A taxonomy of Sequential Pattern Mining Algorithms

Sedigheh abbasghorbani

Affiliation: Young Researchers and Elite Club, Chalus Branch, Islamic Azad University, Chalus, Iran

Email: mis.abbasghorbani@gmail.com

Abstract

Due to the important applications in today's world such as mining web page traversal sequences, users behavior in buying or disease treatments, many algorithms have been introduced in the area of sequential pattern mining over the last decade, most of which have also been modified to support brief representations like closed, maximal, incremental or hierarchical sequences. This article reviews a number of algorithms in each category and puts them in taxonomy of sequential pattern mining techniques as an application. This article investigates these algorithms by introducing taxonomy for classifying sequential pattern mining algorithms based on their theoretical features and say advantage/disadvantage of them. This classification aims at enhancing understanding of sequential pattern mining problems, current status of provided solutions, and direction of research in this area. This article also attempts to provide a comparative performance analysis of many of the key techniques in time execution and memory usage.

Keywords: Data mining, frequent sequential patterns, apriori-base, pattern-Growth, projected database.

1. Introduction

Sequential pattern mining discovers frequent sub-sequences as patterns in a sequence database. A sequence database stores a number of records, where all records are sequences of ordered events, with or without concrete notions of time. An example sequence database is retail customer transactions or purchase sequences in a grocery store showing, for each customer, the collection of store items they purchased every week for one month. Other examples are scientific experiments, natural disasters, and disease treatments, the analysis of DNA sequences, etc. Sequential pattern mining is an important problem with broad applications, including the analysis of customer purchase behavior, web access patterns, scientific experiments, disease treatment, natural disasters, and protein formations. Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min_support threshold, sequential pattern mining is to find all repeating patterns (known as frequent Sub-sequences). Support of sub-sequences whose occurrence frequency in the set of sequences, are not less than min_support. Frequent Sub-sequences can be used later by end users or management to find associations between the different items or events in their data for purposes such as marketing, business reorganization, prediction and planning. Many previous studies contributed to the efficient mining of sequential patterns or other frequent patterns in time-related data. Srikant and Agrawal generalized their definition of Sequential patterns that include time constraints, sliding time window, and named GSP that improved apriori-based algorithm (Ramakrishnan and Agrawal, 1996). A typical apriori-like sequential pattern mining method, such as GSP (Ramakrishnan and Agrawal, 1996), SPADE (Zaki, 2001), SPAM (Ayres, 2002) and so on adopts a multiple-pass, candidate generation and-test approach. Second approach adopts a divide-and-conquers pattern-growth principle that Sequence databases are recursively projected into a set of smaller projected databases based on the current sequential pattern, and sequential patterns are grown in each projected databases by exploring only locally frequent fragments. These algorithms are like prefixspan (Pei et al, 2006), clospan, (Van and Afshar, 2003) and so on. With the increase in the use of the world wide web for e-commerce businesses, web services, and others, web usage mining is one of the most prevalent application areas of sequential pattern mining in the literature such as WAP-MINE (Pei et al, 2000), PLWAP (Ezeife et al, 2005), BIDE (Wang and han, 2007) and so on. Typical applications of web usage mining fall into the area of user modeling, such as web content personalization, web site reorganization, prefetching and caching, e-commerce, and business intelligence. Many novel algorithms have early pruning methods that enable them to look-ahead to avoid generating infrequent candidate sequences. If a certain item (or sequence) is infrequent, then there is no point in growing the suffix or the prefix of the sequence (due to the downward closure of the apriori property). The algorithms like VMSP (Viger et al, 2014), VGEN (Viger et al, 2014), FSGP (Yi et al, 2011) and MaxSP (Viger et al, 2013) are in this category. This article focuses on sequential pattern-mining techniques based on number of output sequences; some algorithms mine all of sequential patterns and some other mine Partial of sequence patterns.

2. Related concepts

Let $I=\{i_1, i_2, \ldots i_n\}$ be a unique set of items. A sequence S is an ordered list of itemsets, denoted as $<s_1, s_2 \ldots s_m>$, where sj is an itemset which is also called an element of the sequence, and $s_j \subseteq I$ (Yun, 2008). The size |S| of a sequence is the number of itemsets in the sequence. The length, $\mathcal{L}(S)$, is the total number of items in the sequence. A sequence with length l is called an l-sequence. A sequence database, SDB={S1, S2, …, Sn}, is a set of tuples <sid, S>, where sid is a sequence identifier and S_k is an input sequence. A sequence $\alpha=<X_1,X_2,\ldots,X_n>$ is called a sub-sequence ($\alpha \subseteq \beta$) of another sequence

$\beta=<Y_1,Y_2,\ldots,Y_m>$ ($n\leq m$), and β is called a super-sequence of the sequence α if there exist an integer L ($L\leq i_1<i_2<..<i_3\leq m$) such that $X_1\subseteq Y_{i1}$, $X_2\subseteq Y_{i2},\ldots \subseteq X_n\subseteq Y_{in}$. A tuple <sid, S> is said contain the sequence α, if S is super-sequence of α ($\alpha\subseteq s$). The support value of a sequence α in a sequence database (SDB) support (α) ($\text{ऽ}(\alpha)$) is the number of sequences in SDB that contain the sequence α ($\text{ऽ}(\alpha) = |\{<sid, S> \mid (<sid,S> \in DB) \wedge (\alpha\subseteq S)\}$). Given the user defined minimum support threshold (ζ), a sequence α is called a frequent sequential pattern in the sequence database if the support of the sequence α is no less than ζ ($\zeta \geqslant \text{ऽ}(\alpha)$).

Table 1, shows the input SDB in our running example (hypothetical SDB). Assume that a minimum support Threshold is 3 (ζ=3). This SDB has eight unique items (a,b,c,d,e,f), and six input sequences. A sequence S=<a(abc)(ac)d(cf)> in SDB has five itemsets: a, (abc), (ac), d, (cf) where items ‘‘a’’ and ‘‘c’’ appear three times in different itemsets of the sequence. The size of S is 5 (|S|=5) and the length of this sequence is 9 ($\mathcal{L}$(S)=9). Sequence <a(bc)d> is a sub-sequence of <a(abc)(ac)d(cf)>. Additionally, the sequence S=<a(bc)d> is a frequent sequential pattern because sequences 1, 5 and 6 contain sub-sequence <a(bc)d> and the support 3 ($\text{ऽ}(\alpha)$=3) of the sequence is no less than ζ(ζ=3). A sequential pattern S=<a(bc)d> is not a frequent pattern since the support 2 (S is sub-sequence of sequences 1 and 2) of this pattern is less than ζ. Based on the anti-monotone property (apriori property), we can know that all super-patterns of the sequential pattern such as <ag>, sequential patterns such as <a(ab)g>, <acg>, and <a(cd)g> are infrequent patterns, because <ag> is infrequent.

Table 1. A sequence database (hypothetical SDB)

sid	S(Sequence)
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)bc>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>
5	<a(ab)(cd)egh>
6	<a(abd)bc>

Table 2. Part of the implementation of GSP on hypothetical SDB

$\mathcal{L}_3$		$\mathcal{C}_4$		$\mathcal{L}_4$.۱	
Seq-3	ऽ	Seq-4	ऽ	Seq-4	ऽ
<a(ab)>	3	< a(ab)c>	3	< a(ab)c>	3
<aac>	4	<a(ab)d>	×		
<(ab)c>	4	<aacb>	0		
<(ab)d>	3	<aacc>	1		
<abc>	5				
<acb>	3				
<acc>	3				
<adc>	3				
<ebc>	3				

3. All of sequence pattern mining

The initial algorithms mine all of frequent sequential patterns. The hypothetical SDB have 30 frequent sequential patterns and these algorithms mine all of them. In general, these are divided into two categories: apriori-base and pattern-Growth base algorithms.

3.1. Apriori-based algirithms

Apriori-based algorithm has three key features that is independent of how they are implemented: 1) Breadth-first Search; because in the k-th iteration of the algorithm, all sequences of length k (seq-k) are made, 2) Production and testing; which manufactured the candidate sequence and checked the amount of supports, 3) Multiple scan the database; which is a very undesirable feature of most of these algorithms and needs the processing time and the I/O cost.

3.1.1. GSP

GSP (Ramakrishnan and Agrawal, 1996) is apriori-based algorithm that generate candidate. If the candidates are not fit with memory, GSP produces only the number of candidates that are according to the memory (Agrawal and Srikant, 19995). This algorithm is combined apriori algorithm with time constraints. The variables in this algorithm include the maximum gap, minimum gap and window size. GSP As well as apriori-base algorithms is including both generate candidate and count support value. In it, a candidate with adjacent sub-seq-(K-1), which its support value is lower than ζ , is eliminated. <c> is adjacent sub-sequence of S = <s_1 ... s_n> if have one of two fallow condition: 1) c is derived from S by removing an item from s_1 or s_n; 2) c is derived from S by removing an item from itemset s_i that it have at least two items. In table 2 the following sequence of <(ab)c>, <a(abb)> and <abc> are adjacent sub-sequence of s = <a(ab)c> at C_4 (C_4 is set of candidate seq-4). Sequence <a(ab)b> has adjacent sub-sequence <aad> and given that there is not this sub-sequence in L_3 (L_3 is set of frequent seq-3), so the sequence <a(ab)d> is infrequent. Later one algorithm was created based on GPS, known as MFS (Mining Frequent Sequences) (Zhang et al. 2001) that suggests a two-step algorithm rather than multiple scans of database. MFS produce same set of frequent sequences that GSP produce, while MFS have better performance from GSP by reducing I/O cost (Zhao and Bhowmick, 2003).

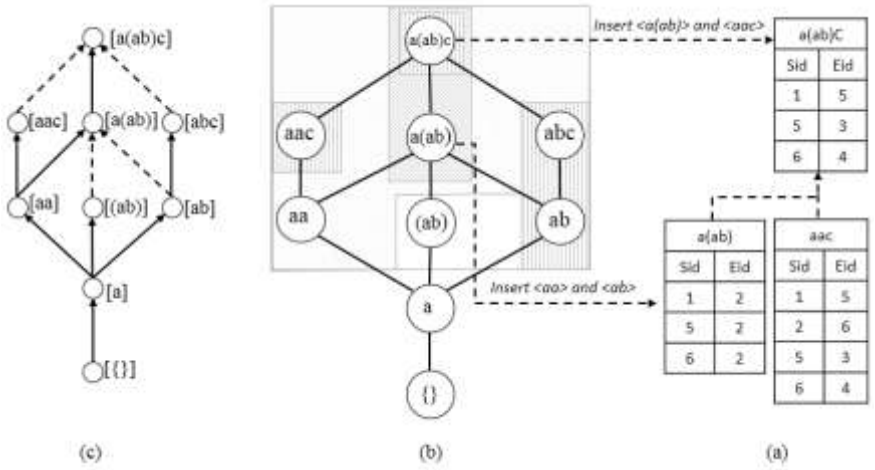


fig 1. Search strategies in network for hypothetical SDB a) showing some sequence's ID-List b) division of equivalence classes [a] to smaller classes and c) create a path for navigation network traversal

3.1.2. SPADE (Sequential PAttern Discovery using Equivalence classes)

SPADE (Zaki, 2001) proposed an algorithm to find frequent sequences using the network search techniques. In this algorithm, all sequences are discovered only by three passes from the database (Zhao and Bhowmick, 2003). This approach has three key features: 1) it uses from a vertical database format (id-list), 2) it use From a network theory to analyze the main search space into smaller spaces to reduce I/O costs and 3) it uses from two Breadth-first and deep- frist search strategies for count frequent sequences in per subnet. Figure 1, show these three features in this paper SDB (hypothetical SDB). Figure 1(a), shown vertical ID-List, and calculating the amount of support by it. ID-List rows

are shown with pairs <Sid, Eid> were Sid is sequence ID and Eid is transaction ID. If in sequence $X \in S$, x1 and x2 represent sub-sequence of prifix sub-seq-(k-2) from X then $X=x_1 \vee x_2$ and $s=|\mathcal{V}(X_1) \cap \mathcal{V}(X_2)|$. For example, the ID-list of X=<a(ab)c)> produced from *temporal join* of x_1=<a(ab)> and <x_2=<aac> (figure 1(a)). Figure 1(b), shows equivalence class for mother class {[a]} ({[aa], [ab], [a(ab)], [aac], [a(ab)c]}). In this is one subnet of the main network of hypothetical SDB. In Figure 1(c) solid and dash arrow drawn based on classification of figure 1(a). For example, the arrow between [aa] and [aac] is solid because both them are in class [aa]. Finally it uses both depth-first and Breadth-first search (BFS and DFS) strategies for calculating frequent sequences. Because of BFS process the lower level first, it prepare much more information for pruning than DFS. On the other hand, DFS hold only ID-List of two consecutive classes in a single path, thus it needs less main memory than BFS. With increasing number of frequent sequences (eg, very small support threshold value), may be the only practical method is DFS.

3.1.3. SPAM (Sequential Pattern Mining)

SPAM (Ayres, 2002) uses the depth-first search strategy for mining sequential patterns. An additional salient feature of SPAM is its property of online outputting sequential patterns of different length; compare this to a breadth-first search strategy that first outputs all patterns of length one, then all patterns of length two, and so on. On the other hand, this algorithm utilizes a depth-first traversal of the search space combined with a vertical bitmap representation to store each sequence. SPAM refer to the process of generating sequence as the sequence-extension step (abbreviated, the S-step), and it refer to the process of generating itemset extended sequences as the itemset-extension step (abbreviated, the I-step). SPAM, components including bitmap representation, S-step/I-step traversal, and S-step/I-step pruning all contribute to this excellent runtime.

CM-SPADE and CM-SPAM (Philippe Fournier-Viger et al. 2014) algorithms are respectively Versus of SPADE and SPAM. Co-occurrence Map (CMAP) is a new structure to store co-occurrence information. CM-SPADE checks pruning for each candidate but CM-SPAM checks each candidate for pruning. In these two algorithms, it is necessary to check for the two last items in CMAPs (stores every items that succeeds each item by s-extension at least min_sup times) and CMAPi (stores every items that succeeds each item by i-extension at least min_sup times) and it is also perform prefix pruning for CM-SPAM. Results show that the modified algorithms, prune a large amount of candidates, and are up to eight times faster than the corresponding original algorithms and that CM-SPADE, CM-SPAM have respectively the best performance for sequential pattern mining and closed sequential pattern mining.

3.2. Pattern-Growth base algorithms

Pattern-Growth base algorithms (Pei et al, 2000) emerged as a solution for the problems of previously algorithms (production and testing problem) after the Apriori-based methods. The main idea of this algorithm is based on avoidance from production candidate and search limited part of the original SDB (Khan and Jain, 2012). Almost all algorithms in this category are begin with construction representative of the main SDB and then one method is defined for segmenting the search space.

3.2.1. WAP-MINE

WAP-MINE (Pei et al, 2000) was created for mining access patterns in the site (As its name suggests). This algorithm consists of two key components: 1) wap-tree (YI, 2005), 2) sequence mining from wap-tree. This algorithm keeps the first occurrence of any item in the frequent itemset, in the header table, to use these items to connect to all nodes of the same type of tree. It should be noted that because WAP-MINE is a method to mining the access pattern of users in websites, There is not

transaction with synchronous items in user sequences (For example, a user cannot click simultaneously on the three links). Therefore, in the hypothetical SDB, the sequences that containing the transaction with synchronous items, one of its items is selected and then, Table 3 created. Figure 2(a) shows the wap-tree for hypothetical SDB (Table 3) and figure 2(b) shows final wap-tree based on c transaction. Dash lines specifies Find path of frequent sequences for any item in header table. This algorithm uses conditional search instead of breadth-based searching (as mentioned in Apriori-based algorithm). Since wap-tree is much smaller than the original database size, therefore mining sequences in wap-tree easier. On the other hand, wap-tree construction with double-scan on SDB is the quite efficient (Nizar and Ezeife, 2010). This algorithm suffers from memory consumption, because the wap-tree algorithm does several renovations during pattern mining. To resolve this problem, PLWAP algorithm (EZEIFE et al, 2005) was presented that converts wap-tree to the binary tree before mining frequent patterns (YI, 2005).

Table 3. Edited SDB for the web access (for algorithm WAP-MINE)

sid	cid	S(Sequence)	Frequent Sub-sequences
1	100	<abadc>	<abadc>
2	200	<acbabc>	<acbabc>
3	300	<fbdcb>	<bdcb>
4	400	<egacbc>	<acbc>
5	500	<aadegh>	<aad>
6	600	<aabc>	<aabc>

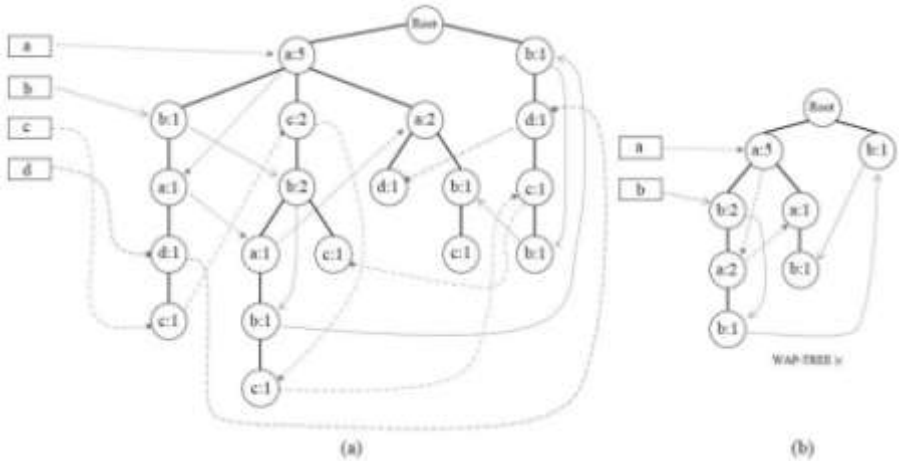


fig 2. Mine frequent sequences whit WAP-tree a) the wap-tree for hypothetical SDB, b) final wap-tree based on c transaction

fig 3. database projection a) level-by-level b) bi-level 3) pseudo projection for hypothetical SDB

3.2.2. Prefixspan

Prefixspan (Pei et al, 2001) use database projection until SDB made much smaller for the next pass and thus faster algorithm and therefore is able to operate in large database. Figure 3(a) shown database projection for frequent seq-1 (for example, the prefix <a> in sequence 1 and in hypothetical SDB, arguments will be as <(abc)(ac)d(cf)> that is suffix of <a>). PrefixSpan have different design methods for database projection including level-by-level projection, bi-level projection and pseudo projection. Bi-level projection (Figure 3(b)) method is design triangular matrix M instead of database projection n*n (level-by-level projection). This matrix show **ֆ** for all seq-2, for example **(**2,5,4)=M[<(b)>,<(a)>] which means that the **ֆ(**<(a), (b)>), **ֆ(**<(b), (a)>) and **ֆ(**<(b, a)>) respectively 2, 5 and 4. The other method is pseudo projection that be not made any physical database and any suffix are shown by a pair offset values and pointers, for example in Figure 3(c), pseudo projection for sequence S_1 = <a(abc)(ac)d(cf)> According to the prefix <a>, the first item is <a> in first place in sequence S1 and so its suffix (<(abc)(ac)d(cf)>) place in point 2 of S_1 and so the final amount of the sequence offset in the prefix <a> are {6, 3, 2}. Note that in this table (Figure 3(c)), ∅ suggests that there is no prefix used in sequence and $ implies that prefix in the sequence, but Suffix is null. Pseudo projection method avoid from physical copy of suffixes, so it is effective in time and place of implementation (Zaki, 2001). Based on the observation, if the original SDB or projection database is too large for placement in the main memory, used the physical design (Pei and Han, 2004).

Table 7. Output sequences of a) Closed sequential patterns mining algorithms and b) Maximal sequential pattern algorithms

a			
sequence	**support**	**sequence**	**support**
<d>	5	<adc>	3
<abc>	5	<bb>	3
<ab>	6	<db>	3
<ac>	6	<a(ab)c>	3
<bc>	6	<dc>	4
<acc>	3	<aac>	4
<f>	3	<e>	4
<acb>	3	<ad>	4
<(ab)d>	3	<(ab)c>	4

<ebc>	3		

b	
Sequence	**support**
<f>	3
<ebc>	3
<adc>	3

3.2.3. Multi-dimensional sequential patterns mining

In addition to the sequential patterns in one dimension that Considers only one feature with time stamps for finding frequent patterns, mine multidimensional sequential patterns can be obtained more and more useful information, For example, students always buy A and after a week will purchase B, while these laws are not arranged for other buyers. The purpose of this particular type of exploration is mining sequential patterns more desirable from different multi-dimensional features. In this algorithm, three dimensions (customer group, city and age group) are considered in the hypothetical SDB (table 6(a)). Three different methods used to mine multi-dimensional sequence patterns (Pinto and et al, 2001): 1) UNI-SEQ method that in which the multi-dimensional SDB (table 6(a)) be converted to MD-extension database. For example, in tuple t=(1, business, tonekabon, middle, <a(abc)(ac)d(cf)>) from the multi-dimensional SDB, the sequence S=<a(abc)(ac)d(cf)> at t can be extended to SMD=<(business tonekabon middle) a(abc)(ac)d(cf)> (SMD: multidimensional sequence)(table 6(b)). 2) dim-seq method that divide multi-dimensional SDB into two following parts: dimension information (X_l, ..., X_m) and sequence s_t. In this method, first multiple multi-dimensional compounds and then related sequence patterns can be found. For example, tuple (*, tonekabon, *) is frequent because there is in three sequences (1, 4 and 6). These compounds of frequently multi-dimensional values are called MD-patterns. Then all sequences in accordance tuple P = (*, tonekabon, *) are collected. At the end, projected database of multi-dimensional pattern (MD-projected database) for P is established - and show like SDB|p – and sequence patterns can be mined from it. Here have SDB|p={<a(abc)(ac)d(cf)>, <eg(af)cbc>, <a(abd)bc>}. Based on the sequence pattern like <ac>, (*, tonekabon, *, <ac>) is a multi-dimensional sequence pattern. 3) seq-dim method that is same as dim-seq but have one different; in seq-dim, first sequence patterns are mined and then related MD-patterns are found. For example, after scan multi-dimensional SDB, sequence S=<(ab)> is identified and then all information related to S Be collected in multidimensional tuple t = {(business tonekabon middle), (education esfahan middle), (business shiraz middle) , (business tonekabon middle)} and then MD-projected database for t (SDB|t) is established and MD-patterns mined from it. Here, (business, *, *) is a MD-patterns and so (business, *, *, <(ab)>) is a multi-dimensional sequence pattern.

4. Partial sequence pattern mining

A user often demanding a small set of patterns because, mining the full set of sequence patterns can be difficult to understand and use patterns. To overcome these problems, user use constraints that often represents her (or his) focus and interests on outputs of algorithms. Some of these algorithms mine frequent sequence patterns based on constraints, other mine closed or maximal patterns and also some of them mine sequential generator patterns.

4.1. Constraints base algorithm

Pei and Han expressed, seven of the constraints (Pei and Han, 2002), (Pei et al, 2002): 1) Item constraints indicates a set of items that should or should not present in sequences; 2) Length constraints (C_{len}), such as $C_{len}(\alpha) \equiv (L(\alpha) \geq 50$ that show minimum length of sequences are 50; 3) Super-pattern constraints: such as $C_{pat}(\alpha) \equiv <(Graphic)(computer)> \subseteq \alpha$ that find sequence patterns that user bough graphics card in first and then digital computer; 4) Cumulative constraints on addition, subtraction, maximum, minimum, division and so on; 5) regular statements constraints (C_{RE}) that apply to the collection of items. Of course a regular express is determined on the alphabet OF sequence elements and with using operators such as the disjunction (|) and Kleene closure (*) (for example {CRE≡<milk*{Bread|(Bread, Yogurt)|Butter}); 6) Distance constraints is applied in SDB that a transaction in any sequence have time stamp. In general, the Distance constraints are shown like "Dur (α) Θ Δt" where $\Theta \in \{\leq, \geq\}$ and Δt is a integer number; 7) Gap constraints is applied in SDB that a transaction in any sequence have time stamp (Dur(α) Θ Δt). Constraints are available in both monotonic and anti- monotonic. A CM constraint is monotonic if a sequence as α satisfies the CM constraint, then its Super-sequences also satisfy CM. on the other hand, a CA constraint is anti-monotonic if a sequence as α satisfies the CM constraint, then non-empty sub-sequences of α also satisfy CA.

4.1.1. Prefix Growth

Prefix-Growth (Pei et al, 2002) uses prefixspan algorithm by applying a user-defined limits for sequence patterns mining. C_{pm} constraint is prefix monotonic if for every sequence α and for the sequences that has α as a prefix of themselves, satisfies the constraint. C_{pu} constraint is prefix anti-monotonic if for every sequence α and for every prefix of it, satisfies the constraint. The constraint is prefix monotonic or anti-monotonic, is said prifix-monotone that uses prefix-growth in itself.

4.1.2. SPRIT

SPIRIT family algorithms (Garofalakis et al, 1999) use regular statements (CRE) to constraints (with different degrees of limitations). Four different SPIRIT algorithms based on their constraints are exist [22]. These algorithms act differently in produce candidate sequences and candidate pruning.. These algorithms based on their limitation are as follow: SPIRIT (N) (N represents Naive), SPIRIT (L) (L represents legal), SPIRIT (V) (V represents valid) and SPIRIT (R) (R represents Regular). On the other hand, Relationship between their set of frequent seq-k (F_k) is as follow: F_k SPIRIT(R)$\subseteq$ F_k SPIRIT(V)$\subseteq$ F_k SPIRIT(L)$\subseteq$ F_k SPIRIT(N).

4.2. Closed sequential patterns mining

When long sequential patterns or small support Threshold are used, mining algorithms performance drastically reduced (Khan and Jain, 2012). This is not surprising; Suppose the SDB have only a long sequence $<(a_1)(a_2)...(a_{100})>$, This sequence may be produce 2^{100}-1 frequent sequences if ζ nearly 1. An alternative but equally powerful way is mining only frequent closed sequences instead of mining full set of all frequent sequences. Closed sequential patterns called for a sequence that contains no super-sequences with the same support. Let FS set of all sequence patterns, in the case of CS as a closed sequential patterns is defined as follows: $CS=\{\alpha | \alpha \in F_S$ And there is no $\beta \in F_S$ such that $\alpha \prec \beta$ and $s(\alpha)=s(\beta)\}$. In hypothetical SDB, CS has closed 19 frequent sequences ({ac, ab, bc, aac, e, a (ab) c, dc, acc, acb, (ab) d, ebc, (ab) c, ad, abc, adc, bb, db , d, f}), While FS has 30 frequent sequences.

4.2.1. BIDE+ (BI-Directional Extension)

BIDE+ algorithm (Wang and han, 2007) is actually developed based on PrefixSpan algorithm. Currently, most frequent closed sequential patterns mining algorithms as CLOSET (Pei et al, 2001), CHARM (Zaki and Hsiao, 2002), TFP (Han et al, 2002) and CloSpan need to maintain a set of closed frequent patterns (or candidates) in memory, until new sequences are mined. For avoid maintenance closed mined sequences, BIDE+ uses bi-directional search (BIDE paradigm is proposed for mining closed sequences without candidate maintenance). This search uses two concepts: Backward development event and Control of forward development event. let $sp=e_1,e_2,\ldots e_n$ as a prefix seq-n and e' one item. If for any of s'p= e1,e2,…,ei-1,e',ei,…,en (for 1 <i≤n) or s'p=e',e1,e2,…en (for i = 1) exist ʂ(sp)= ʂ(s'p), then e' is a Backward development event of the prefix sp and sp is not closed. Full forward development event set for Sp prefix sequence is locally frequent items set (Items obtained from scanning sp projection database that their support values are equal to ʂ(sp). Closed control bi-directional search is in this case, if any event in forward development and Backward development of sp prefix does not exist, then sp is closed sequence otherwise, sp necessarily is not closed sequence. So, the forward directional extension is used to grow the prefix patterns and also check the closure of prefix patterns, while the backward directional extension can be used to both check closure of a prefix pattern and prune the search space. Of course, other algorithms were developed to mine closed sequences, such as, CLUSEQ (Yang and Wang, 2003) that use clustering sequences and SeqIndex (Cheng et al, 2005) that use a parallel and combination of previous methods (Cong et al, 2005) to mine frequent sequences.

4.2.2. Clasp (Closed Sequential Patterns algorithm)

ClaSP algorithm (Gomariz et al, 2013) has two major phases: 1) The first phase will produce a subset of FS that that is called frequent closed candidate (FCC) and kept in main memory; 2) the second phase is a post-pruning phase to eliminate all sequences that are not closed in FCC and finally acquires accurate frequent closed sequence (FCS). To prune the search space, ClaSP uses the access control method like pruning method of CloSpan algorithm (Van and Afshar, 2003). This pruning method remove non closed sequences and find new closed sequences by controlling the closed sequence are mined previously. For Sequence S <s1, s2, ..., sn> and item α, s◊α means accession α to S. This accession can be I-extension (Itemset extension) that just $s◊iα=<t_1, \ldots t_m \cup \{α\}>$ If $\forall k \in t_m$ have k<α or can be S-extension (sequence extension), which is $s◊iα=<t_1, \ldots t_m, α\}>$. For example, <(ab)> is I-extension and <ab> is S-extension related to <(a)>. This method tries to find for all sequences $\mathcal{P}=<αe_j>$ and $\mathcal{P}'=<αe_ie_j>$ that all are events P in P'. Typically, if at any time find α that contain e_J, while an item e_i is between them, in which case, algorithm safely avoidance from check subtree p. ClaSP use some pruning rules to mine closed sequences easier: 1) For each sequence like S, if among all the sequences in its projection database (PD), an item α always happen before the item β (I-extension or S-extension), in this case, PD◊α◊β=PD◊β, So ∀ γ, s◊β◊γ is not closed. On the other hand, we do not need to search any sequences in the branch s◊β. For example, in projection database for itemset <(ab)> ($PD_{<(ab)>}$={<(-c)cdc>, <dc>, <(cd)>, <(-d)c)>}), in hypothetical SDB, Given that <(cd)> and <cd> are available in all sub-sequences of PD, PD<(ab)d> and PD<(ab)f > both are closed and must be examined separately; 2) If S⊆S' and L(DP(S))=L(DP(S')) in this case, for every γ of ʂ(S◊γ)= ʂ(S'◊γ). For example, in hypothetical SDB, $DP_{<ac>}=DP_{<c>}$={<(c)(cf)>,<(bc)(bc>, <b>, <bc>, <(c) }and $L(DP_{<ac>}) = L(DP_{<c>})$, thereby ʂ(<acb)>= ʂ(<cb>). Consider the above meaning, do not need to grow <ac>, as all children of <c> is the same as <ac> children and vice versa and also they have same support value, so a sequence that begins with c, can include sequences with <ac> prefix.

CM-ClaSP (Philippe Fournier-Viger et al. 2014) algorithm is Versus of ClaSP. This algorithm performance is like CM-SPAM that is noted in section 3.1. Results show CM-SPAM also prune a

large amount of candidates, and are up to eight times faster than the corresponding original algorithms and that CM-ClaSP have best performance for sequential pattern mining and closed sequential pattern mining. Table 4 shows five datasets characteristics that are used in this paper for compare. Table 5 shows percentage of Candidate reduction in CM-SPADE, CM-SPAM and CM-ClaSP from SPADE, SPAM and ClaSP at different min_sups. In average, the Number of candidates pruned are between 50-98% for all but the Snake dataset (becuse Snake is very dense). Generally the number of pruned candidates decreases when min_sup is set lower.

Table 4 . DataSets characteristics

	Sequence count	Item count	Avg.Seq.Length	Type of data
BMS	59601	497	2.51	Web Click Stream
Kosarak	10000	10094	8.14	Web Click Stream
Snake	163	60	60	Protein sequences
sing	800	256	65	sign language
FIFA	20450	2990	34.74	Web Click Stream

Table 5. percentage of Candidate reduction in CM-SPADE, CM-SPAM and CM-ClaSP

	Our DS	BMS	kosarak	Snake	Sing	fifa
CM-SPADE	23% - 66%	75% - 76%	98%	25% - 26%	69%	63% - 69%
CM-SPAM	23% - 74%	78% - 93%	94% - 98%	28%	63%	61% - 63%
CM-ClaSP	23% - 62%	79% - 93%	75%	18%	63%	67% - 68%

Table 6. a) hypothetical SDB with customer group, city and age group fields, b) multi-dimensional SDB (SMD)

sid	cust-grp	city	age-grp	S(Sequence)
1	business	tonekabon	middle	<a(abc)(ac)d(cf)>
2	professional	esfahan	young	<(ad)c(bc)(ae)bc>
3	education	esfahan	middle	<(ef)(ab)(df)cb>
4	education	tonekabon	retired	<eg(af)cbc>
5	business	shiraz	middle	<a(ab)(cd)egh>
6	business	tonekabon	middle	<a(abd)bc>

sid	Multidimensional sequences
1	< (business tonekabon middle) a(abc)(ac)d(cf)>
2	< (professional esfahan young) (ad)c(bc)(ae)bc>
3	< (education esfahan middle) (ef)(ab)(df)cb>
4	< (education tonekabon retired) eg(af)cbc>
5	< (business shiraz middle) a(ab)(cd)egh>
6	< (business tonekabon middle) a(abd)bc>

4.3. Maximal sequential pattern mining

As noted, closed sequential patterns are not included in another pattern having the same support. These category of algorithms, have lossless patterns but their output set is still quite large for some applications. Maximal sequential patterns Developed against Closed sequential patterns. In maximal sequential pattern algorithms, patterns are not included in another pattern. In other word, maximal sequential patterns are not included in other patterns. This category of algorithms have lossless patterns with an extra database scan but it is generally much smaller than closed patterns, for example, in our DS, in Closed sequential pattern algorithms we have 19 frequent sequence pattern but in Maximal sequential pattern algorithms, we have 3 frequent sequence pattern in output that These patterns included Other patterns (table 7(a) and table 7(b)).

4.3.1. MaxSP (Maximal Sequential Pattern miner)

MaxSP (Philippe Fournier-Viger et al, 2013) is a new pattern-growth algorithm to discover maximal sequential patterns. it uses Two steps to check if a pattern *S* is maximal: 1) maximal-forward extension checking: if an item is found in a SDB that can be appended to *S* (after the last item), then *S* is not maximal, 2) maximal-backward extension checking: in this checking, the sequences Scanned To find sequences that containing S. if an item can be appended to S before the first item, or between any two items of S, then S is not maximal. This algorithm is Versus BIDE (without storing intermediate candidates in memory). MaxSP consumes less memory because less sequence need to be scanned and less patterns need to be created. In other word, MaxSP outperforms the BIDE algorithm in execution time and memory, and has better scalability. Its output is closed and maximal (CM) or maximal (M) sequences (table 8).

Table 8. MaxSP run on hypothetical SDB

Sequence	support	Statuse	Sequence	support	Statuse
<a(ab)c>	3	CM	<(ab)d>	3	CM
<acc>	3	CM	<bb>	3	CM
<acb>	3	CM	<db>	3	CM
<adc>	3	M	<ebc>	3	M

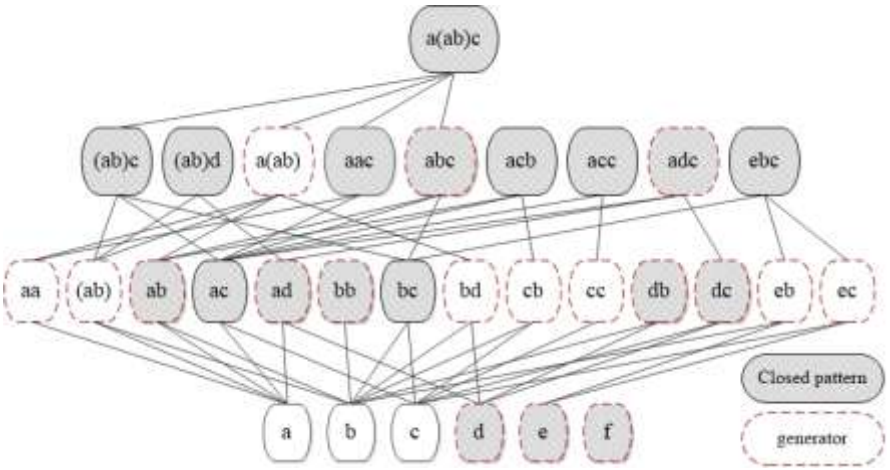


Fig 4. All closed/generator sequential patterns in our SDB

4.3.2. VMSP (Vertical mining of Maximal Sequential Patterns)

VMSP (Philippe Fournier-Viger et al, 2014) is a new vertical algorithm to discover maximal sequential patterns. This algorithm includes three novel strategies: 1) EFN (Efficient Filtering of Non maximal patterns): EFN States that A pattern cannot be contained in another pattern if its support is smaller or A pattern cannot contain another pattern if its support is larger. 2) FME (Forward-Maximal Extension): FME checks the algorithm performs a depth-first search. This algorithm grows patterns by appending items to smaller patterns one item at a time. In FME, checking super-pattern for a pattern **S** can be ignored if the recursive call to the search procedure with **S** produces a frequent pattern. 3) CPC (Candidate pruning with Co-occurrence map): Pruning in CPC formed in this way that for a pattern S, an i-extension (s-extension) with an item x will result in an infrequent patterns if there exists a pair of items in the resulting pattern that is not in $CMAP_i$ ($CMAP_S$). VMSP is up to 100 times faster than MaxSP.

4.4. Mining sequential generator patterns

As noted, A sequential pattern α is said to be closed if there isn't other sequential patterns like β, such that $\alpha \subseteq \beta$, and their supports are equal. A sequential pattern α is said to be a generator if there isn't

other sequential patterns like β, such that $\beta \subseteq \alpha$, and their supports are equal. The problem of mining closed (generator) sequential patterns is to discover the set of closed (generator) sequential patterns. Figure 4, show closed and generator sequence pattern in our SDB (ζ=3). There are 30 sequential patterns, such that 19 are closed (identified by a gray color) and only 18 are generators (identified by a dashed red line). It can be observed that in this case, the number of generators is less than the number of closed patterns.

4.4.1. FSGP (Frequent Sequential Generators Patterns)

Generally, the time cost of the existing algorithms for mining sequential generator patterns is great. FSGP (Yi et al, 2011) is one algorithm for mining sequential generator patterns with the safe pruning strategy that consuming a little time cost and the mechanism of sequential generators checking fast. The safe pruning strategy is that for Given a sequence $S_n=e_1e_2\ldots e_{n-1}e_n$, its one sub-sequence $S_{n-1}=e_1e_2\ldots e_{n-3}e_{n-2}e_n$ while the i^{th} item is removed from the sequence S_n . If the sequence S_n substitute for the sequence S_{n-1} safely with reference to SDB, then the sequence S_n can be pruned safely. In fact, safe pruning is better than the pruning method in the algorithm including the forward pruning and the backward pruning which consume very much time and space cost.

4.4.2. VGEN (Vertical sequential generator miner)

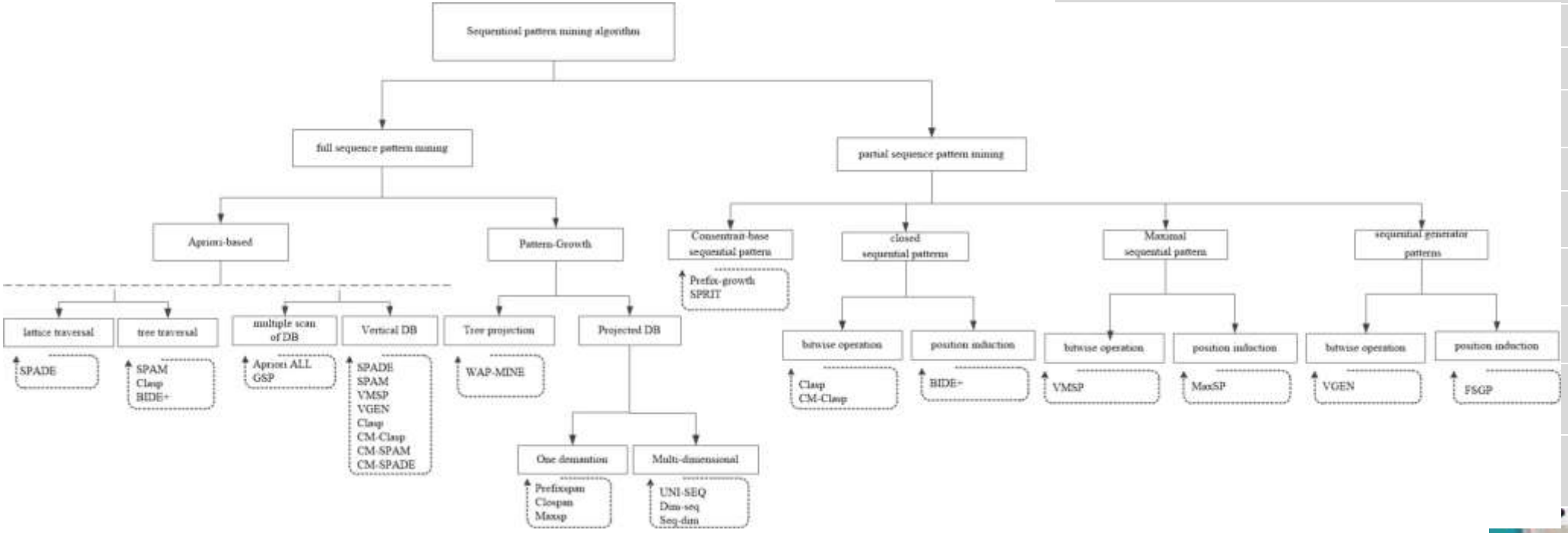
VGEN (Philippe Fournier-Viger et al, 2014) is a new vertical algorithm to discover generator sequential patterns. It includes three novel strategies to efficiently identify generators and prune the search space: 1) ENG (Ecient ltering of Non-Generator patterns): ENG is performed using a novel structure named Z that stores the set of generator patterns found so far. In fact, by using the above strategy, it is obvious that when the search procedure terminates, Z contains the set of sequential generator patterns, 2) BEC (Backward Extension checking): BEC strategy aims at avoiding sub-pattern checks. In other word, During sub-pattern checking for a pattern α, if a smaller pattern β can be found in Z such that the projected database is identical, then α is not a generator as well as any extension of α, and 3) CPC (Candidate Pruning by Co-occurrence map): Pruning in CPC formed in this way that for a pattern S, an i-extension (s-extension) with an item x will result in an infrequent patterns if there exists a pair of items in the resulting pattern that is not in $CMAP_i$ ($CMAP_S$).

5. A taxonomy of Sequential Pattern Mining Algorithms

Much research has been done in sequence patterns algorithms Categories area that each have been studied one or more aspect of the characteristics of these algorithms. For example, in (Chandra and Sammulal, 2013) algorithm is divided into two categories: basic and extended and (Nizar and Ezeife, 2010) has provided a more complete, Or (Nizar and Ezeife, 2010) has provided a more complete Category of sequential patterns mining algorithms - and the algorithms of this paper are placed in it- that categorized The main techniques into apriori-based, pattern-growth, early-pruning, and hybrids of these three techniques. This article has provided new taxonomy for these algorithms, that another aspect to be considered in it. Figure **5** provides a top-down hierarchical view of the taxonomy and the said algorithms in this paper are showed on its right category. This article focuses on sequential pattern-mining techniques based on number of output sequences; some algorithms mine all of sequential patterns and some other mine Partial of sequence patterns. So, The main techniques can be categorized into apriori-based, pattern-growth as the algorithms that mine all of sequential patterns, and others that mine Partial of sequence patterns. Each algorithm is that in both branches of Taxonomy (full sequence pattern mining and partial sequence pattern mining), generated only Part of frequent sequence patterns but it have apriori-base or pattern-Growth base characteristic. The main

characteristics of an Apriori-based algorithm is (1) to perform a breadth-first search and (2) combine pairs of patterns to generate larger patterns, and (3) to scan the database to compute the support of candidates. In SPAM and ClaSP, characteristics (1), (2) and (3) are not respected, since they use a depth-first search, combine items with patterns, and use bitwise operations to calculate support without scanning the database but because of generate candidates in SPAM, it is apriori-based algorithm. For BIDE, characteristics, (1), (2), are also not respected, since it use a depth-first search, does not generate candidate, and use a pattern-growth approach with database projections. For SPADE, there could be a case that it has some similarities to Apriori, especially if it is use with a breadth-first search but it can also be used with a depth-first search, and it uses a vertical database rather than an horizontal database. All algorithms based on SPAM such as CM-SPAM, VGEN, VMSP and so on, also generate candidates but they have different number of frequent sequence pattern in themselves output. Therefore, they were placed in different branch in the taxonomy, for example, VGEN mine sequential generator patterns and because the hypothetical SDB have 30 frequent sequential patterns, such that only 18 of them are generators, then VGEN, mine Partial of sequence patterns. On the other hand, the algorithms such SPAM and SPADE mine all 30 sequential patterns. These algorithm scan the database only once to create a structure sometimes called sid-list for each pattern containing a single item (a vertical database structure is used the sid-lists rather than an hypothetical SDB) and Then, these algorithms will recursively generate candidate patterns of larger size. For each candidate pattern generated, these algorithms will not scan the database to calculate their support. They will instead make a join operation (also called intersection operation) to calculate the sid-list of the pattern by joining sid-lists of smaller patterns. The sid-list allows calculate directly the support of a pattern without scanning the database. Thus, these algorithms only need to scan the database once to create the initial sid-lists of patterns containing single items and are in Apriori-base branch. Pattern-growth algorithms have been tested extensively on mining the web log and found to be fast. Some of These algorithms make projected database (or project a new hypothetical SDB) like Clospan, prefixspan, UNI-SEQ and so on, and others make projected trees like WAP-MINE and MaxSP. In general, a user often demanding a small set of patterns because, mining the full set of sequence patterns can be difficult to understand and use patterns. To overcome these problems, user use constraints that often represents her (or his) focus and interests on outputs of algorithms. Prefix-growth and SPRIT are the examples Of these algorithms. Many algorithms like BIDE and Clasp – in hypothetical SDB - mined only 19 closed sequential patterns of all (30 sequential patterns). Bitwise operations calculate support value at each iteration. Position induction enables an algorithm to look-ahead to avoid generating infrequent candidate sequences. The other view, some methods as SPADE, SPAM, VMSP and VGEN include bitwise or logical operations and some use positional tables as MaxSP and FSGP (These algorithms utilize a sort of position induction to prune candidate sequences very early in the mining process and to avoid support counting as much as possible) and others store support along with each item in a tree structure. One disadvantage of bitmap representations is the amount of computation incurred by bitwise operations used to count the support for each candidate sequence (Nizar and Ezeife, 2010).

Advantages and Disadvantag
Some candidates are produced may not exist in the database (Zhao and Bhowmick, 2003).
– It need only two scan of DB; – High memory usage.
– High cost for combine ID-lists (Khan and Jain, 2012). – About two times faster than GSP at lower ζ (Zaki , 2001).
Excellent runtime with S-step/I-step traversal, and S-step/I-step pruning
Have A small search space



ern-
mini
ng
algo
rith
ms.

year	Algorithm	theorical Basis
1996	GSP (Agrawal, Srikant, 1996)	$\mathcal{C}_K=\mathcal{L}_{K-1}\bowtie\mathcal{L}_{K-1}$(Combine sequences in $\mathcal{L}_{k-1}$ with itself) that two $\mathcal{L}_{K-1}$ have contiguous sub-sequences
2000	WAP-MINE (Pei, 2000)	Construct WAP-Tree for sequences and conditional search on it
2001	SPADE (Zaki, 2001)	Vertical DB (ID-List) and lattice structure
2002	SPAM (Ayrez et al, 2002)	Depth-first search and vertical bitmap representation
2001	Prefixspan (Pei et al, 2001])	Construct Projected-DB for Prefix and mine it
2001	UNI-SEQ (Pinto, 2001)	Use MD-extension of sequence
2003	Clospan (VAN et al, 2003)	Uses a set of pruning rules
2007	BIDE+ (Wang, Han, 2007)	Use bi-directional search
2011	FSGP (Yi et al, 2011)	Use safe pruning strategy
2013	Clasp (Gomariz et al, 2013)	Produce FCC and post-pruning
2013	MaxSP (Philippe Fournier-Viger et al, 2013)	Maximal-forward extension and maximal-backward extension
2014	CM-SPADE, CM-SPAM and CM-clasp (Philippe Fournier-Viger et. al, 2014)	CMAPs and CMAPi
2014	VMSP (Philippe Fournier-Viger et al, 2014)	– Use EFN, FME and CPC strategy – Vertical DB
2014	VGEN (Philippe Fournier-Viger et al, 2014)	– Use ENG, BEC and CPC strategy – Vertical DB

Table 10. Comparative analysis of algorithm performance

Algorithm	Dataset size	Minimum suporrt (ζ)	Execution time (ms)	Memory usage (mb)
BIED+	Medium (250k)	Low (10%)	24882	504.56
		Medium (30%)	780	523.47
	Large (1M)	Low (10%)	42494	350.51
		Medium (30%)	7988	842.96
Clasp	Medium (250k)	Low (10%)	2731	156.23
		Medium (30%)	796	67.56
	Large (1M)	Low (10%)	6115	304.74
		Medium (30%)	1202	266.56
Clospan	Medium (250k)	Low (10%)	2418	156.23
		Medium (30%)	468	67.56
	Large (1M)	Low (10%)	28018	401.63
		Medium (30%)	1810	123.56
Prefixspan	Medium (250k)	Low (10%)	13806	552.168
		Medium (30%)	686	107.99
	Large (1M)	Low (10%)	22698	638.49

		Medium (30%)	1810	172.26
FSGP	Medium (250k)	Low (10%)	5943	917.86
		Medium (30%)	1654	211.96
	Large (1M)	Low (10%)	14820	505.35
		Medium (30%)	717	107.02
VGEN	Medium (250k)	Low (10%)	1607	154.29
		Medium (30%)	483	756.436
	Large (1M)	Low (10%)	1654	215.41
		Medium (30%)	256	209.95

6. Taxonomy with performance analysis

This section presented the proposed taxonomy, illustrated in table 9 that summarizes the supporting theory and Advantages and Disadvantages which each this paper algorithms is based on. While table 10 shows a comparative performance analysis of algorithms from each of the taxonomy categories. Experimentation was performed on a 1.6 GHz Intel Core i5-4200U with 6 gigabytes of memory, running Windows seven 64-bit, with the same implementations of the programs on a real data set of sign language utterance. This is about 250 kilobyte and it has 500 sequences (as medium data set). Also the algorithms tested on another data set from the novel Leviathan that Thomas Hobbes converts it as SDB. Each word in it is transformed to an item. This is about 1 megabyte and it has 5834 sequences (as large size)[1]. These were run at different minimum support values: the low minimum support is 10% and medium minimum supports of 30% and is showed in table 10. CPU execution time is reported by the program of each algorithm in millisecond, while physical memory usage was in Megabytes (MB) by SPMF software[2]. Careful investigation of table 10 shows how the apriori-based Clasp algorithm could become as data set size grows from medium (| D |=250K) to large (| D |=1M), due to traversal of the large lexicographical tree; although it is a little faster than PrefixSpan (Pei et al, 2004) on medium and large data sets due to the utilization of bitmaps as compared to the projected databases of PrefixSpan. VGEN - Except in medium size and with medium ζ -enjoys the fastest execution times, as it clearly separates itself from clospan and PrefixSpan (from the same category of algorithms), especially at large minimum support values when more frequent patterns are found and with large data sets. In medium size of dataset and medium ζ, Clasp Has the lowest memory consumption that It may for the ClaSP pruning rules.

7. Conclusion

This article presents sequential pattern mining algorithms within 1996 to 2014, and Categories them based on the number of frequent sequences that are mined. Elementary algorithms, extracted all of frequent sequence patterns that many of them may not be the user's requirements. So other algorithms were developed to mine subset of frequent sequence patterns. A thorough discussion of their characteristic features of the four categories of algorithms, with advantage/disadvantage of the different methods and techniques, is presented. It is also noted that literature has recently introduced ways to minimize support counting, although some scholars claim they can avoid it (Wang and han, 2007), (CHIU et al, 2004). Minimizing support counting is strongly related to minimizing the search space. Due of the above, the following requirements should be considered for a reliable sequential pattern-mining algorithm. First, a method must generate a search space that is as small as possible. Features that allow this include early candidate sequence pruning with position induction and search space partitioning. Sampling of the dataset and lossy compression (i.e., concise representation) can

[1] http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php
[2]SPMF is available at http://www.philippe-fournier-viger.com/spmf/index.php?link=algorithms.php

also be used to generate a smaller search space. Second, it is important to narrow the search process within the search space. An algorithm can have a narrow search procedure such as depth-first search. Third, methods other than tree projection should be investigated for finding reliable sequential pattern-mining techniques. Due to these issues, this paper selected the mining sequence pattern algorithms and reviewed performance of them and offered Taxonomy of them.

Acknowledgement The author would like to thank Philippe Fournier-Viger (SPMF Project leader), for his help with the experimental run of the algorithms, and the anonymous reviewers for their productive criticism in SPMF forum.

Reference

Agrawal, R. and Srikant, R. (19995). Mining Sequential Patterns: Generalizations and Per-formance Improvements. Research Report RJ 9994, IBM Almaden Research Center,San Jose, California.

Ayres, j. and et al. (2002). Sequential PAttern Mining using A Bitmap Representation. ACM. 1-7.

Chandra, V. and Sammulal, P. (2013). Survey on Sequential Pattern Mining Algorithms. International Journal of Computer Applications, 24-31.

Cheng, H. and et al. (2005). Seqindex: indexing sequences by sequential pattern analysis. Proceeding of the 2005 SIAM international conference on data mining (SDM'05), 601–605.

CHIU, D. and et al. (2004). an efficient algorithm for mining frequent sequences by a new strategy without support counting. In Proceedings of the 20th International Conference on Data Engineering. 375–386.

Cong, S. and et al. (2005). Parallel mining of closed sequential patterns. Proceeding of the 2005 ACMSIGKDD international conference on knowledge discovery in databases (KDD'05), 562–567.

EZEIFE, C. and et al. (2005). PLWAP sequential mining: Open source code. In Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementation (SIGKDD), ACM, New York, 26–35.

Garofalakis, and et al. (1999). SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 223-234.

Gomariz, A. and et al. (2013). ClaSP: An Efficient Algorithm for MiningFrequent Closed Sequences. Springer-Verlag Berlin Heidelberg. 50-61.

Han, J. and et al. (2002). Mining T op-K Frequent Closed P atterns without Minimum Support. ICDM'02.

Khan, I. and Jain, A. (2012). A Comprehensive Survey on Sequential Pattern Mining. International Journal of Engineering Research & Technology (IJERT), 1-6.

Nizar, R. and Ezeife, I. (2010). A Taxonomy of Sequential Pattern Mining Algorithms. ACM Computing Surveys. 1-41.

Pei, J. and et al. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proceedings of the International Conference on Data Engineering. 215–224.

Pei, J. and et al. (2000). Mining Access Patterns Efficiently from WebLogs*. Springer-Verlag Berlin Heidelberg, 396-407.

Pei J. and et al. (2000). Mining freq. patt.without candidate Generation. in Proceeding of ACM SIGMO International Conference Management of Data, 1-12.

Pei, J. and et al. (2001). CLOSET: An efficient algorithm for mining fr equent closed itemsets . DMKD'01 workshop.

Pei, J. and et al. (2002). Mining Sequential Patterns with Constraints in Large Databases. ACM, 4-9.

Pei, J. and et al. (2002). Mining Sequential Patterns with Constraints in Large Databases. CIKM'02, 18-25.

Pei, J. and Han, J. (2004). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE transactions on knowledge and data engineering, 10, 1-16

Pei, J. and Han, J. (2002). Constrainted sequence pattern mining: a pattern growth view*. SLGKDD Exploraition, 4, 1-9.

Philippe Fournier-Viger, p. and et al. (2014). Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information. Springer International Publishing Switzerland. 40-52.

Philippe Fournier-Viger, p. and et al. (2013). Mining Maximal Sequential Patterns without Candidate Maintenance. Springer.

Philippe Fournier-Viger, p. and et al. (2014). VMSP: E_cient Vertical Mining of Maximal Sequential Patterns. Springer.

Philippe Fournier-Viger, p. and et al. (2014). VGEN: Fast Vertical Mining of Sequential Generator Patterns. Springer.

Pinto, H. and et al. (2001). Multidimensional Sequential Pattern Mining. natural sciences and engineering research.

Ramakrishnan, S. and Agrawal, R. (1996). Mining sequential patterns: generalizations and performance improvements. Springer Berlin Heidelberg. 1-15.

Van, X. and Afshar, R. (2003). CloSpan: Mining Closed Sequential Patterns in Large Datasets*. National Science Foundation NSF. 1-10.

Wang, H. et al. (2006). Design and Implementation of a Web Usage Mining Model Based On Upgrowth and Preflxspan. Communications of the IIMA. 69-84.

Wang, J. and Han, J.(2007). BIDE: Efficient mining of frequent closed sequences. National Science Foundation under Gran. 79–90.

YI, L. (2005). Mining Web Log Sequential Patterns with Position Coded Pre-Order Linked WAP-Tree*. Springer Science Business Media, Inc. Manufactured in The Netherlands. 5-38.

Yi, s. and et al. (2011). An effective algorithm for mining sequential generators. Sciencedirect. Pcreodciead Eian Einngeienreinerging 15. 3653 – 3657.

Yun, u. (2008). A new framework for detecting weighted sequential patterns in large sequence databases. Sciencedirect. 110-122.

Zaki, M. (2001). SPADE: An efficient algorithm for mining frequent sequences. Kluwer Academic Publishers. Manufactured in The Netherlands. 31-60.

Zhang, M. and et al. (2001). a GSP-based eficient algorithm for mining frequent sequences. In Proc. 2001International Conference on Artificial Intelligence (IC-AI 2001).

Zaki, M. and Hsiao, C. (2002). CHARM: An efficient algo-rithm for closed itemset mining. SDM'02.

Yang, J. and Wang, W.(2003). CLUSEQ: efficient and effective sequence clustering. Proceeding of the 2003 international conference on data engineering (ICDE'03), 101–112.

Zhao, Q. and Bhowmick, R. (2003). Sequential Pattern Mining: A Survey. Technical Report, CAIS, Nanyang Technological University, Singapore. (2003118), 1-27.