Engineering Applications of Artificial Intelligence 38 (2015) 183–189

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

# Combination of dynamic bit vectors and transaction information for mining frequent closed sequences efficiently

Minh-Thai Tran [a], Bac Le [b], Bay Vo [c,*]

[a] Faculty of Information Technology, Information Technology College, Ho Chi Minh City, Vietnam
[b] Department of Computer Science, University of Science, VNU-Ho Chi Minh, Vietnam
[c] Faculty of Information Technology, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam

ARTICLE INFO

*Article history:*
Received 24 May 2014
Received in revised form
23 October 2014
Accepted 28 October 2014
Available online 28 November 2014

*Keywords:*
Dynamic bit vector
Frequent closed sequence
CloFS-DBV

ABSTRACT

Sequence mining algorithms attempt to mine all possible frequent sequences. These algorithms produce redundant results, increasing the required storage space and runtime, especially for large sequence databases. In recent years, many studies have proved that mining frequent closed sequences is more efficient than mining all frequent sequences. The desired information can be fully extracted from frequent closed sequences. Most algorithms for mining frequent closed sequences use a candidate maintenance-and-test paradigm. The present paper proposes an algorithm called CloFS-DBV that uses dynamic bit vectors. Various methods are employed to reduce memory usage and runtime. Experimental results show that CloFS-DBV is more efficient than the BIDE and CloSpan algorithms in terms of execution time and memory usage.



## 1. Introduction

Sequential pattern mining is a fundamental problem in knowledge discovery and data mining with broad applications, including those in the analysis of customer purchase behavior, web access patterns, scientific experiments, disease treatment, natural disaster prevention, and protein formation. Sequential pattern mining includes two main stages: frequent pattern mining and rule mining. Many studies have modified the AprioriAll algorithm (Agrawal and Srikant, 1995) for mining frequent sequential patterns. Unlike the general mining of frequent sequences, the mining of frequent closed sequences has not been extensively studied. Although some algorithms have been proposed, such as CloSpan (Yan et al., 2003), CLOSET+ (Wang et al., 2003), and BIDE (Wang et al., 2007), their performance is poor for large databases. BIDE detects frequent sequences, not closed ones, and prunes candidates early, instead of using maintenance-and-test patterns.

Recently, many authors have proposed techniques that present data in a vertical format (Song et al., 2005), use projection databases operation (Pei et al., 2001), use bit vector data structures (Song et al., 2008), all of which have been shown to be effective. However, the storage space and execution time can be further reduced in the mining process for large sequence databases.

The present study proposes the CloFS-DBV algorithm, which uses a vertical data format and data compression, and divides the search space to reduce the required storage space and execution time for mining frequent closed sequences. The rest of the paper is organized as follows. Section 2 gives the problem definition. Section 3 summarizes related work. Sections 4 and 5 present the proposed algorithm and experimental results, respectively. The conclusions and future work are given in Section 6.

## 2. Problem definition

Consider a sequence database with a set of distinct events $I = \{i_1, i_2, i_3, \cdots, i_n\}$, where $i_j$ is an event (or an item), where $1 \le j \le n$. A set of unordered events is called an itemset. Each itemset is put in brackets, for example $(ABC)$. To simplify notation, for itemsets that contain only a single item, the brackets are omitted, for example $B$. A sequence $S = \{e_1, e_2, e_3, \cdots, e_m\}$ is an ordered list of events, where $e_j$ $(1 \le j \le m)$ is an itemset. Suppose that $\ell$ is the number of events in a sequence. A sequence with length $\ell$ is called an $\ell-sequence$. For example, $AB(AE)CB$ is a $6-sequence$. A sequence $S_a = a_1, a_2, \cdots, a_m$ is contained in another sequence $S_b = b_1, b_2, \cdots, b_n$ if there exist integers $1 \le i_1 < i_2 < \cdots < i_m \le n$ such that $a_i = b_{i1}$, $a_2 = b_{i2}, \cdots, a_m = b_{im}$. If sequence $S_a$ is contained in sequence $S_b$, $S_a$ is called a subsequence of $S_b$ and $S_b$ is called a supersequence of $S_a$, denoted as $S_a \subseteq S_b$. A sequence database is denoted as $D = \{s_1, s_2, s_3, \cdots, s_{|D|}\}$, where $|D|$ is the number of sequences in $D$ and $s_i$ $(1 \le i \le |D|)$ is a transaction in the form *ID*, *Sequence*, where the attribute *ID* is used to describe the information of $s_i$ corresponding to transaction information over time.

The absolute support (support) of a sequence $S_a$ in a sequence database $D$ is calculated as the number of occurrences of $S_a$ in the

* Corresponding author.
*E-mail addresses:* minhthai@itc.edu.vn (M.-T. Tran), lhbac@fit.hcmus.edu.vn (B. Le), bayvodinh@gmail.com (B. Vo).

http://dx.doi.org/10.1016/j.engappai.2014.10.021
0952-1976/

**Table 1**
Example sequence database *D*.

| ID | Sequence |
|---|---|
| 1 | CAA(AC) |
| 2 | AB(ABC)B |
| 3 | A(BC)ABCE |
| 4 | AB(BC)AD |

transactions of $D$, denoted as $\sup^D(S_a)$. The support of a sequence is given in the notation *sequence* : *support*. For example, a sequence $AB$ with support 3 is represented as $AB:3$.

Given a minimum support threshold *minSup*, a sequence $S_a$ is a frequent sequence on $D$ if $sup^D(S_a) \geq minSup$. If sequence $S_a$ is frequent and there exists no proper supersequence $S_b$ of $S_a$ with the same support, $S_a$ is called a frequent closed sequence, i.e., there does not exist $S_b$ such that $S_a \subseteq S_b$ and $sup^D(S_a) = sup^D(S_b)$. The problem of mining frequent closed sequences is to find a complete set of frequent closed sequences for an input sequence database $D$ and a given minimum support threshold *minSup*.

**Example 1.** Consider the sequence database in Table 1. The database has five unique items $I = \{A, B, C, D, E\}$ and four transactions, i.e., $|D| = 4$. Assume that the minimum support threshold is $minSup = 2$ (50%). If all frequent sequences of $D$ are mined with the given *minSup*, the following 32 sequences are obtained: $S_{FS} = \{A:4, AA:4, AB:3, AC:4, (AC):2, AAB:2, AAC:2, A(AC):2, ABA:3, ABB:3, ABC:3, A(BC):3, ACA:2, ACB:2, ABAB:2, AB(BC):2, A(BC)A:2, A(BC)B:2, B:3, BA:3, BB:3, BC:3, (BC):3, BAB:2, B(BC):2, (BC)A:2, (BC)B:2, C:4, CA:3, CB:2, CC:2, CAC:2\}$. In contrast, mining the frequent closed sequences yields $S_{FCS} = \{AA:4, AC:4, AAC:2, A(AC):2, ABA:3, ABB:3, ABC:3, A(BC):3, ABAB:2, AB(BC):2, A(BC)A:2, A(BC)B:2, CA:3, CAC:2\}$, which has only 14 sequences.

Frequent closed sequences $S_{FCS}$ are thus more compact than general frequent sequences $S_{FS}$. This is due to subsequence $S_a$ with the same support as that of supersequence $S_b$ being absorbed by $S_b$ without affecting the mining results. For example, sequence $(BC)A:2$ is absorbed by sequence $A(BC)A:2$ because $(BC)A \subseteq A(BC)A$ and $sup^D((BC)A) = sup^D(A(BC)A) = 2$.

At first, the frequent sequences with length 1 are mined from a sequence database. After that, these frequent sequences will combine (or extend) each other to form new candidates with length 2. This process is repeated until there are no new generated frequent sequences. In general, the sequences with length $k$ are used to generate sequences with length $k+1$. Besides the generation of candidates, the checking of frequent closed sequences is applied in each process. The following definitions are used in the process of extending sequences and checking frequent closed sequences.

**Definition 1.** (*substring of a sequence*). Let $S$ be a sequence. $sub_{i,j}(S)$ $(i \leq j)$ is defined as a substring of length $(j-i+1)$ from position $i$ to position $j$ of $S$. For example, $sub_{1,3}(BABC)$ is $BAB$ and $sub_{4,4}(BABC)$ is $C$.

**Definition 2.** (*extending a sequence from a 1-sequence*). Let $\alpha$ and $\beta$ be two frequent 1-sequences. $\{t_\alpha.p_\alpha\}$ and $\{t_\beta.p_\beta\}$ are the transactions and positions of sequences $\alpha$ and $\beta$, respectively. There are two forms of sequence extension

Itemset extension : $\langle(\alpha\beta)\rangle\{t_\beta.p_\beta\}$, if $(\alpha < \beta) \wedge (t_\alpha = t_\beta) \wedge (p_\alpha = p_\beta)$. (2.1)

Sequence extension : $\langle\alpha\beta\rangle\{t_\beta.p_\beta\}$, if $(t_\alpha = t_\beta) \wedge (p_\alpha < p_\beta)$ (2.2)

**Definition 3.** (*extending a sequence from a k-sequence*). Let $\alpha$ and $\beta$ be two frequent $k$-sequences $(k > 1)$, $u = sub_{k,k}(\alpha)$, and $v = sub_{k,k}(\beta)$. $\{t_\alpha.p_\alpha\}$ and $\{t_\beta.p_\beta\}$ are the transactions and positions of sequences $\alpha$ and $\beta$, respectively. There are two forms of sequence extension

Itemset extension : $\alpha +_i \beta = sub_{1,k-1}(\alpha)(uv)\{t_\beta.p_\beta\}$

if $(u < v) \wedge (t_\alpha = t_\beta) \wedge (p_\alpha = p_\beta) \wedge (sub_{1,k-1}(\alpha) = sub_{1,k-1}(\beta))$ (3.1)

Sequence extension : $\alpha +_s \beta = \alpha v\{t_\beta.p_\beta\}$,

if $(t_\alpha = t_\beta) \wedge (p_\alpha < p_\beta) \wedge (sub_{1,k-1}(\alpha) = sub_{1,k-1}(\beta))$ (3.2)

**Definition 4.** Let $S = e_1e_2\cdots e_n$. An item $e'$ can be added to a pattern extension of $S$ in one of three positions

$S' = e_1e_2\cdots e_ne' \wedge (\sup^D(S') = \sup^D(S))$ (4.1)

$\exists i(1 \leq i < n)$such that $S' = e_1e_2\cdots e_ie'\cdots e_n \wedge (sup^D(S') = sup^D(S))$ (4.2)

$S' = e'e_1e_2\cdots e_n \wedge (sup^D(S') = sup^D(S))$ (4.3)

In (4.1), item $e'$ appears after $e_n$, so item $e'$ is called a forward-extension and $S'$ is called a forward-extension sequence. For example, sequence $AC:4$ is a forward-extension of sequence $A:4$ because sequence $C$ is extended after sequence $A$ and their support is 4. In (4.2) and (4.3), item $e'$ appears before $e_n$, so item $e'$ is called a backward-extension and $S'$ is called a backward-extension sequence.

For example, sequence $CAC:2$ is a backward-extension of sequence $CC:2$ because sequence $A$ is extended in the middle of sequence $CC$ and their support is 2.

**Definition 5.** Let $S = e_1e_2\cdots e_n$. The starting position of sequence $S$ is the position of the first appearance of itemset $e_1$. For example, in the sequence $AB(ABC)CB$, the starting position of sequence $(ABC)$ is 3, and that of sequence $ABB$ is 1.

## 3. Related work

Mining frequent sequences was first proposed in 1995 by Agrawal and Srikant with their AprioriAll algorithm, which is based on the Apriori property. Agrawal and Srikant then expanded the mining problem in a general way with the GSP algorithm (Srikant and Agrawal, 1996). Since then, many frequent sequence mining algorithms have been proposed to improve mining efficiency. The algorithms use various approaches for organizing data and storing mined information. Typical algorithms include SPADE (Zaki, 2001), PrefixSpan (Pei et al., 2001), SPAM (Ayres et al., 2002), and LAPIN-SPAM (Yang and Kitsuregawa, 2005). The SPAM algorithm organizes data in a vertical bitmap format and uses a dictionary tree structure to store mined information. PrefixSpan uses database projection for sequence extension to reduce the search space, with the data presented horizontally. The LAPIN-SPAM algorithm uses a list to store the final positions of items and a set of boundary positions of the prefix to reduce the scope of the search space.

Various algorithms have been proposed for mining non-redundant frequent sequences to reduce the required storage space and runtime for mining rules. Frequent closed sequence mining and frequent closed itemset mining algorithms include A-CLOSE (Pasquier et al., 1999), CLOSET (Pei et al., 2000), CHARM (Zaki and Hsiao, 2002), and CLOSET+ (Wang et al., 2003). Most of these algorithms maintain mined frequent itemsets in order to test frequent closed sequences, which require a lot of memory. CLOSET+ uses a two-level hash-index structure and a tree structure for storing the itemsets to reduce memory space and the time required for testing closed itemsets. CloSpan (Yan et al., 2003) uses a maintain-and-test pattern method and combines a hash-index structure with a tree structure for storing sequences. This algorithm prunes patterns