Pruning Discovered Sequential Patterns Using Minimum Improvement Threshold

Stanislaw Prinke, Marek Wojciechowski, Maciej Zakrzewicz

Poznan University of Technology
Institute of Computing Science
ul. Berdychowo, 60-965 Poznan, Poland
{marek,mzakrz}@cs.put.poznan.pl

Abstract. Discovery of sequential patterns is an important data mining problem with numerous applications. Sequential patterns are subsequences frequently occurring in a database of sequences of sets of items. In a basic scenario, the goal of sequential pattern mining is discovery of all patterns whose frequency exceeds a user-specified frequency threshold. The problem with such an approach is a huge number of sequential patterns which are likely to be returned for reasonable frequency thresholds. One possible solution to this problem is excluding the patterns which do not provide significantly more information than some other patterns in the result set. Two approaches falling into that category have been studied in the context of sequential patterns: discovery of maximal patterns and closed patterns. Unfortunately, the set of maximal patterns may not contain many important patterns with high frequency, and discovery of closed patterns may not reduce the number of resulting patterns for sparse datasets. Therefore, in this paper we propose and experimentally evaluate the minimum improvement criterion to be used in the post-processing phase to reduce the number of sequential patterns returned to the user. Our method is an adaptation of one of the methods previously proposed for association rules.

1 Introduction

Sequential pattern mining [3] is an important data mining problem with numerous applications including analysis of retail data, data registered during scientific experiments, Web server logs, etc. Informally, sequential patterns are subsequences frequently occurring in a database of sequences of sets of items. The most common frequency measure is support, expressed as a number or percentage of data sequences containing a given pattern. In a basic scenario, the goal of sequential pattern mining is discovery of all patterns whose support exceeds a user-specified minimum support threshold. Several algorithms have been proposed for this task, e.g., AprioriAll [3], GSP [11], and PrefixSpan [9]. An obvious problem with such an approach is a huge number of sequential patterns which are likely to be returned as mining results, especially for low support thresholds. Unfortunately, in order to be able to discover non-trivial, interesting, or even surprising patterns, typically a user will have to choose a minimum support value that will lead to a large collection of patterns difficult to comprehend.

In general, there are two strategies to cope with the above problem. One is to allow a user to specify constraints on discovered patterns referring to their structure, e.g., the presence of certain items [5][7][13]. However, relying on constraint-based mining as a means of reducing the size of the mining result has two disadvantages. Firstly, a user may not have any requirements on the pattern structure. Secondly, as pointed out in [8], mining with constraints reduces the possibility of reusing the mining results by other users.

The second strategy to reduce the number of frequent patterns presented to the user is excluding the patterns which do not provide significantly more information than some other patterns in the result set. Two approaches falling into that category have been studied in the context of sequential patterns: discovery of maximal sequential patterns [3] and closed sequential patterns [14]. Maximal sequential patterns are frequent patterns that are not a subsequence of any other sequential pattern. Pruning non-maximal patterns was proposed as a post-processing step already together with the first sequential pattern mining algorithm AprioriAll in [3]. Unfortunately, it was immediately observed that the set of maximal patterns may not contain many important patterns having high support. As a result, some patterns that could have an impact on decision making might not be presented to the user.

A clearly better solution is discovery of closed sequential patterns. A sequential pattern is a closed sequential pattern if it is not a subsequence of any other sequential pattern having exactly the same support. Closed patterns have the following advantages: (1) they form a so-called condensed representation of all frequent patterns, i.e., all frequent patterns with their supports can be generated if necessary from closed patterns, so only closed patterns have to be stored; (2) they can be generated directly from the database, which is typically faster than mining all frequent patterns, and then pruning the set of discovered patterns in a post-processing phase. Nevertheless, the level of reduction of the number of returned patterns thanks to mining only closed patterns depends strongly on the nature of the dataset and may not be noticeable for sparse datasets.

To address the limitations of previously proposed methods, in this paper we propose and experimentally evaluate the minimum improvement criterion to be used in the post-processing phase to reduce the number of sequential patterns returned to the user. The idea is to require that for a sequential pattern to be included in the result set, its support should be greater than the support of all its super-sequences present in the result set by more than the user-specified minimum improvement threshold. For extreme minimum improvement threshold values the method results in mining closed and maximal sequential patterns. Our method is an adaptation of one of the methods previously proposed for association rules, in the context of which the problem of reducing the size of the generated pattern set has been studied much more intensively.

The paper is organized as follows. Section 2 describes related work. In Section 3 we review basic definitions regarding sequential pattern mining, including maximal and closed sequential patterns, and the definitions regarding association rules which are relevant for our discussion. In Section 4 we introduce our improvement measure for sequential patterns, and present the pruning algorithm exploiting the criterion. Section 5 presents experimental results regarding the effectiveness and performance of the proposed method. Section 6 contains concluding remarks.

2 Related Work

To the best of our knowledge, only two methods of reducing the number of discovered sequential patterns by pruning the patterns that do not provide significantly more information than some other patterns have been proposed so far. These methods are discovery of maximal sequential patterns [3] and closed sequential patterns [14].

Much more work on reducing the size of data mining results has been done in the context of the related problem of discovery of frequent itemsets and association rules [1]. In [12] the authors introduced the concept of a rule cover for association rules. The idea was to discover the set of rules covering all the data in the database and prune the remaining rules. As this method loses the completeness of association rule mining, it was not studied further in other works.

Another solution was proposed in [4], where pruning was performed according to the value of the minimum improvement threshold specified by a user. Improvement provided by a rule was defined as the minimum difference between its confidence and the confidence of any of its proper sub-rules, where a proper sub-rule is a simplification of the rule formed by removing one or more items from its antecedent.

Recently, two interesting approaches aiming at reduction of the size of the collection of discovered frequent patterns have been proposed as generalizations of the concept of closed frequent itemsets [6]. [10] proposed g-closed itemsets that could be used to derive all frequent itemsets and their supports within the error equal to a user-specified tolerance factor. The number of g-closed itemsets is typically significantly smaller than the number of closed itemsets, and for the tolerance of zero the output is exactly the set of frequent closed itemsets. G-closed itemsets can be discovered in a post-processing phase or directly from the database using one of the two algorithms proposed by the authors.

In [8] it was observed that very often users do not need exact support values of discovered patterns and they would be satisfied with approximations. The paper introduced the concept of condensed frequent pattern bases, which are collections of patterns that can be used to generate all frequent patterns with a guaranteed error on their support. The major motivation was reduction of the number of mined, stored, and analyzed patterns. The paper also proposed efficient algorithms for mining condensed frequent pattern bases offering a satisfactory compression ratio directly from the database. In two particular cases the proposed approach reduces to mining maximal and closed patterns respectively.

A study of previous research on frequent patterns clearly shows that novel solutions are typically introduced in the context of frequent itemsets or association rules, and then adapted for sequential patterns. As for the techniques of pruning the patterns that do not provide significantly more information than some other patterns, the research on sequential patterns is definitely a few steps behind frequent itemsets and association rules. This paper aims at lessening this gap by redefining the improvement measure from [4] in the context of sequential patterns, and verifying its usefulness for filtering uninteresting sequential patterns.

3 Basic Definitions

3.1 Sequential Patterns

Let $L = \{l_1, l_2, ..., l_m\}$ be a set of literals called *items*. An *itemset* is a non-empty set of items. A *sequence* is an ordered list of itemsets. A sequence s is denoted as $<S_1\ S_2\ ...\ S_n>$, where S_i is an itemset ($S_i \subseteq L$). S_i is called an *element* of the sequence and denoted as $(x_1\ x_2\ ...\ x_m)$, where x_k is an item. The *size* of a sequence is the number of items in the sequence. The *length* of a sequence is the number of elements in the sequence.

A sequence $\alpha = <A_1\ A_2\ ...\ A_n>$ is called a *subsequence* of another sequence $\beta = <B_1\ B_2\ ...\ B_m>$ (and β a *super-sequence* of α), denoted as $\alpha \sqsubseteq \beta$, if there exist integers $i_1 < i_2 < ... < i_n$ such that $A_1 \subseteq B_{i1}$, $A_2 \subseteq B_{i2}$, ..., $A_n \subseteq B_{in}$.

A *sequence database* D is a set of tuples [sid, s], where sid is a sequence identifier and s is a sequence. We say that a tuple [sid, s] *contains* a sequence α if α is a subsequence of s ($\alpha \sqsubseteq s$). The *support* of a sequence α in a sequence database D (denoted as $sup(\alpha)$) is the number of tuples in D that contain α. A sequence α is called a (*frequent*) *sequential pattern* in a sequence database D if its support in D is above the user-specified threshold *minsup*.

A sequential pattern α is *maximal* if there exists no sequential pattern β ($\beta \neq \alpha$) such that α is a subsequence of β.

A sequential pattern α is *closed* if there exists no sequential pattern β ($\beta \neq \alpha$) such that α is a subsequence of β and $sup(\beta) = sup(\alpha)$.

3.2 Association Rules

Let $L = \{l_1, l_2, ..., l_m\}$ be a set of literals called *items*. An *itemset* is a non-empty set of items. An *association rule* is an expression of the form $X \rightarrow Y$, where X and Y are itemsets ($X \subset L$, $Y \subset L$) such that $X \cap Y = \emptyset$. X is called an *antecedent* and Y a *consequent* of the rule $X \rightarrow Y$.

A transaction database TD is a set of tuples [tid, T], where tid is a transaction identifier and T is an itemset ($T \subseteq L$). We say that a tuple [tid, T] *contains* an itemset X if $X \subseteq T$. The *support* of an itemset X in a transaction database TD (denoted as $sup(X)$) is the number of tuples in TD that contain X.

The *support* of an association rule $X \rightarrow Y$ in a transaction database TD (denoted as $sup(X \rightarrow Y)$) is the support of $X \cup Y$. The *confidence* of a rule $X \rightarrow Y$ (denoted as $conf(X \rightarrow Y)$) is defined as $sup(A \cup B)$ / $sup(A)$. The *improvement* of a rule $X \rightarrow Y$ (denoted as $imp(X \rightarrow Y)$) is defined as $min(\forall X' \subset X, conf(X \rightarrow Y) - conf(X' \rightarrow Y))$.

4 Pruning Discovered Sequential Patterns According to the Minimum Improvement Threshold

In this section we formally define the improvement measure for sequential patterns as an adaptation of the measure proposed in [4] for association rules, and then we propose an algorithm that can be used to filter uninteresting patterns from the set of discovered patterns. Our approach aims at providing a post-processing mechanism that will allow a user to interactively adjust the number of presented patterns by hiding the patterns that do not carry significantly more information than some other patterns.

4.1 Improvement Measure for Sequential Patterns

The definition of the improvement measure for association rules is not directly applicable to sequential patterns as it refers to confidence of rules, which is not defined for sequential patterns. However, we claim that improvement can be redefined for sequential patterns in a way preserving its general idea, i.e., capturing the difference in some pattern interestingness measure between a pattern and its sub- and super-patterns.

As the support is the most important and typically the only evaluated measure of sequential patterns' interestingness, we define our improvement measure for sequential patterns in terms of differences in pattern support:

$$imp(\alpha) = min(\forall \alpha' \mid \alpha \sqsubseteq \alpha' \wedge \alpha \neq \alpha', sup(\alpha) - sup(\alpha'))$$

The above formula says that the improvement provided by a given sequential pattern is the minimum difference between its support and the support of any proper super-sequence of the pattern. According to the definition, a high value of the improvement measure means that adding any items to the pattern would result in significant decrease in support.

Example 1. Let us consider a sequential pattern $\alpha = \langle(3)(4\ 5)\rangle$ having the support $sup(\alpha) = 1000$. Let us assume that its only proper frequent super-sequences are $\beta = \langle(3\ 6)(4\ 5)\rangle$ and $\gamma = \langle(3)(4\ 5)(7)\rangle$ with the following supports: $sup(\beta) = 900$, $sup(\gamma) = 950$. According to our definition, in this case: $imp(\alpha) = 50$.

4.2 Pruning Algorithm

The improvement measure can be used to filter the collection of discovered sequential patterns. The assumption is that a pattern being a subsequence of another sequential pattern is interesting only if its support is significantly higher than the support of that second pattern.

Obviously, improvement should be evaluated and tested only for non-maximal patterns for the following two reasons. Firstly, to calculate the pattern improvement we need to know the support of its most frequent super-sequence, and for maximal patterns this will not be provided in the set of frequent patterns. Secondly, maximal patterns should be presented to the user regardless of their improvement values because their super-sequences cannot be considered more interesting since they are not even frequent.

The minimum improvement threshold (*minimp*) for non-maximal patterns should be a parameter set by a user[1] and only patterns whose improvement is greater than *minimp* should be returned. Such an approach will not miss many interesting patterns that could be filtered out when mining maximal patterns, and typically should be more selective than mining closed patterns. In fact, for $minimp=0$ the set of presented sequential patterns will be exactly the set of closed sequential patterns, and for $minimp=\infty$ only maximal patterns will be retained.

According to the definition of the improvement measure, for a specified *minimp* threshold, a sequential pattern α will be considered uninteresting if it has at least one frequent super-sequence β such that $sup(\beta) \geq sup(\alpha) - minimp$. To illustrate a possible problem with such a direct application of the pattern improvement measure, let us analyze the following example.

Example 2. Consider three sequential patterns $\alpha = \langle(3)\rangle$, $\beta = \langle(3)(4)\rangle$, and $\gamma = \langle(3)(4)(5)\rangle$ with the following supports: $sup(\alpha) = 1000$, $sup(\beta) = 950$, $sup(\gamma) = 900$. Assume that β is the only proper frequent super-sequence of α and γ is the only proper frequent super-sequence of β. Thus, $imp(\alpha) = 50$, $imp(\beta) = 50$, and $imp(\gamma)$ will not be evaluated as γ is a maximal pattern. If we prune the collection of patterns using $minimp = 60$, only γ will be presented to the user.

Notice that in the above example, β would be removed because of γ and α would be removed because of β. The point is that a user would not see α despite the fact that in the set of presented patterns there were no super-sequences of α with the support greater than or equal to $sup(\alpha) - minimp$. This may or may not be what a user actually expects. To address the above problem, we propose a pruning algorithm that will hide a pattern which does not satisfy the minimum improvement criterion only if at least one of its super-sequences that make it uninteresting is guaranteed to be included in the set of presented patterns.

The pruning algorithm is depicted in Fig. 1. The algorithm takes a set of frequent patterns *FP* and the minimum improvement threshold *minimp* as input, and returns the set of interesting patterns to be presented *PP*. The assumption is that the pattern sets *FP* and *PP* will be partitioned into subsets containing patterns of a given size, denoted as FP_i and PP_i respectively.

The algorithm achieves its goal by analyzing each sequential pattern after all its super-sequences have been tested (line 1). Each of the patterns is compared only with larger patterns that have already been included in the set of patterns to be presented (line 5). The effect of the whole procedure is as if pattern improvement used for filtering was evaluated taking into account only patterns guaranteed to be retained.

[1] Similarly to the minimum support threshold, *minimp* can be expressed as a number or percentage of sequences in the database.

Conceptually, the difference in supports should be compared with *minimp* only if a subsequence relation between the patterns holds. However, since the improvement test is computationally simpler than the subsequence test (linear complexity wrt. pattern length), we propose to first perform the improvement test for all considered pairs of patterns, and the subsequence test only if the improvement indicates the possibility of pruning (line 6).

```
1.   for $i := max(\forall s \mid s \in FP, size(s))$ downto 1 do
2.     $PP_i := FP_i$;
3.     forall $p$ in $FP_i$ do
4.       for $j := i+1$ to $max(\forall s \mid s \in FP, size(s))$ do
5.         forall $q$ in $PP_j$ do
6.           if $sup(p) \le sup(q) + minimp$ and $p \sqsubseteq q$ then
7.             $PP_i := PP_i \setminus \{p\}$
8.           endif
9.         endfor
10.      endfor
11.    endfor
12.  endfor
```

Fig. 1. Pruning algorithm

The algorithm performs two nested loops over the collection of patterns. Thus, its complexity with respect to the number of patterns is $O(n^2)$.

5 Experimental Results

In order to evaluate effectiveness and efficiency of the proposed sequential pattern filtering method, we performed several experiments using two different synthetic datasets generated with GEN [2]. The first dataset (denoted as GEN1) was generated using the following parameter values: number of customers = 1000, average number of transactions per customer = 8, average number of items in transaction = 1, number of different items = 100, number of patterns = 500, average maximal pattern length = 4, number of itemsets = 60, average maximal itemset length = 1. For the second dataset (denoted as GEN2) the following parameter values were used: number of customers = 1000, average number of transactions per customer = 8, average number of items in transaction = 4, number of different items = 100, number of patterns = 500, average maximal pattern length = 8, number of itemsets = 60, average maximal itemset length = 4.

The first generated dataset (GEN1) was a sparse dataset. For the generation of GEN2, the parameters were adjusted in order to generate a more dense dataset, i.e., containing patterns with more items, with small differences in support between patterns and their sub-patterns.

The experiments were conducted on a PC with AMD Athlon 1.5 GHz processor. For sequential pattern discovery we used our own implementation of the GSP algorithm from [11]. Our post-processing pattern filtering procedure was tested on pattern collections stored in main memory.

In the first series of experiments we counted the number of removed sequential patterns by the post-processing procedure for different values of the minimum improvement threshold (*minimp*). The minimum support threshold for sequential pattern discovery was set to 1% in case of GEN1 dataset, and 10% for GEN2 dataset. Figures 2 and 3 show the number of removed patterns as a function of *minimp* for GEN1 and GEN2 datasets respectively (*minimp* is expressed as the percentage of the total number of sequences in the source dataset).

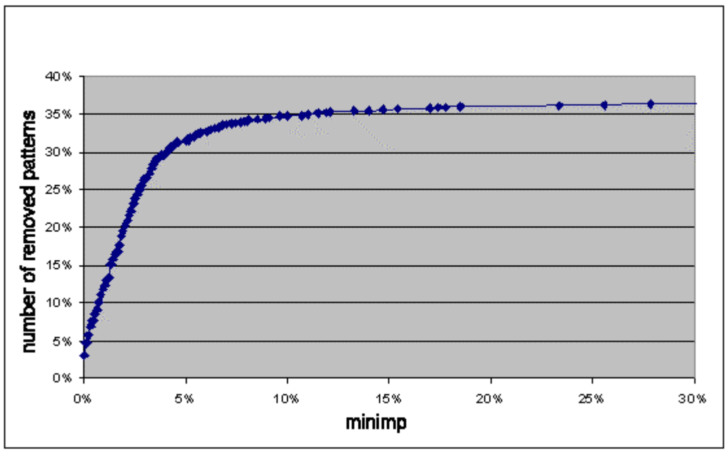

Fig. 2. Number of removed patterns (GEN1 dataset)

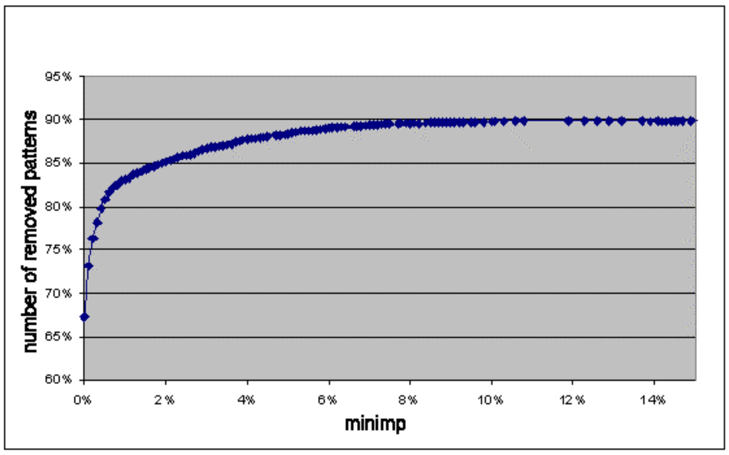

Fig. 3. Number of removed patterns (GEN2 dataset)

The charts show that for both datasets the number of removed patterns changed similarly with the increase of *minimp*, which proves that our method is useful for different kinds of datasets. However, for *minimp* = 0, which corresponds to mining closed sequential patterns, there were only about 3% of patterns removed for GEN1, and as many as 67% for GEN2. This was due to different characteristics of the two datasets. Pruning non-closed patterns is known to result in significant reduction of the pattern set for dense datasets (such as GEN2) but is not satisfactory for sparse datasets (such as GEN1).

Nevertheless, starting with *minimp*=0, even with a slight increase of *minimp* the number of removed patterns grew very rapidly in case for both datasets. Then, at certain point, the number of pruned patterns stabilized and further increase of *minimp* did not increase the number of removed patterns significantly. This was due to the fact that for a certain value of *minimp*, only maximal patterns and a small number of very short patterns with a particularly high support are retained. The difference between a number of closed and maximal sequential patterns was about 30% of the total number of frequent patterns for both datasets, which proves that the space for adjusting the number of presented patterns using *minimp* is large.

The values of *minimp* for which a certain level of pruning was achieved were different for GEN1 and GEN2 datasets. Obviously, the effect of pruning for a given *minimp* threshold depends on the characteristics of a particular sequential pattern collection, which are the consequence of the characteristics of the source dataset and the minimum support threshold chosen for sequential pattern mining.

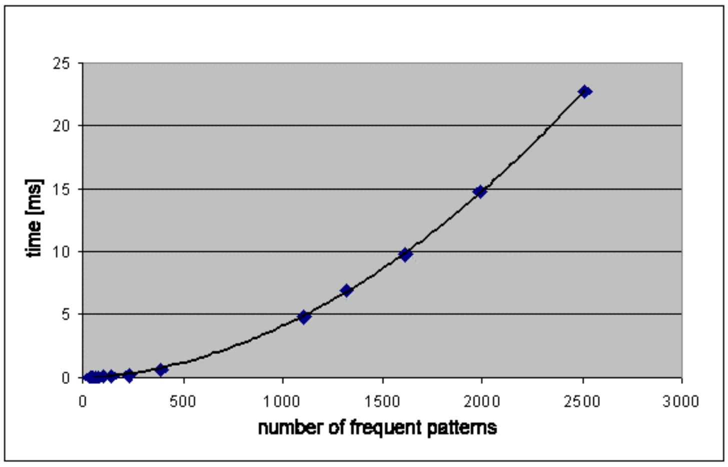

Fig. 4. Execution times (GEN1 dataset)

To evaluate performance of the proposed pattern filtering method, in the second series of experiments we measured the execution time for different number of frequent patterns and various *minimp* thresholds. The collections of sequential patterns were generated from the GEN1 dataset using different minimum support thresholds varying from 0.6% to 10%. Figure 4 presents the execution times of the pruning procedure for different numbers of discovered sequential patterns. Each of the presented execution times is an average over a series of executions for a number of different values of *minimp*.

The chart confirms that processing time of the pruning procedure is proportional to the square of the number of patterns, and shows that even for a few thousands of discovered patterns the procedures completes in a fraction of a second which makes it appropriate for interactive use. Comparing the time needed to filter the patterns according to *minimp* using our algorithm to the time needed to discover the patterns using GSP (not reported here) for dataset GEN1 and minimum support thresholds varying from 0.6% to 10%, we observed that the execution time of our pruning procedure was always less than 1% of the execution time of GSP.

6 Concluding Remarks

In this paper we addressed the problem of reducing the number of discovered sequential patterns presented to the user. The paper has two following contributions. Firstly, we defined a new measure of sequential patterns' interestingness, called improvement, as an adaptation of an analogous measure previously proposed for association rules. Secondly, we discussed possible strategies of using the new improvement measure to prune the collection of discovered sequential patterns, and proposed a post-processing algorithm, which handles selection of closed and maximal sequential patterns as two extreme cases.

The experiments show that the proposed method can significantly reduce the number of sequential patterns, according to user's needs, and offers satisfactory efficiency to be used in interactive data mining environments.

In the paper, we have not considered integration of pruning according to the minimum improvement threshold into sequential pattern mining algorithms. We strongly believe that the proposed pruning criterion is best suited for post-processing, allowing a user to interactively adjust the number of presented patterns from the set of discovered patterns.

References

1. Agrawal R., Imielinski T., Swami A: Mining Association Rules Between Sets of Items in Large Databases. Proc. of the 1993 ACM SIGMOD Conf. on Management of Data (1993)
2. Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T.: The Quest Data Mining System. Proc. of the 2nd Int'l Conference on Knowledge Discovery in Databases and Data Mining (1996)
3. Agrawal R., Srikant R.: Mining Sequential Patterns. Proc. 11th ICDE Conf. (1995)
4. Bayardo R.J., Agrawal R., Gunopulos D.: Constraint-based rule mining in large, dense databases. Proceedings of the 15th International Conference on Data Engineering (1999)
5. Garofalakis M., Rastogi R., Shim K.: SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. Proceedings of 25th VLDB Conference (1999)
6. Pasquier N., Bastide Y., Taouil R., Lakhal L.: Discovering frequent closed itemsets for association rules. Proc. 7th Int'l Conf. On Database Theory (1999)

7. Pei J., Han J., Wang W.: Mining sequential patterns with constraints in large databases. Proceedings of the 11th International Conference on Information and knowledge management (2002)
8. Pei J., Dong G., Zou W., Han J.: Mining Condensed Frequent-Pattern Bases. Proceedings of the IEEE 2002 International Conference on Data Mining (2002)
9. Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U., Hsu M-C.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. Proceedings of the 17th International Conference on Data Engineering (2001)
10. Pudi V., Haritsa J.R.: Generalized Closed Itemsets for Association Rule Mining. Proceedings of the 19th International Conference on Data Engineering (2003)
11. Srikant R., Agrawal R.: Mining Sequential Patterns: Generalizations and Performance Improvements. Proc. of the 5th EDBT Conference (1996)
12. Toivonen H., Klemettinen M., Ronkainen P., Hätönen K., Mannila H.: Pruning and grouping discovered association rules. MLnet Workshop on Statistics, Machine Learning, and Discovery in Databases (1995)
13. Wojciechowski M.: Interactive Constraint-Based Sequential Pattern Mining. Proceedings of the 5th East European Conference on Advances in Databases and Information Systems (2001)
14. Yan X., Han J., Afshar R.: CloSpan: Mining closed sequential patterns in large datasets. Proceedings of SIAM Int. Conf. on Data Mining (2003)