mining.

In essence, the end users usually consider a small number of patterns that contain at least one item from a set of given item constraints [2]. For example, when analyzing the sequence data-bases of supermarkets, there are a lot of inter-sequence patterns. However, at a time, managers only concerned the patterns contained a number of specific items. Therefore, the problem of mining inter-sequence patterns with item constraints requires thorough studies. However, this problem has not been previously considered. We propose two novel algorithms for fast mining inter-sequence patterns with item constraints. The main contributions of this article are as follows. First, the problem of mining inter-sequence patterns with item constraints is introduced. Second, we develop a theorem to reduce the time of determining whether a candidate sequence satisfies the constraints. The ISP-IC (Inter-Sequence Pattern with Item Constraint mining) algorithm based on the above theorem for mining inter-sequence patterns with item constraints is then proposed. Third, a lemma and the second algorithm, named *i*ISP-IC, are proposed. Then, we present a parallel version of *i*ISP-IC, named *pi*ISP-IC algorithm. Final, experiments are conducted to show the effectiveness of the *pi*ISP-IC in terms of mining time and memory usage compared to the post-processing of the-state-of-the-art method for mining inter-sequence patterns (EISP-Miner) which is called by POST-EISP-Miner, ISP-IC, and *i*ISP-IC algorithms.

The paper is organized as follows. Section 2 presents the related works. Then Section 3 summaries the basic concept of inter-sequence patterns, DBV structure, and the problem of patterns with constraints. Section 4 proposes three algorithms for mining inter-sequence patterns with item constraints attached with advanced examples. Section 5 presents the experimental results. The conclusions and future works are given in Section 6.

2 Related works

Inter-sequence patterns demonstrate the anti-monotone Apriori property (if any $k-$pattern is infrequent, then all its $(k+1)$-super-patterns are infrequent) Wang and Lee [28]. Based on this property, Wang and Lee proposed the M-Apriori algorithm for mining inter-sequence patterns. However, the large numbers of candidates and database scans are the cause for decreasing the performance of this algorithm. The authors proposed a two-phase-algorithm named EISP-Miner (Enhanced Inter-Sequence Pattern Miner) for finding a complete set of frequent inter-sequence patterns. The first phase is to find 1-patterns and converts the original sequences into a structure for each 1-pattern, called a pattern-list which stores two pieces of information including a frequent pattern and a list of locations. The second phase uses an inter-sequence pattern tree (ISP-tree) to enumerate all frequent inter-sequence patterns by joining pattern-lists in a depth-first search manner. In 2012, Vo et al. proposed DBV-ISP algorithm, an enhanced version of EISP-Miner algorithm in which dynamic bit vector (DBV) is used instead of pattern-list to improve the performance. However, we cannot use these algorithms for mining inter-sequence patterns with item constraints because they have to find

all the inter-sequence patterns and then filter these patterns to identify the patterns satisfying the item constraints. Therefore, it is necessary to study an approach for mining inter-sequence patterns with item constraints.

3 Basic concepts

3.1 Inter-sequence pattern mining

Consider the set of items $t = \{u_1, u_2, u_3, \ldots, u_m\}$, where u_i is an item ($1 \leq i \leq m$). A sequence $s = \langle t_1, t_2, t_3, \ldots, t_n\rangle$ is the set of ordered itemsets, where $t_i \subseteq t$ ($1 \leq i \leq n$) is an itemset. A sequence database $\Phi = \{s_1, s_2, s_3, \ldots, s_{|\Phi|}\}$, where $|\Phi|$ is the number of sequences in D and s_i ($1 \leq i \leq |\Phi|$) is a tuple $\langle DAT, Sequence\rangle$, where DAT is the property of s_i used to describe the contextual information. Table 1 shows an example sequence database (Φ_{Ex}). This database is used throughout the paper.

Let d_1 and d_2 be two DAT values associated with sequences s_1 and s_2, respectively. Let d_1 be the reference point. $[d_2 - d_1]$ is the span between s_2 and s_1. The sequence s_2 at domain attribute (DAT) d_2 with respect to d_1, denoted by $s_2[d_2 - d_1]$, is called an e-sequence (extended sequence). For example, with Φ_{Ex} shown in Table 1, let the first sequence be the reference point. Then, the e-sequence of the second transaction is $\langle C(ABC)A\rangle[1]$.

Let $s[d] = \langle t_1, t_2, t_3, \ldots, t_m\rangle[d]$ be a sequence where t_i is an itemset ($1 \leq i \leq m$) and $[d]$ is the span of s. The t_i associated with $[d]$ is defined as an extended itemset (e-itemset), denoted by $\langle t_i\rangle[d]$. If $t_i = (u_1, u_2, u_3, \ldots, u_n)$, where each u_i is the items ($1 \leq i \leq n$), then the u_i associated with $[d]$ is defined as an extended item (e-item), denoted by $(u_i)[d]$. For example, $\langle C(ABC)A\rangle[1]$ has 3 e-itemsets ($\langle C\rangle[1]$, $\langle (ABC)\rangle[1]$, and $A[1]$) and 3 e-items ($(C)[1]$, $(A)[1]$, and $(B)[1]$).

Table 1 Example sequence database (Φ_{Ex})

Dat	*Sequences*
1	$\langle C(AB)\rangle$
2	$\langle C(ABC)A\rangle$
3	$\langle AD\rangle$
4	$\langle ABD\rangle$
5	$\langle AC\rangle$
6	$\langle BCD\rangle$
7	$\langle (AB)C\rangle$

Table 2 Megasequences of Φ_{Ex} with $maxspan = 1$

Dat	*Megasequences*
1	$\langle C(AB)\rangle[0]\ \langle C(ABC)A\rangle[1]$
2	$\langle C(ABC)A\rangle[0]\langle AD\rangle[1]$
3	$\langle AD\rangle[0]\ \langle ABD\rangle[1]$
4	$\langle ABD\rangle[0]\ \langle AC\rangle[1]$
5	$\langle AC\rangle[0]\langle BCD\rangle[1]$
6	$\langle BCD\rangle[0]\ \langle (AB)C\rangle[1]$
7	$\langle (AB)C\rangle[0]$

Given the set of k sequences $\langle d_1, s_1\rangle$, $\langle d_2, s_2\rangle, \ldots, \langle d_k, s_k\rangle$ in Φ, $\Psi = s_1[0] \cup s_2[d_2 - d_1] \cup \ldots \cup s_k[d_k - d_1]$ is called a mega-sequence with $k \geq 1$. In a mega-sequence, the span between DAT of the first transaction and that of the last transaction have to be less than or equal to $maxspan$ (i.e., $d_k - d_1 \leq maxspan$ in Ψ). For Φ_{Ex} shown in Table 1, with $maxspan = 1$ and $Dat = 1$ as the reference point, the list of mega-sequences is shown in Table 2.

Consider the pattern $\beta = (u_1)[v_1], (u_2)[v_2], \ldots, (u_n)[v_n]$, where $(u_i)[v_i]$ ($1 \leq i \leq n$) is an e-item (extended item). The number of e-items in β is called the length of β and a sequence with length k is denoted as a k-sequence. There are three types of relationship between two e-items $(u_i)[v_i]$ and $(u_{i+1})[v_{i+1}]$ in β ($1 \leq i < n$). **(1) Itemset extension**: if $v_i = v_{i+1}$and $(u_i u_{i+1})$ is an itemset, then $(u_{i+1})[v_{i+1}]$ is an itemset extension ($+_I$) of $(u_i)[v_i]$. **(2) Sequence extension**: if $v_i = v_{i+1}$and $\langle u_i u_{i+1}\rangle$ is a sequence, then $(u_{i+1})[v_{i+1}]$ is a sequence extension ($+_S$) of $(u_i)[v_i]$. **(3) Inter extension**: if $v_i < v_{i+1}$, then $(u_{i+1})[v_{i+1}]$ is an inter extension ($+_T$) of $(u_i)[v_i]$.

For example, in Φ_{Ex}, $\langle BCD\rangle[0]\ \langle (AB)C\rangle[1]$ is a 6-sequence. $(B)[0]$ is an itemset extension of $(CD)[0]$. $(B)[0]$ is an inter extension of $(C)[1]$ and $AB[1]$ is a sequence extension of $(C)[1]$. In other words, $(B)[0] +_I (CD)[0] = \langle (BCD)\rangle[0]$, $(AB)[0] +_S (C)[0] = \langle (AB)C\rangle[0]$, and $(B)[0] +_T (C)[1] = \langle B\rangle[0]\langle C\rangle[1]$.

Given two sequences, $s = \langle s_1, s_2, \ldots, s_n\rangle$ and $s' = \langle s'_1, s'_2, \ldots, s'_m\rangle$ with $n \leq m$, s is the sub-sequence of s' if and only if $\exists n, j_1, j_2, \ldots, j_n$ such that (1) $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ and (2) $s_1 \subseteq s'_{j_1}$, $s_2 \subseteq s'_{j_2}$, $\ldots$, $s_n \subseteq s'_{j_n}$. For example, a sequence $\langle A(BC)DF\rangle$ is the subsequence of $\langle A(ABC)(AC)D(CF)\rangle$, but it is not a subsequence of $\langle (ABC)(AC)D(CF)\rangle$. Consider two patterns $\alpha = s_1[i_1], s_2[i_2], \ldots, s_n[i_n]$ and $\beta = s'_1[j_1], s'_2[j_2], \ldots, s'_m[j_m]$, with $1 \leq n \leq m$. α is a subsequence of β if and only if there exist n e-sequences denoted by $s'_{k1}[j_{k1}], s'_{k2}[j_{k2}], \ldots,$

$s'_{kn}[j_{kn}]$ in β, such that $i_1 = j_{k1}$ and s_1 is a subsequence of s'_{k1}; $i_2 = j_{k2}$ and s_2 is a sub-sequence of s'_{k2};…; $i_n = j_{kn}$ and s_n is a sub-sequence of s'_{kn}. In this case, β is a super-pattern of α. For example, both $\langle D\rangle[0]\langle BAC\rangle[3]$ and $\langle(AD)\rangle[0]\langle(AC)C\rangle[1]$ are sub-sequences of $\langle D(AD)\rangle[0]$ $\langle B(AC)C\rangle[1]\langle BACC\rangle[3]$.

Definition 1 Let $a = (i_m)[x]$ and $b = (i_n)[y]$ be 2 e-items. $a = b$ if and only if $(i_m = i_n) \wedge (x = y)$ and $a < b$, if and only if $x < y$ or $((x = y) \wedge (i_m < i_n))$.

For example, $(C)[1] = (C)[1]$, $(B)[1] < (A)[2]$, and $(A)[1] < (C)[1]$.

Definition 2 Let p be a pattern. Function $sub_{ij}(p)$ is defined as $(j\text{-}i+1)$ subset e-items of p from position i to position j.

For example, $sub_{1,4}(\langle(AD)\rangle[0]\langle(AC)C\rangle[1]) = \langle(AD)\rangle[0]$ $\langle(AC)\rangle[1]$ and $sub_{5,5}(\langle(AD)\rangle[0]\ \langle(AC)C\rangle[1]) = (C)[1]$.

Definition 3 Let $\alpha = \langle u\rangle[0]$ and $\beta = \langle v\rangle[0]$ be two frequent 1-patterns. α is joinable to β in any instance. There are three types of join operation: **(1) itemset extension**: $\alpha \cup_I \beta = \{\langle(uv)\rangle[0]\} \,|\{\langle(uv)\rangle[0]\}$; **(2) sequence extension**: $\alpha \cup_S \beta = \{\langle uv\rangle[0]\}$; **(3) inter extension**: $\alpha \cup_T \beta = \{\langle u\rangle[0]\langle v\rangle[x] \,|1 \leq x \leq maxspan\}$.

For example, let $maxspan = 2$, $\langle A\rangle[0] \cup_I \langle B\rangle[0] = \langle(AB)\rangle[0]$; $\langle A\rangle[0] \cup_S \langle B\rangle[0] = \langle AB\rangle[0]$ and $\langle A\rangle[0] \cup_T \langle B\rangle[0] = \{\langle A\rangle[0]\langle B\rangle[1], \langle A\rangle[0]\langle B\rangle[2]\}$.

Definition 4 Let α and β be 2 frequent k-patterns, where $k > 1$, $sub_{k,k}(\alpha) = (u)[i]$, and $sub_{k,k}(\beta) = (v)[j]$. α is joinable to β if $sub_{1,k-1}(\alpha) = sub_{1,k-1}(\beta)$ and $i \leq j$, which generates three types of join operations: (1) itemset extension: $\alpha \cup_I \beta = \{\alpha +_I (v)[j] \,|(i = j) \wedge (u < v)\}$; (2) sequence extension: $\alpha \cup_S \beta = \{\alpha \cup +_S (v)[j] \,|(i = j)\}$; (3) inter extension: $\alpha \cup_T \beta = \{\alpha +_T (v)[j] \,|(i < j)\}$.

For example, $\langle AB\rangle[0] \cup_I \langle AC\rangle[0] = \langle A(BC)\rangle[0]$, $\langle AB\rangle[0]$ $\cup_S \langle AC\rangle[0] = \langle ABC\rangle[0]$, and $\langle AB\rangle[0] \cup_T \langle A\rangle[0]\langle C\rangle[2] = \langle AB\rangle[0]\langle C\rangle[2]$.

3.2 DBV-PatternList data structure

DBV structure is based on the BitTable structure including two elements. (1) Start position: the position of the first non-zero byte in the bit vector; (2) Bit-vector: the list of bytes after removing all zero bytes at the beginning. For example, give a BitTable "000011", the DBV is { Start: 5, Bit-vector: "11"} .

Using DBV structure reduces the memory usage and computational operations in the intersection between two-bit vectors. Using a look-up table Vo et al. [26], the algorithm traverses the DBV once to determine the support of its pattern and therefore the complexity is $O(n)$ where n is the number of elements in its Bit-vector.

Based on the DBV concept, Vo et al. [26] proposed the DBV-PatternList structure that combines DBV and PatternList. The structure includes: (1) Sequence: storage information of the sequence; (2) Block sequence: one DBV and a list of transaction positions corresponding to each sequence. For example, in Φ_{Ex}, consider the structure of sequence $\langle A\rangle[0]$, where A appears at sequences 1, 2, 3, 4, 5, and 7. DBV-PatternList of $\langle A\rangle[0]$ is shown in Fig. 1.

In Fig. 1b, the structure of a DBV-PatternList contains a number of block sequences. Each block sequence is placed in a cell containing information regarding the position of sequences that appear in the sequence database and the position of the sequence in each transaction. For detail, A appears at sequences 1, 2, 3, 4, 5, and 7, so, bit vector of A is 11111010 and therefore, DBV of A is {15, 10} (note that we use 4 bits for an integer). To find the appropriate extended sequence based on the sequence extension operation, the intersection operation has to be carried out on each block sequence. In Fig. 1, PatternList of $\langle A\rangle[0]$ requires 26 bytes, whereas DBV-PatternList of $\langle A\rangle[0]$ requires only 20 bytes.

3.3 Mining patterns with item constraints

Various strategies have been proposed for mining frequent pattern with itemset constraints. There are three main approaches including post-processing, pre-processing, and constrained patterns filtering. Post-processing approaches for mining patterns with item constraints firstly mine patterns and then check them against the constraints Ng

Fig. 1 Structures of (**a**) PatternList and (**b**) DBV-PatternList of ⟨A⟩[0]

Sequence	⟨A⟩[0]					
DAT	1	2	3	4	5	7
	↓	↓	↓	↓	↓	↓
Transaction position	2	2 3	1	1	1	1

(a)

Sequence	⟨A⟩[0]					
Start	1					
DBV	15				10	
	↓	↓	↓	↓	↓	↓
Transaction position	2	2 3	1	1	1	1

(b)

et al. [17]; Lin et al. [15]. Pre-processing approaches firstly restrict the source dataset to records that contain the constraints and then find patterns in the filtered dataset Lin et al. [15]. Constrained patterns filtering approaches integrate the constraints into the actual mining process to generate only patterns that satisfy the constraints. CAP Ng et al. [17] and MFS_DoubleCons Duong et al. [4] algorithms use this strategy for mining frequent patterns with item constraints.

3.4 Problem statement

Given a sequence database Φ, the minimum support (*minsup*), and a set of items $\chi = \{u_1, u_2, \ldots, u_k\}$, the problem of mining inter-sequence patterns with item constraints is to find all sequences $\alpha = s_1[w_1], s_2[w_2],\ldots, s_m[w_m]$ such that $\exists s_i[w_i] \in \alpha, \exists b_j \in s_i[w_i]: b_j \in \chi$.

For example, let $\chi = \{C\}$. Then, the sequence $\langle C(AB)\rangle[0]\ \langle C(ABC)A\rangle[1]$ satisfies the constraints, whereas the sequence $\langle AD\rangle[0]\langle A\rangle[1]$ does not.

4 Mining inter-sequence patterns with item constraints

4.1 ISP-IC algorithm

Theorem 1 *If a sequence α satisfies constraint χ, then the sequence γ, generated from α, also satisfies constraint χ.*

Proof If α satisfies the constraint, it means that $\exists s_i[w_i] \in \alpha$, $\exists b_j \in s_i[w_i]: b_j \in \chi$. There are three cases to consider:

1. **Itemset extension**: There are two sub-cases. (i) If itemset extension $\notin$ itemset$_i$ of α, then $\exists$itemset$_i$ in γ that contains item $b_j \in \chi$. (ii) If itemset extension $\in$ itemset$_i$, then itemset$_i(\alpha) \subset$ itemset$_i(\gamma)$, and thus $\exists b_j \in$ itemset$_i(\gamma)$ such that $b_j \in \chi$.
2. **Itemset extension**: itemset$_i$ is not changed. Therefore, $\exists$itemset$_i \in \gamma$ includes item b_j such that $b_j \in \chi$.
3. **Inter extension**: Based on the inter extension definition, itemsets in the sequence just have their indices changed, and therefore $\exists$itemset$_i$ contains item $b_j \in \chi$.

Based on Theorem 1, we propose the ISP-IC algorithm for mining inter-sequence patterns with item constraints. First, the ISP-IC algorithm finds all items (I_1) that satisfy the threshold. Then, the algorithm inserts $p \in I_1$ into the DBV-tree. $\forall p \in I_1$ such that $p \in \chi$, $cs(p) =$ true. Next, the algorithm calls **Join_1-PatternList** to combine 1-PatternLists. Finally, for each $p \in$ *DBV-Tree.root*, the algorithm calls the **Join_*k*-PatternList** function to combine p with other ***k*-PatternLists** that follow it. The details of the ISP-IC algorithm are presented in Alg.1. □

Algorithm 1 ISP-IC algorithm

Input: Φ, *minsup*, *maxspan*, and $\chi = \{u_1, u_2, \ldots, u_k\}$
Output: *DBV-results*
DBV-Tree = *NULL*;
I_1 = {DBV-PatternList(i) |$i \in I$ and $sup(i) \geq minsup$} ;
For each p in I_1 **do**
  Add p to *childnodes*(*DBV-Tree*);
  If IC-Check (p, χ) = true **then**
    Add p to *DBV-results*
    Mark $cs(p)$ as true
Join_1-PatternList(*DBV-Tree*, *minsup*, *maxspan, DBV-results,* χ);
For each p in *childnodes*(*DBV-Tree*) **do**
  Join_k-PatternList (p, *minsup*, *maxspan*,*DBV-results,* χ);
Return *DBV-results*;

Function Join_1-PatternList (*DBV-Tree*, *minsup*, *maxspan*,*DBV-results,* χ)
Let *lists* = *childnodes*(*DBV-Tree*);
For each X in *lists* **do**
  For each Y in *lists* **do**
    If ($sup(Z = X \cup_I Y) \geq minsup$ **then**
      Add Z to *childnodes*(X);
      If $cs(X)$ = true or $cs(Y)$ = true or **IC-Check**(Z, χ) = true **then**
        Add Z to *DBV-results*
        Mark $cs(Z)$ as true
    If ($sup(Z = X \cup_S Y) \geq minsup$ **then**
      Add Z to *childnodes*(X);
      If $cs(X)$ = true or $cs(Y)$ = true or **IC-Check**(Z, χ) = true **then**
        Add Z to *DBV-results*
        Mark $cs(Z)$ is true
    For d =1 to *maxspan* **do**
      If ($sup(Z = X \cup_T Y) \geq minsup$ **then**
        Add Z to *childnodes*(X);
        If $cs(X)$ = true or $cs(Y)$ = true or **IC-Check**(Z, χ) = true **then**
          Add Z to *DBV-results*
          Mark $cs(Z)$ is true
Function Join_*k*-PatternList (p, *minsup*, *maxspan*,*DBV-results,* χ)
Let *lists* = *childnodes*(p);
For each X in *lists* **do**
  For each Y in *lists* **do**
    If ($sup(Z = X \cup_I Y) \geq minsup$ **then**
      Add Z to *childnodes*(X);
      If $cs(X)$ = true or $cs(Y)$ = true or **IC-Check**(Z, χ) = true **then**
        Add Z to *DBV-results*

Mark $cs(Z)$ as true
If $(sup(Z = X \cup_S Y) \geq minsup$ **then**
Add Z to $childnodes(X)$;
If $cs(X) =$ true or $cs(Y) =$ true or **IC-Check**$(Z, \chi) =$ true **then**
Add Z to *DBV-results*
Mark $cs(Z)$ is true
If $(sup(Z = X \cup_T Y) \geq minsup$ **then**
Add Z to $childnodes(X)$;
If $cs(X) =$ true or $cs(Y) =$ true or **IC-Check**$(Z, \chi) =$ true **then**
Add Z to *DBV-results*
Mark $cs(Z)$ is true
Join_*k*-PatternList(X, *minsup*, *maxspan*,*DBV-results*, χ);

Function IC-Check(*sequence*, χ)
For each I in *sequence* **do**
For each γ in I **do**
If $\gamma \in \chi$ **then**
Return true
Return false

4.2 An illustrative example

Consider Φ_{Ex}, which includes 7 sequences with items $I = \{A, B, C, D\}$, $maxpan = 1$, $minsup = 25\%$, and $\chi = \{C\}$. DBV-PatternList of each item was shown in Table 3.

Table 3a shows the bit arrays for the items in Φ_{Ex}. In this example, assume that each block is encoded by 4 bits. In Φ_{Ex}, bit array of A is {1111, 1010} and bit vector of A is {15, 10} . For each sequence, the position of each item in the sequence is stored as in Table 3b.

Once the sequences have been encoded into binary form and the positions in the sequence have become sequence blocks (with each sequence block containing 4 bits), the sequence database is transformed into a vertical format database and DBV-PatternList structures are created, as shown in Fig. 2. Note that index column the transaction position and come from Table 3b.

Then, the branch of $\langle A \rangle[0]$ with two levels is obtained. The nodes on the second-level branch are then computed, followed by those on the third-level branch, and so on, until no extended sequences are found. Figure 3 shows a part of a DBV-tree extended on branch $\langle A \rangle[0]$. Note that the node $\langle C \rangle[0]$ (red node) is the constraint and the nodes $\langle AC \rangle[0]$ and $\langle A \rangle[0]\langle C \rangle[1]$ (gray node) are inserted into the results without being checked whether they satisfy the constraint because their parent $C[0]$ satisfies the constraint.

4.3 iISP-IC algorithm

Lemma 1 *Let α satisfy constraint χ then $\forall\beta$, following sequences $\gamma_I = \alpha \cup_I \beta$, $\gamma_S = \alpha \cup_S \beta$, and $\gamma_T = \alpha \cup_T \beta$ also satisfy constraint χ.*

Based on Lemma 1, we propose the *i*ISP-IC algorithm. For each $p \in childnodes(DBV\text{-}Tree)$, if p satisfies χ, *i*ISP-IC calls the **Join_k-PatternList_NoChecking** function to combine p with other ***k*-PatternLists** that follow it. Otherwise, *i*ISP-IC calls the **Join_*k*-**PatternList_Plus function. The details of the *i*ISP-IC algorithm are presented in Alg.2.

Algorithm 2 *i*ISP-IC algorithm

Input: Φ, *minsup*, *maxspan*, and $\chi = \{u_1, u_2, \ldots, u_k\}$
Output: *DBV-results*
Let *DBV-Tree* = *NULL*;
$I_1 = \{$DBV-PatternList(i) $|i \in I$ and $sup(i) \geq minsup\}$;
For each p in I_1 **do**
Add p to $childnodes(DBV\text{-}Tree)$;
If IC-Check $(p, \chi) =$ true **then**
Add p to *DBV-results*
Mark $cs(p)$ as true
Join_1-PatternList(*DBV-Tree*, *minsup*, *maxspan*, *DBV-results*, χ);
For each p in $childnodes(DBV\text{-}Tree)$ **do**
If $cs(p) =$ true **then**
Join_k-PatternList_Plus(p, *minsup*, *maxspan*,*DBV-results*, χ);
Else
Join_k-PatternList_NoChecking(p, *minsup*, *maxspan*, *DBV-results*);
Return *DBV- results*;

Function Join_*k*-PatternList_Plus(p, *minsup*, *maxspan*, *DBV-results*, χ)
Let $lists = childnodes(p)$;
For each X in *lists* **do**
For each Y in *lists* **do**
If $(sup(Z = X \cup_I Y) \geq minsup$ **then**
Add Z to $childnodes(X)$;
If $cs(Y) =$ true or **IC-Check**$(Z, \chi) =$ true **then**
Add Z to *DBV-results*
Mark $cs(Z)$ as true
If $(sup(Z = X \cup_S Y) \geq minsup$ **then**
Add Z to $childnodes(X)$;
If $cs(Y) =$ true or **IC-Check**$(Z, \chi) =$ true **then**
Add Z to *DBV-results*
Mark $cs(Z)$ is true
If $(sup(Z = X \cup_t Y) \geq minsup$ **then**
Add Z to $childnodes(X)$;

If $cs(Y) =$ true or **IC-Check**$(Z, \chi) =$ true **then**
Add Z to *DBV-results*
Mark $cs(Z)$ is true
If $cs(X) =$ true **then**
Join_*k*-PatternList_NoChecking(X, *minsup*, *maxspan*,*DBV-results*);
Else
Join_*k*-PatternList_Plus(X, *minsup*, *maxspan*,*DBV-results*, χ);

Function Join_*k*-PatternList_NoChecking(p, *minsup*, *maxspan*, *DBV-results*)
Let *lists* = *childnodes*(p);
For each X in *lists* **do**
For each Y in *lists* **do**
If $(sup(Z = X \cup_I Y) \geq minsup$ **then**
Add Z to *DBV-results* and *childnodes*(X);
If $(sup(Z = X \cup_S Y) \geq minsup$ **then**
Add Z to *DBV-results* and *childnodes*(X);
If $(sup(Z = X \cup_T Y) \geq minsup$ **then**
Add Z *DBV-results* and *childnodes*(X);
Join_*k*-PatternList_NoChecking(X, *minsup*, *maxspan*,*DBV-results*);

In the example, suppose that $\chi = \{A\}$, the red node in Fig. 4. The *i*ISP-IC algorithm calls the **Join_*k*-PatternList_NoChecking** function to expand the node $\langle A\rangle[0]$, as shown in Fig. 4. Based on Lemma 1, this function does not check all its child nodes.

4.4 The complexity discussion

The complexity of sequential pattern mining algorithms is based on the number of patterns in the search space, and the cost of the operations for generating and processing each itemset Fournier-Viger et al. [5].

Let $t = \{u_1, u_2, u_3, \ldots, u_m\}$ is the set of items, and let $|t|$denote its cardinality. We have to consider all possible *permutations* of the items as the possible frequent candidates. The subsequence search space is conceptually infinite since it comprises all sequences in t. The dataset D consists of bounded length sequences in practice. Let j be the length of the longest sequence in the dataset D, we will have to consider all candidate sequences of length up to j in the worst case, which gives the following bound on the size of the search space $|t|^1 + |t|^2 + |t|^3 + \ldots + |t|^j$ $(O(|t|^n))$ since at level k there are $|t|^k$ possible subsequences of length k.

However, the complexity of the proposed algorithms depends on the number of sequences, the number of itemsets in each sequence, and the constraints. Therefore, it is hard to compute the time complexity in the average case.

4.5 piISP-IC: a parallel version of iISP-IC algorithm

In the parallel mining of ISPs, each branch of the search tree can be regarded as a single task, which can be processed independently to generate ISPs. An example is given in Fig. 5 There are three tasks on level 1 of the task tree. Tasks 1 2, and 3 process branches $AB$and C respectively.

The interaction between cores in multi-core CPU's can be implemented by various mechanisms, affecting overall CPU performance due to shared workloads between cores. The most efficient way to allow on-chip interprocess communication (IPC) is through the use of shared memory. This is a very important feature for efficient hardware utilization of multi-core systems. Shared memory IPC is more efficient because data does not need to be copied from one process memory space to another. Instead, a memory space that is shared between the communicating processes is created. By this way, IPC that uses shared memory avoids many costly memory operations.

The task parallel formulation distributes the tasks among the processors in the following way. First, the tree is expanded using the data-parallel algorithm at level $k+1$, with $k > 0$. Then, the different nodes at level k are distributed among the processors. Once this initial distribution is done, each processor proceeds to generate the subtrees underneath the nodes to which they have been assigned.

*pi*ISP-IC algorithm is generally based on *i*ISP-IC algorithm and add a parallel implementation of tasks instead

Table 3 Binary vectors of BitTable and their transformation into BitArray

(a)			(b)			
Item	**Bit array**	**Bit vector**	**Dat**	***A***	***B***	***C***
A	1111,1010	15,10	1	2	2	1
B	1101,0110	13,6	2	2, 3	2	1, 2
C	1100,1110	12,14	3	1		
			4	1	2	
			5	1		2
			6		1	2
			7	1	1	2

Fig. 2 DBV-PatternList of $A[0]$, $B[0]$ and $C[0]$

Sequence	<A>[0]					
Start	1					
DBV	15				10	
Index	1(2)	2(2,3)	3(1)	4(1)	1(1)	3(1)

Sequence	<B>[0]				
Start	1				
DBV	13			6	
Index	1(2)	2(2)	4(2)	2(1)	3(1)

Sequence	<C>[0]				
Start	1				
DBV	12		14		
Index	1(1)	2(1,2)	1(2)	2(2)	3(2)

of threads. The advantage of *pi*ISP-IC is that each task is assigned for searching a branch of the tree and is processed independently. The advantages over using threads are as follows. First, tasks require less memory than do threads. Second, a thread runs on only one core, whereas a task can run on multiple cores. Finally, threads require more processing time than executing tasks because the operating system needs to allocate data structures for threads, such as initialization and destruction and must perform context switching between threads.

The time complexity of sequential pattern mining algorithms depends on the number of patterns in the search space, and the cost of the operations for generating and processing each itemset. In multi-core processors, a key factor is task scheduling. With the widespread use of multi-core processors, designing an effective task scheduling strategy has been a hot issue. Currently, the time complexity of task scheduling for multi-core processors is considered as an NP (Non-Deterministic Polynomial) problem and no optimal solutions exist. The existing scheduling algorithms can only get the suboptimal solutions based on solution approximated by heuristics. Heuristics are generally considered as the most suitable method to find the suboptimal solutions for NP problems.

5 Experimental studies

This section compares the mining time of POST-EISP-Miner, ISP-IC, *i*ISP-IC, and *pi*ISP-IC, to confirm the effectiveness of the proposed methods. All the experiments were performed on a PC with Intel Xeon Processor E5-2680 v2 (25M Cache, 2.80 GHz, 20 threads) CPUs installed with 768 GB of main memory and coded in C# in Visual Studio 2015. Synthetic databases were generated using the IBM synthetic data generator to mimic transactions in a retail environment. The synthetic data generation program used the following parameters: C was the average number of itemsets per sequence, T was the average number of items per itemset, S was the average number of itemsets in maximal sequences, I was the average number of items in maximal sequences, N was the number of distinct items, and D was the number of sequences. Two synthetic datasets (C6T5S4I4N1KD10K and T10I4D100K) and BMSWebView2 which contains 77,512 sequences of click-stream data, were used in the experiments. These databases are available at http://1drv.ms/1EM0bqm.

5.1 Runtime

In Fig. 6, we compare the runtime of *pi*ISP-IC, *i*ISP-IC, ISP-IC and POST-EISP-Miner for C6T5S4I4N1KD10K dataset with various settings. Figure 6A fixes *maxspan* = 1 and changes the *threshold* from 21 to 25 and we easily found that the runtime of *i*ISP-IC is less than that of ISP-IC and nearly twice as that of *pi*ISP-IC, while POST-EISP-Miner's decreased a little. In a similar way, Fig. 6B fixes *maxspan* = 2, Fig. 6C fixes *maxspan* = 3, Fig. 6D fixes *maxspan* = 4 and Fig. 6E fixes *maxspan* = 5, *pi*ISP-IC is always the best algorithm for C6T5S4I4N1KD10K dataset. POST-EISP-Miner cannot perform with *maxspan* = 5 with any threshold, takes a long time without results. Next, in Fig. 6F, we fix *threshold* = 21, and change *maxspan* from 1 to 5. *pi*ISP-IC is still better than *i*ISP-IC, ISP-IC, and POST-EISP-Miner algorithms. Especially, when we decrease the *threshold*, the runtime of *i*ISP-IC, ISP-IC and POST-EISP-Miner significantly increase while that of *pi*ISP-IC gradually increase. In general, *pi*ISP-IC outperform *i*ISP-IC, ISP-IC, and POST-EISP-Miner for this dataset.

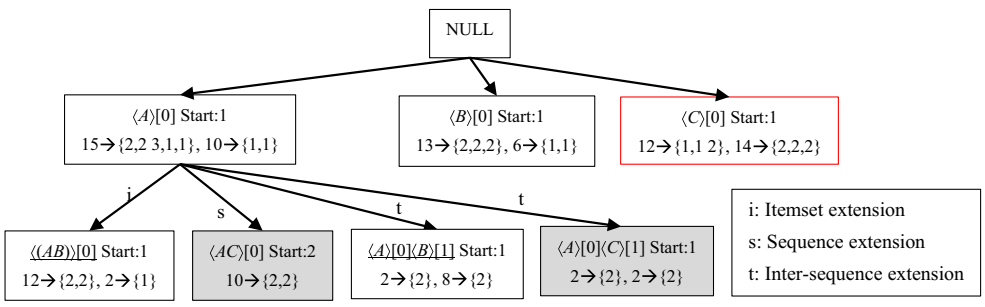

Fig. 3 DBV-tree extended on $\langle A \rangle[0]$

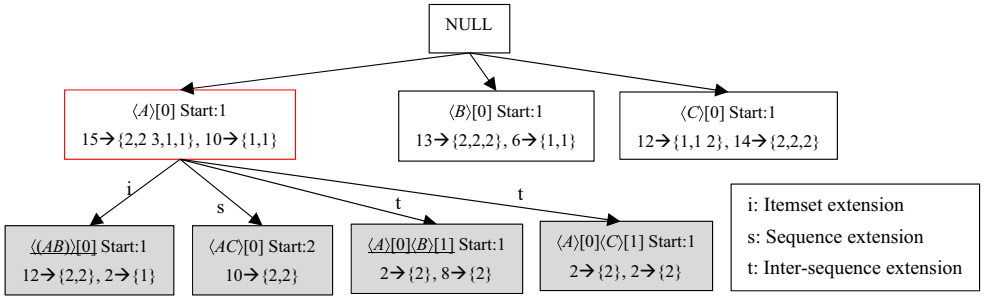

Fig. 4 DBV-tree extended on ⟨*A*⟩[0] using *i*ISP-IC algorithm

Figure 7 performs the same for T10I4D100K dataset. We easy found that the runtimes of *i*ISP-IC and ISP-IC are nearly the same, POST-EISP-Miner is highest while the runtime of *pi*ISP-IC is much better than that of *i*ISP-IC, ISP-IC, and POST-EISP-Miner. POST-EISP-Miner cannot perform with $minsup = 4$ and $maxspan = 5$.

Figure 8 compares the runtime of *pi*ISP-IC, *i*ISP-IC, ISP-IC and POST-EISP-Miner for BMSWebView2 dataset with various settings. Figure 8A fixes $maxspan = 1$, Fig. 8B fixes $maxspan = 2$, Fig. 8C fixes $maxspan = 3$, Fig. 8D fixes $maxspan = 4$ and Fig. 8E fixes $maxspan = 5$, *pi*ISP-IC is always the best algorithm for mining inter-sequence patterns with item constraints for BMSWebView2 dataset. Especially, with the smaller threshold, the time gaps between the runtime of *pi*ISP-IC and those of *i*ISP-IC, ISP-IC and POST-EISP-Miner are larger. In Fig. 8F, we fix *threshold* = 1, and change *maxspan* from 1 to 5, the runtime of *pi*ISP-IC is less than those of *i*ISP-IC, ISP-IC, and POST-EISP-Miner. Therefore, *pi*ISP-IC outperform *i*ISP-IC, ISP-IC, and POST-EISP-Miner for this dataset. In short, through the above experiments, we can conclude that *pi*ISP-IC outperform *i*ISP-IC, ISP-IC, and POST-EISP-Miner for mining inter-sequence patterns with item constraints.

5.2 Memory usage

Figure 9 reports the memory usage of *pi*ISP-IC, *i*ISP-IC, ISP-IC and POST-EISP-Miner for C6T5S4I4N1KD10K

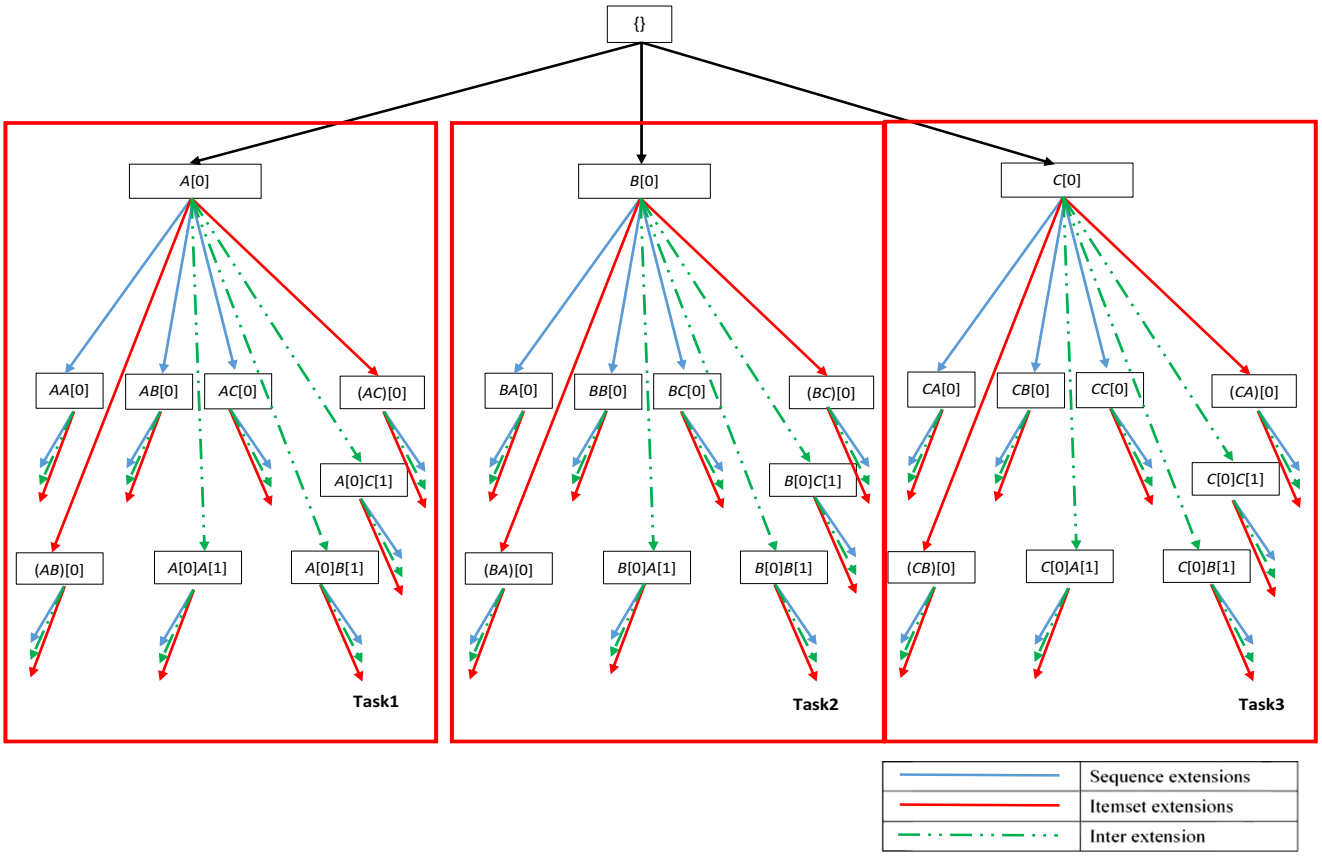

Fig. 5 Example of task tree

dataset with various settings. Figure 9A fixes *maxspan* = 1, Fig. 9B fixes *maxspan* = 2, Fig. 9C fixes *maxspan* = 3, Fig. 9D fixes *maxspan* = 4 and Fig. 9E fixes *maxspan* = 5, we found that the memory usage of POST-EISP-Miner is the largest. However, the gap between the memory usage of *pi*ISP-IC and those of *i*ISP-IC and ISP-IC are insignificant for C6T5S4I4N1KD10K dataset. Then, Fig. 8F, which shows the memory usage of *pi*ISP-IC, *i*ISP-IC, ISP-IC and POST-EISP-Miner when we fix *threshold* = 21 and change *maxspan* from 1 to 5, confirm that statement again. POST-EISP-Miner cannot perform with *maxspan* =5 and *threshold* = 21, the memory value with *maxspan* = 5 in Fig. 9F is an example.

Figures 10-11 reports the memory usage of *pi*ISP-IC, *i*ISP-IC, ISP-IC and POST-EISP-Miner for T10I4D100K and BMSWebView2. The results show that the memory usages of these algorithms are nearly the same. Summary, although the memory usage of *pi*ISP-IC is largest (compared to those of *i*ISP-IC, ISP-IC, and POST-EISP-Miner), the gap between the memory usage of *pi*ISP-IC and those of *i*ISP-IC, ISP-IC and POST-EISP-Miner are insignificant.

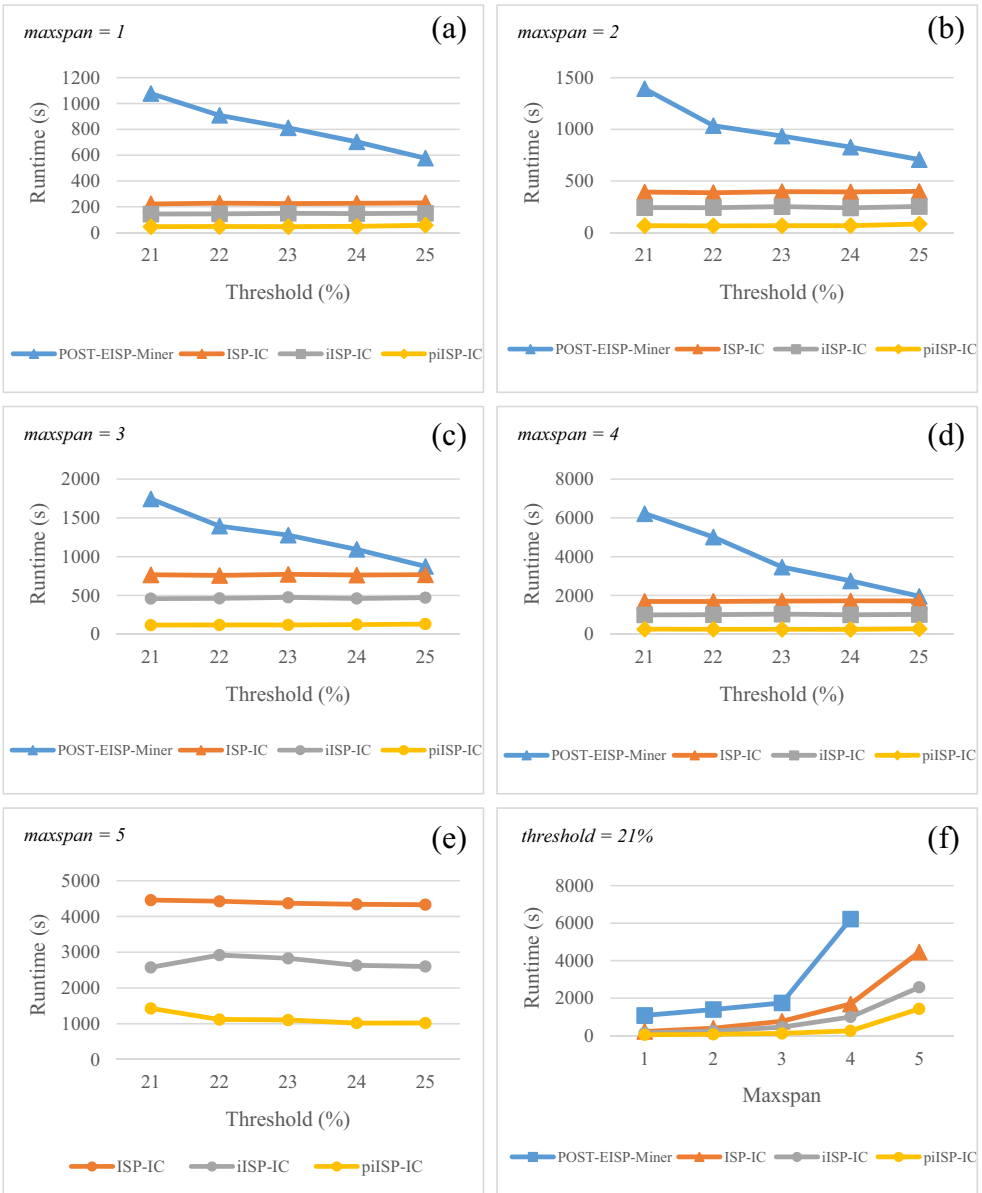

Fig. 6 Mining time of piISP-IC, iISP-IC, ISP-IC and POST-EISP-Miner for C6T5S4I4N1KD10K dataset with 15 random item constraints and **A** maxspan = 1; **B** maxspan = 2; **C** maxspan = 3; **D** maxspan = 4; **E** maxspan = 5; **F** threshold = 21%

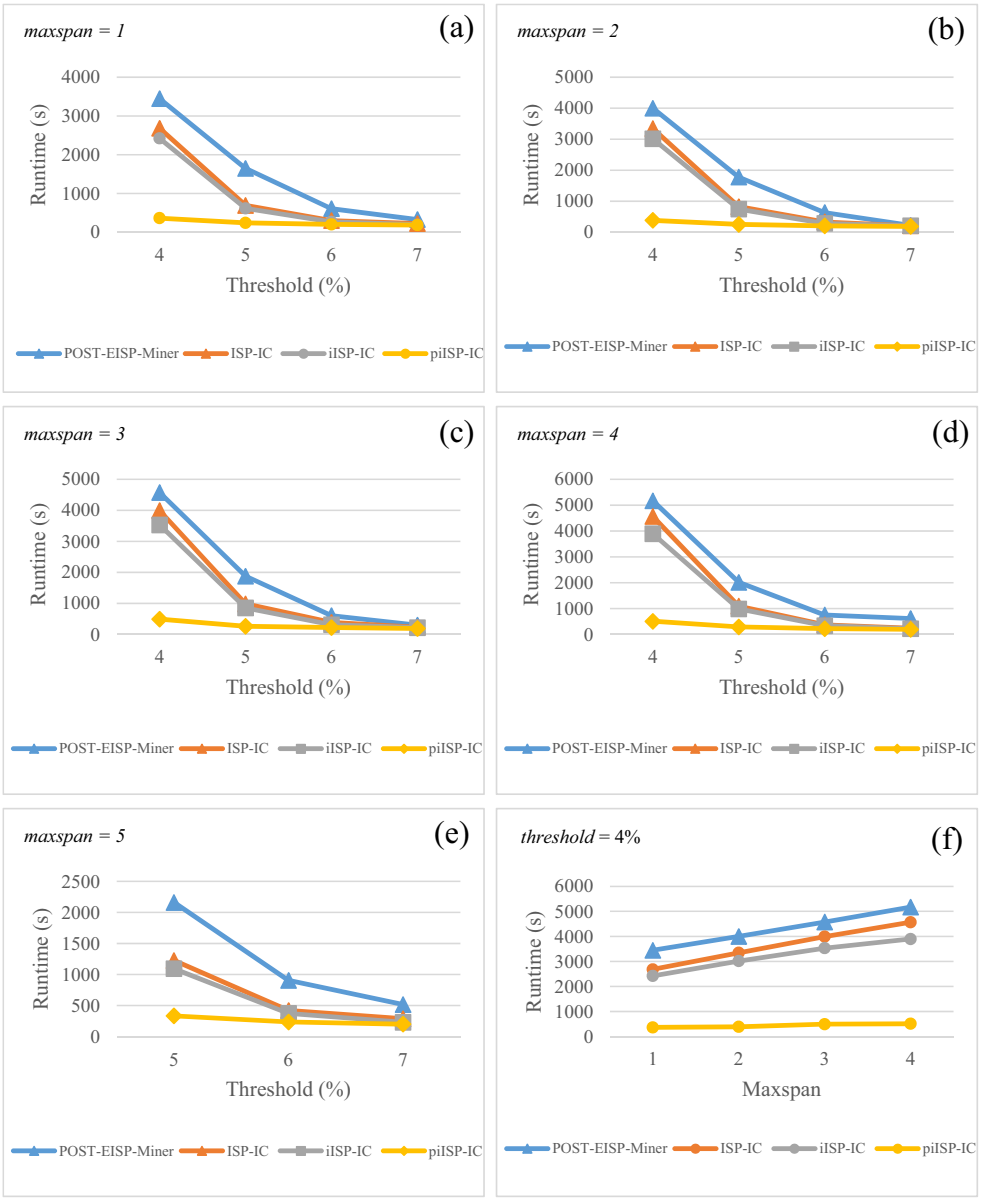

Fig. 7 Mining time of piISP-IC, iISP-IC, ISP-IC and POST-EISP-Miner for T10I4D100K dataset with 15 random item constraints and **A** maxspan = 1; **B** maxspan = 2; **C** maxspan = 3; **D** maxspan = 4; **E** maxspan = 5; **F** threshold = 4%

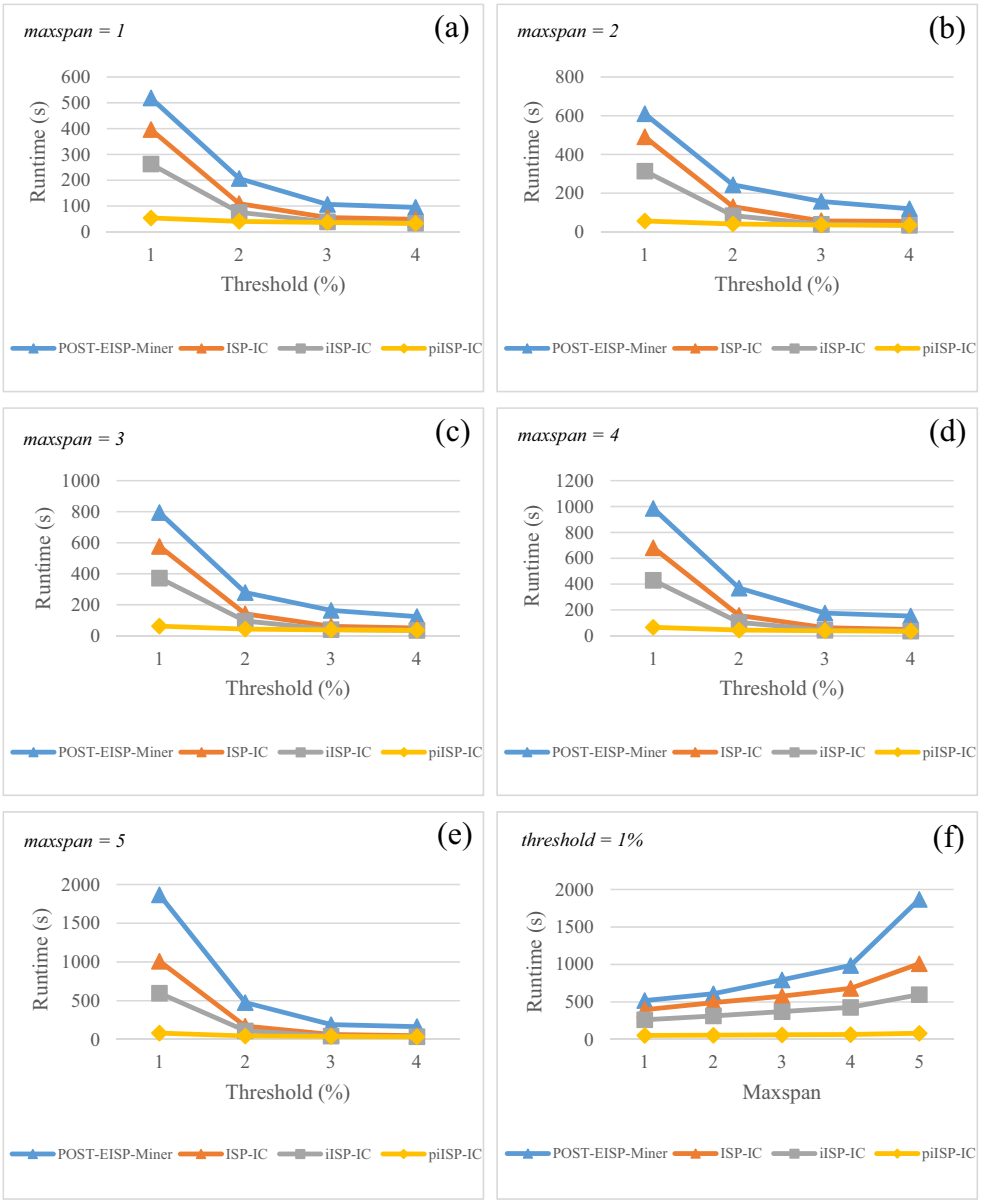

Fig. 8 Mining time of piISP-IC, iISP-IC, ISP-IC and POST-EISP-Miner for BMSWebView2 dataset with 15 random item constraints and **A** maxspan = 1; **B** maxspan = 2; **C** maxspan = 3; **D** maxspan = 4; **E** maxspan = 5; **D** threshold = 1%

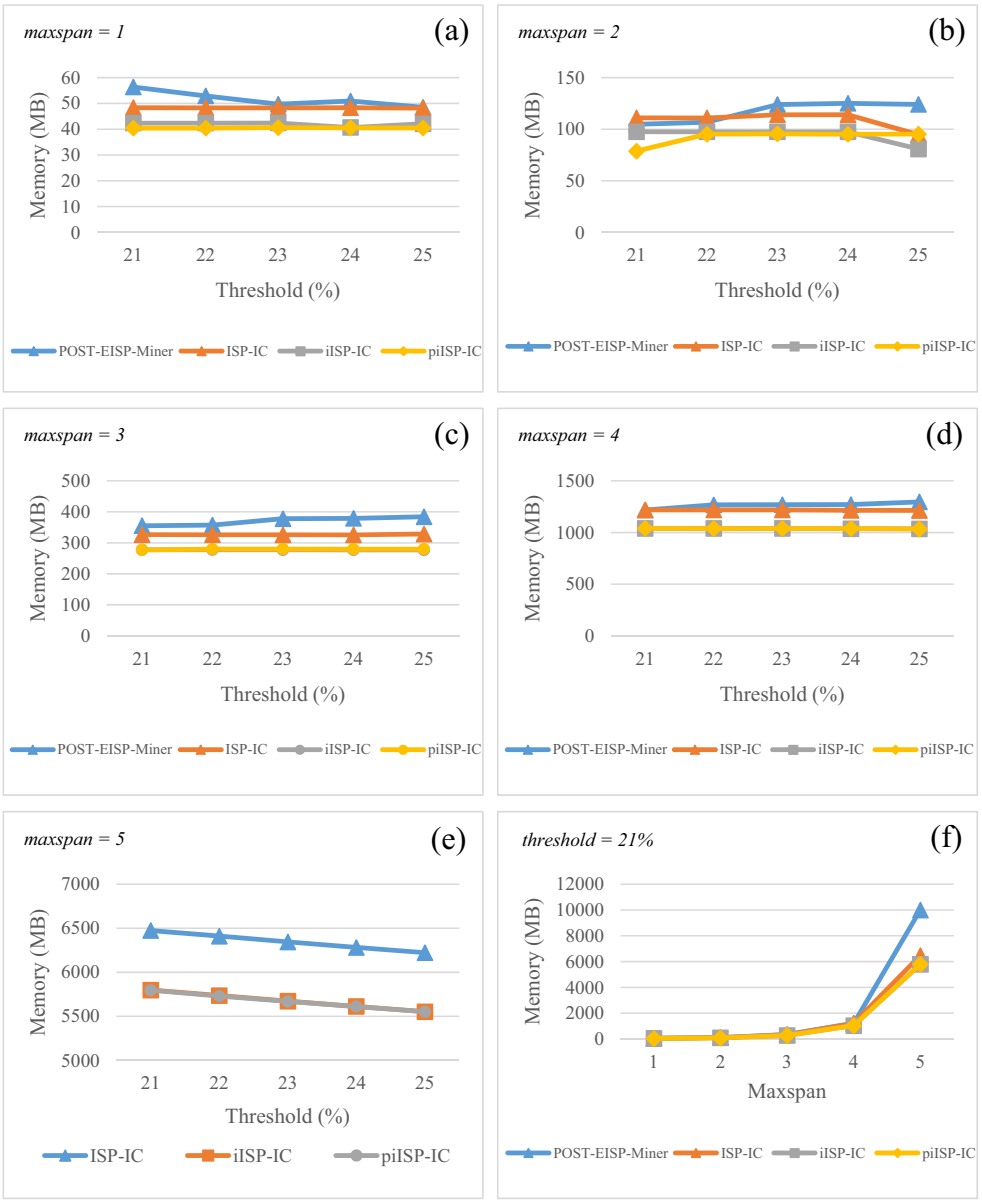

Fig. 9 Memory usage of piISP-IC, iISP-IC, ISP-IC and POST-EISP-Miner for C6T5S4I4N1KD10K dataset with 15 random item constraints and **A** maxspan = 1; **B** maxspan = 2; **C** maxspan = 3; **D** maxspan = 4; **E** maxspan = 5; **F** threshold = 21%

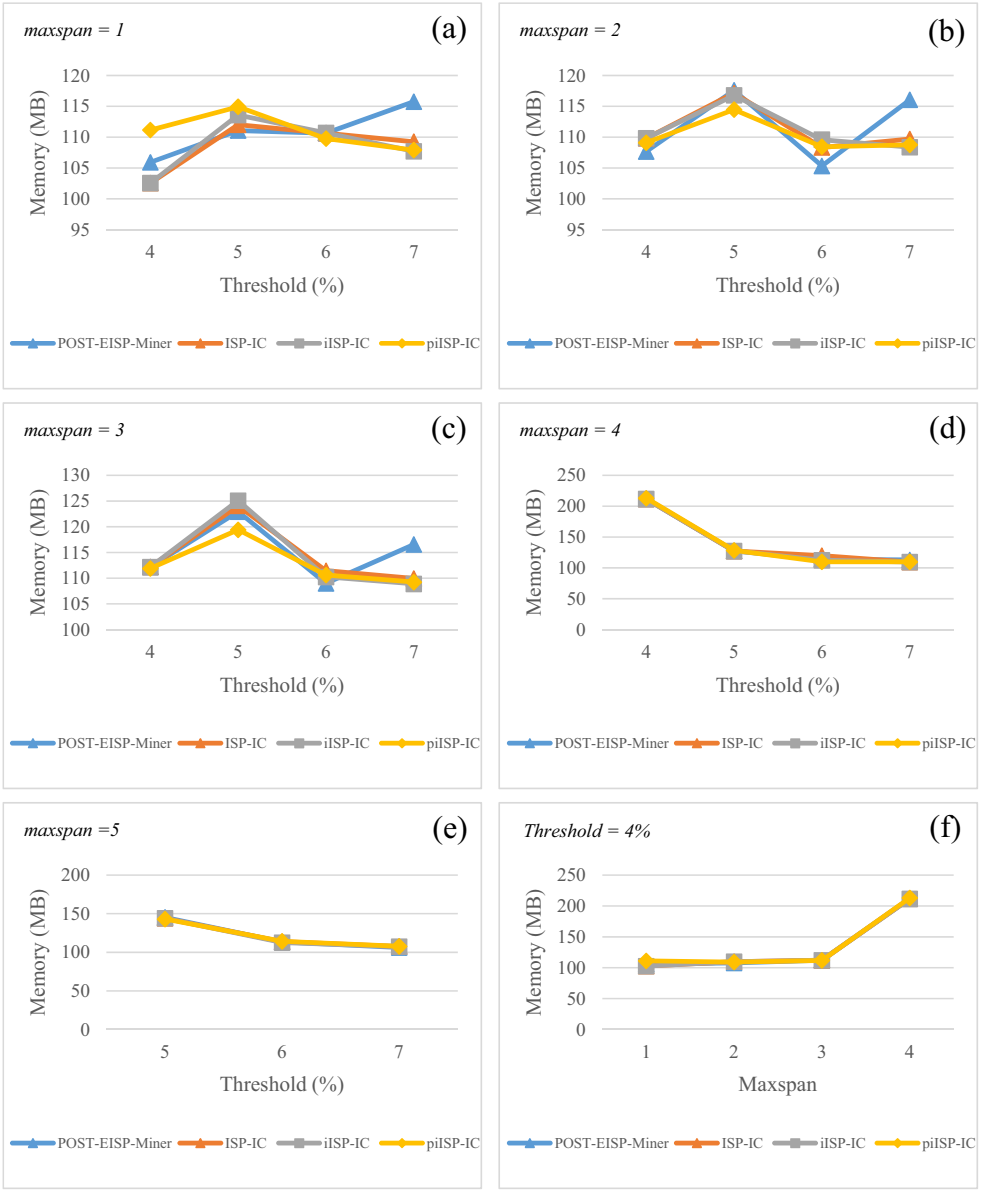

Fig. 10 Memory usage of piISP-IC, iISP-IC, ISP-IC and POST-EISP-Miner for T10I4D100K dataset with 15 random item constraints and **A** maxspan = 1; **B** maxspan = 2; **C** maxspan = 3; **D** maxspan = 4; **E** maxspan = 5; **F** threshold = 4%

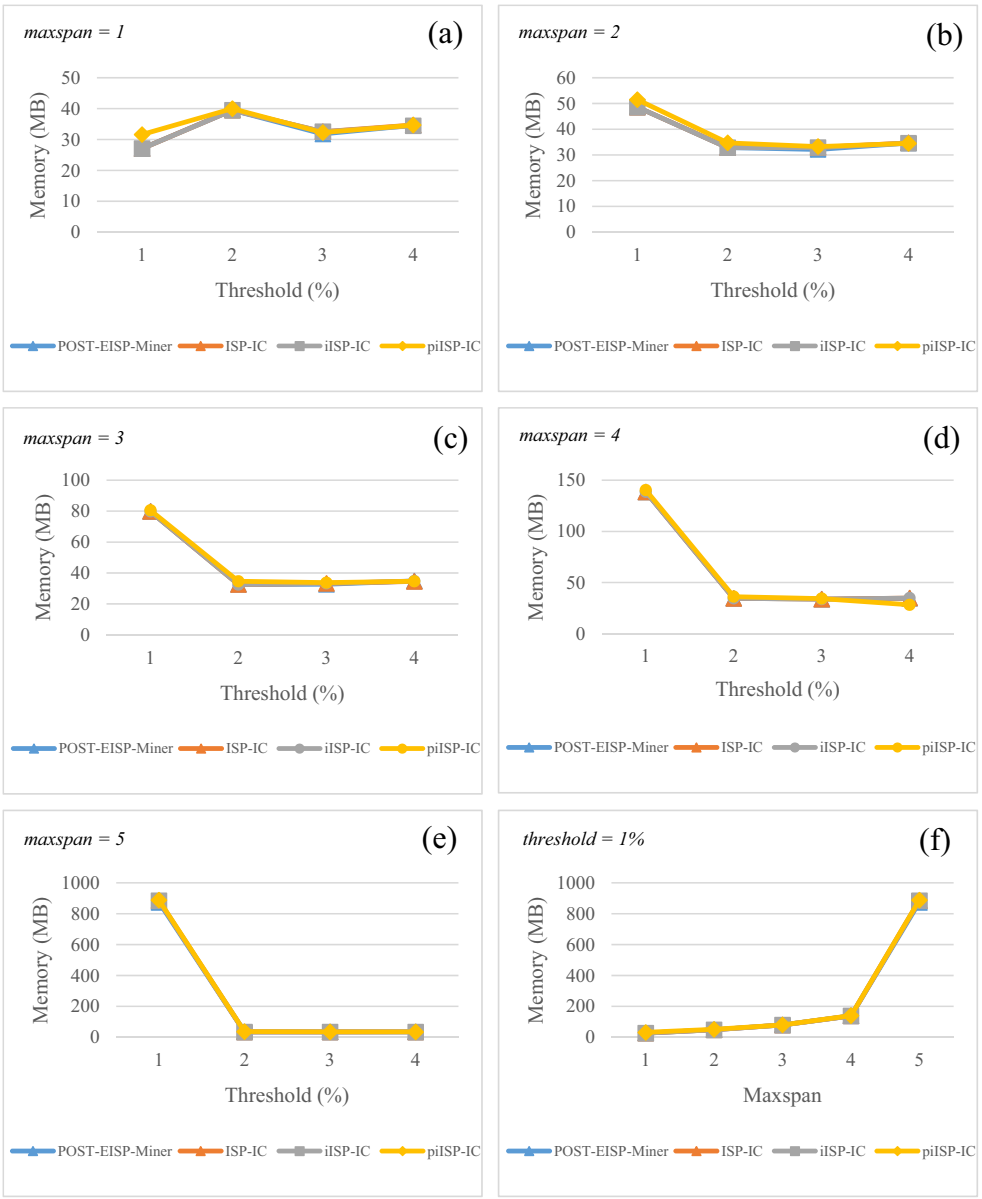

Fig. 11 Memory usage of piISP-IC, iISP-IC, ISP-IC and POST-EISP-Miner for BMSWebView2 dataset with 15 random item constraints and **A** maxspan = 1; **B** maxspan = 2; **C** maxspan = 3; **D** maxspan = 4; **E** maxspan = 5; **F** threshold = 1%

6 Conclusions and future studies

This paper proposed three novel methods for mining inter-sequence patterns with item constraints. Firstly, the problem of mining inter-sequence patterns with item constraints was presented. Secondly, a theorem was developed to reduce mining time by fast determining whether an inter-sequence pattern satisfies item constraints. Based on this theorem, an efficient algorithm for mining inter-sequence patterns with constraints (ISP-IC algorithm) was proposed. Thirdly, a lemma was presented for improving the strategy of ISP-IC. Based on this lemma, the *i*ISP-IC algorithm was proposed. Fourthly, we presented a parallel version of *i*ISP-IC named *pi*ISP-IC to improve the performance. Finally, experiments were conducted to verify the proposed approaches. Experimental results show that the *pi*ISP-IC algorithm is better than the POST-EISP-Miner, ISP-IC and *i*ISP-IC algorithms.

This paper focused on item constraints. In the future, we will study mining inter-sequence patterns with both itemset and sequence constraints as well as the combination of many constraints and closed inter-sequence patterns with constraints (including item, itemset, and sequence constraints).

Acknowledgments This research is funded by Foundation for Science and Technology Development of Ton Duc Thang University (FOSTECT), website: http://fostect.tdt.edu.vn, under Grant FOSTECT.2015.BR.01.

References

1. Ayres J, Flannick J, Gehrke J, Yiu T (2002) Sequential pattern mining using a bitmap representation. In: Proceedings of the KDD'02, pp 429–435
2. Bucila C, Gehrke JE, Kifer D, White W (2003) Dualminer: A dual-pruning algorithm for itemsets with constraints. Data Min Knowl Discov 7(3):241–272
3. Cao L, Zhang H, Zhao Y, Luo D, Zhang C (2011) Combined mining: discovering informative knowledge in complex data. IEEE Trans Syst, Man, Cybern Part B 41(3):699–712
4. Duong H, Truong T, Vo B (2014) An efficient method for mining frequent itemsets with double constraints. Eng Appl Artif Intell 27:148–154
5. Fournier-Viger P, Lin JC-W, Kiran RU, Koh YS, Thomas R (2017) A survey of sequential pattern mining. Data Sci Pattern Recogn (DSPR) 1(1):54–77
6. Gouda K, Hassaan M, Zaki MJ (2010) Prism: A primal-encoding approach for frequent sequence mining. J Comput Syst Sci 76(1):88–102
7. Kaneiwa K, Kudo Y (2011) A sequential pattern mining algorithm using rough set theory. Int J Approx Reason 52(6):881–893
8. Jeyabharathi J, Shanthi D (2016) Enhanced sequence identification technique for protein sequence database mining with hybrid frequent pattern mining algorithm. Int J Data Min Bioinforma 16(3):205–229
9. Jung H, Chung K (2015) Sequential pattern profiling based bio-detection for smart health service. Clust Comput 18(1):209–219
10. Le B, Tran MT, Vo B (2015) Mining frequent closed inter-sequence patterns efficiently using dynamic bit vectors. Appl Intell 43(1):74–84
11. Lee AJT, Wang CS, Weng WY, Chen YA, Wu HW (2008) An efficient algorithm for mining closed inter-transaction itemsets. Data Knowl Eng 66(1):68–91
12. Lee AJT, Wang CS (2007) An efficient algorithm for mining frequent inter-transaction patterns. Inf Sci 177(17):3453–3476
13. Liao VCC, Chen MS (2014) DFSP: a Depth-First SPelling algorithm for sequential pattern mining of biological sequences. Knowl Inf Syst 38(3):623–639
14. Lin CJ, Wu C, Chaovalitwongse WA (2015) Integrating human behavior modeling and data mining techniques to predict human errors in numerical typing. IEEE Trans Human-Mach Syst 45(1):39–50
15. Lin WY, Huang KW, Wu CA (2010) MCFPTree: An FP-tree-based algorithm for multi constraint patterns discovery. Int J Bus Intell Data Min 5(3):231–246
16. Lu H, Feng L, Han J (2000) Beyond intra-transaction association analysis: mining multi-dimensional inter-transaction association rules. ACM Trans Inf Syst 18(4):423–454
17. Ng RT, Lakshmanan LVS, Han J, Pang A (1998) Exploratory mining and pruning optimizations of constrained association rules. In: Proceedings of the SIGMOD'98, pp 13–24
18. Pham TT, Luo J, Hong TP, Vo B (2015) An efficient method for mining non-redundant sequential rules using attributed prefix-trees. Eng Appl Artif Intell 32:88–99
19. Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu M-C (2004) Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Trans Knowl Data Eng 16(11):1424–1440
20. Saif-Ur-Rehman J, Habib A, Salam A (2016) Ashraf Top-K Miner: top-K identical frequent itemsets discovery without user support threshold. Knowl Inf Syst 48(3):741–762
21. Salehi M, Kamalabadi IN, Ghoushchi MBG (2014) Personalized recommendation of learning material using sequential pattern mining and attribute based collaborative filtering. Educ Inf Technol 19(4):713–735
22. Scalmato A, Sgorbissa A, Zaccaria R (2013) Describing and recognizing patterns of events in smart environments with description logic. IEEE Trans Cybern 43(6):1882–1897
23. Tran MT, Le B, Vo B (2015) Combination of dynamic bit vectors and transaction information for mining frequent closed sequences efficiently. Eng Appl Artif Intell 38:183–189
24. Tung A, Lu H, Han J, Feng L (2003) Efficient mining of Inter-transaction association rules. IEEE Trans Knowl Data Eng 15(1):43–56
25. Vo B, Tran MT, Nguyen H, Hong TP, Le B (2012a) A dynamic bit-vector approach for efficiently mining inter-sequence patterns. In: Proceedings of the IBICA'12, pp 51–56
26. Vo B, Hong TP, Le B (2012) DBV-Miner: A dynamic bit-vector approach for fast mining frequent closed itemsets. Expert Syst Appl 39(8):7196–7206
27. Vo B, Pham S, Le T, Deng ZH (2017) A novel approach for mining maximal frequent patterns. Expert Syst Appl 73:178–186
28. Wang CS, Lee AJT (2009) Mining inter-sequence patterns. Expert Syst Appl 36(4):8649–8658
29. Wang CS, Liu YH, Chu KC (2013) Closed inter-sequence pattern mining. J Syst Softw 86(6):1603–1612
30. Wright AP, Wright AT, McCoy AB, Sittig DF (2015) The use of sequential pattern mining to predict next prescribed medications. J Biomed Inf 53:73–80
31. Xue Y, Li T, Liu Z, Pang C, Li M, Liao Z, Hu X (2015) (In press). A new approach for the deep order preserving submatrix problem based on sequential pattern mining. International Journal of Machine Learning and Cybernetics. https://doi.org/10.1007/s13042-015-0384-z
32. Yen SJ, Lee YS (2013) Mining non-redundant time-gap sequential patterns. Appl Intell 39(4):727–738
33. Yun U, Pyun G, Yoon E (2015) Efficient mining of robust closed weighted sequential patterns without information loss. Int J Artif Intell Tools 24(1):1550007. [28 pages]. https://doi.org/10.1142/S0218213015500074
34. Yun U, Ryu K, Yoon E (2011) Weighted approximate sequential pattern mining within tolerance factors. Intell Data Anal 15(4):551–569
35. Yun U, Ryu K (2010) Discovering important sequential patterns with length-decreasing weighted support constraints. Int J Inf Technol Decis Making 9(4):575–599
36. Zhang S, Du Z, Wang JTL (2015) New techniques for mining frequent patterns in unordered trees. IEEE Trans Cybern 45(6):1113–1125
37. Yun U, Kim D (2017) Mining of high average-utility itemsets using novel list structure and pruning strategy. Fut Gener Comput Syst 68:346–360
38. Ryang H, Yun U (2016) High utility pattern mining over data streams with sliding window technique. Expert Syst Appl 214-231:57
39. Kim D, Yun U (2016) Efficient mining of high utility pattern with considering of rarity and length. Appl Intell 45(1):152–173

40. Ryang H, Yun U, Ryu K (2016) Fast algorithm for high utility pattern mining with the sum of item quantities. Intell Data Anal 20(2):395–415
41. Kieu T, Vo B, Le T, Deng ZH, Le B (2017) Mining top-k co-occurrence items with sequential pattern. Expert Syst Appl 85:123–133
42. Zhang B, Lin JCW, Fournier-Viger P, Li T (2017) Mining of high utility-probability sequential patterns from uncertain databases. PLoS ONE 12(7):e0180931. https://doi.org/10.1371/journal. pone.0180931
43. Lin JCW, Gan W, Hong TP, Chen HY, Li ST (2016) An efficient algorithm to maintain the discovered frequent sequences with record deletion. Intell Data Anal 20(3):655–677
44. Lin JCW, Gan W, Fournier-Viger P, Hong TP (2016) Efficiently updating the discovered sequential patterns for sequence modification. Int J Softw Eng Knowl Eng 26(8):1285–1314
45. Zhang J, Wang Y, Yang D (2015) CCSpan: Mining closed contiguous sequential patterns. Knowl-Based Syst 89:1–13