Discovering Frequent Episodes in Sequences of Complex Events[*]

Marek Wojciechowski

Poznan University of Technology
Institute of Computing Science
ul. Piotrowo 3a, 60-965 Poznan, Poland
Marek.Wojciechowski@cs.put.poznan.pl

Abstract. Data collected in many applications have a form of sequences of events. One of the popular data mining problems is discovery of frequently occurring episodes in such sequences. Efficient algorithms discovering all frequent episodes have been proposed for sequences of simple events associated with basic event types. But in many cases events are described by a set of attributes rather than by just one event type attribute. The solutions handling such complex events proposed so far assume that a user provides a template of episodes to be discovered. This assumption does not allow users to discover all surprising relationships between event attributes. In this paper, we propose extensions to algorithms initially designed for simple events making them capable of handling complex events in the same manner.

1 Introduction

Data mining, also called knowledge discovery in databases, consists in efficient discovery of previously unknown and potentially useful knowledge from large sets of data. One of the most important data mining problems is discovery of frequently occurring patterns in event sequences. Application areas for this problem include analysis of telecommunication systems, discovering frequent buying patterns, user interface studies, etc.

The problem of mining frequent episodes was introduced in [8] in the context of analysis of telecommunication networks. The input dataset was a sequence of notifications (alarms) recorded during the operation of the system. The goal was to find all collections of events occurring close to each other frequently enough. Each event in the source data sequence had an associated event type end its occurrence time. There was a set of predefined event types, where each event type could correspond to the certain type of signal sent from a certain module. An episode was considered frequent if its number of occurrences or a percentage of windows of a user-defined size containing the episode was above a user-defined threshold. An episode could provide such information like “event of type *A* is followed by event of

[*] This work was partially supported by the grant no. KBN 43-1309 from the State Committee for Scientific Research (KBN), Poland.

type *B* 30 times in the event sequence” or “events of type *A*, *B*, and *C* occur together in 5% of windows of 60 time units”.

The above formulation was not sufficient since usually events are described by a set of attributes rather than by a single basic event type. For example, alarms coming from telecommunication systems might be described by alarm type, module type and element id. Of course, one could treat concatenation of attribute values as event types and use the approach mentioned above without any modifications. Unfortunately, in that case it would be impossible to discover potentially interesting relationships between subsets of attributes describing events. An example of an episode build from subsets of event attributes in the context of telecommunication network alarms could be: “alarm type *A* in the module M1 is followed by an alarm in the element *#123* 30 times in the event sequence”. Elements of such episodes are not fully specified events, which makes them more general and possibly more frequent and interesting. Discovery of episodes like the one above was possible in the approaches presented in [6] and [9]. The model discussed there considered episodes build from various unary and binary predicates but assumed that a user specifies a template for episodes to be discovered. This limitation made the discovery process similar to querying rather than mining. Moreover, relying on user’s expectations makes discovery of some unexpected patterns impossible.

In this paper, we propose extensions to algorithms initially designed for simple events making them capable of handling complex events in the same manner. In our approach we do not expect from a user any knowledge concerning relationships between events. We extend the algorithms presented in [8] by adding an initial phase consisting in finding all frequently occurring combinations of event attribute values. Episodes considered in our model are build from such frequent combinations of event attribute values, which we call event descriptions.

The paper is organized as follows. Section 2 presents the related work. In Section 3 we introduce several definitions and formalize the problem of mining frequent patterns in sequences of complex events. Section 4 presents the algorithm for the discovery process. In Section 5 we evaluate performance and scalability of our algorithm. Section 6 contains some concluding remarks.

2 Related Work

The problem of mining frequent episodes in event sequences was introduced in [8]. The episodes could have different type of ordering: full (serial episodes), none (parallel episodes) or partial and had to appear within a user-defined time window. They were mined over a single source data sequence and their statistical significance (frequency) was measured as a percentage of windows containing the episode or as a number of occurrences. Each event in a source data sequence had an associated event type and its occurrence time. Efficient algorithms were presented for serial and parallel episodes. The algorithms were based on the Apriori algorithm [2] introduced for association rules discovery [1] and exploited the property that an episode can be frequent if and only if all of its subepisodes are frequent.

In [9], the model was extended to handle events described by a set of attributes. Episodes mined in sequences of such events were build of a set of unary and binary predicates on event attributes. To make discovery of such complex episodes feasible, it was assumed that a user has to specify a class of interesting patterns by providing a template. In [6], a language capable of specifying episodes of interest based on logical predicates was presented and a few further extensions to the model were added.

In [5], an issue of mining sequential relationships between events in a time sequence involving multiple time granularities was addressed. The events forming a source data sequence were occurrences of event types. It was assumed that a user specifies a class of interesting relationships by providing a "rough pattern".

Another approach to the problem of mining frequent patterns in sequential data was presented in [3]. The approach was motivated by the discovery of frequent buying patterns in data collected by companies selling various products. The class of patterns considered there, called *sequential patterns*, had a form of sequences of sets of items. The source database had a form of a set of sequences where each sequence contained sets of items bought in subsequent transactions by one customer. The statistical significance of a pattern (called support) was measured as a percentage of data sequences containing the pattern. In [11], the problem was generalized by adding taxonomies (is-a hierarchies) on items and time constraints such as minimum and maximum gap between adjacent elements of a pattern.

Many ideas implemented in algorithms for discovery of frequent patterns in sequential data were first introduced in the context of association rules [1]. The problem of mining association rules was motivated by market basket analysis. Most approaches to association rules discovery involve discovery of frequently occurring sets of items. The majority of algorithms performing this task are variants of Apriori [2]. For each frequent set Apriori considers all its subsets, which makes Apriori-like algorithms ineffective when discovered patterns are long. In [4] a new algorithm called Max-Miner was proposed for efficient discovery of long patterns from databases.

In [7] and [10], the problem of mining multiple-level association rules was studied. In both cases, the goal was to discover association rules at high concept levels. In [10] concept hierarchies (taxonomies), having a form of trees, were stored as separate objects in the database. In the initial phase of the discovery process, information stored in taxonomies was used to supplement transactions with all ancestors of items present in a given transaction. Then the Apriori algorithm was used. Such an approach was reasonable because the number of ancestors of an item was small. The idea of taxonomies was also applied to the problem of mining sequential patterns mentioned earlier. A slightly different approach to discovery of association rules at different concept levels was presented in [7], where items were described by a set of attributes. Each attribute corresponded to a certain concept level. The algorithm started with the discovery of rules at the highest concept level. Then additional attributes were taken into account and more specific rules were discovered. It was assumed that there is only one optimal order in which attributes can be added to form more specific meaningful descriptions.

3 Problem Statement

Definition 1. Given the set $R = \{A_1, ..., A_m\}$ of event attributes with domains $D_1, ..., D_m$, an *event e* over R is a (m+1)-tuple $(a_1, ..., a_m, t)$, where $a_i \in D_i$ and t is a real number, the occurrence time of e.

Definition 2. An *event sequence* $S = <e_1, ..., e_n>$ is a collection of events over R ordered according to their occurrence times. The total number of events forming an event sequence is called the *length* of the event sequence.

Definition 3. An *event description ed* over R is a set of (attribute, value) pairs $\{(A_{i1}, v_1), ..., (A_{ik}, v_k)\}$, where $A_{ip} \in R$ and for all $p \neq q$ we have $A_{ip} \neq A_{iq}$ and for all $p=1..k$ $v_p \in D_{ip}$. The above definition implies that $|ed| \leq m$. We also require that $|ed| > 0$. ($|ed|$ is called the size of a description). Let ED be the set of all possible event descriptions.

An event description provides information on the nature of an event. Informally, it is a non-empty set of event attribute values, where for each attribute we can have at most one value (some attributes can be omitted). If we take into account values of all the attributes of a given event, we have a full description of its nature. But in some cases, more general descriptions that take into account only a subset of all possible attributes can also be useful. More general descriptions apply to more events, hence they can be used to discover statistically significant relationships (episodes) in the source event sequence.

Definition 4. An event description ed = $\{(A_{i1}, v_1), ..., (A_{ik}, v_k)\}$ describes an event e = $(a_1, ..., a_m, t)$, iff for each p=1..k we have $a_{ip} = v_p$.

Informally, a description describes an event if and only if, for each attribute included in the description, values of the attribute in the description and the event are the same.

Definition 5. An *episode* $\varphi = (V, \leq, g)$ is a set of nodes V, a partial order $\leq$ on V, and a mapping $g : V \rightarrow ED$ associating each node with an event description.

According to the above definition, an episode can be seen as a directed acyclic graph. The interpretation of an episode is that the events described by descriptions associated with the nodes have to occur in the order defined by $\leq$. Our definition of an episode is a variation of the one from [8]. In its original form, nodes are mapped to basic event types. Since in our approach there is more than one attribute associated with an event, we map the nodes to event descriptions.

The most important classes of episodes are parallel and serial episodes. An episode is parallel if the partial order relation $\leq$ is empty (no ordering constraints are specified on elements of the episode). An episode is serial if the partial order relation $\leq$ is a total order. Parallel and serial episodes are important because they are easy to interpret by end users and they can be discovered efficiently from long event sequences. Moreover, any complex partially ordered episode could be seen as a recursive combination of parallel and serial episodes. Thus, recognition of partially ordered episodes can be reduced to recognition of simple parallel and serial episodes.

Definition 6. An episode $\varphi = (V, \leq, g)$ *occurs* in an event sequence $S = <e_1, ..., e_n>$, iff there exists an injective mapping $h : V \rightarrow \{1, ..., n\}$ such that $g(v)$ describes $e_{h(v)}$ for all $v \in V$ and for all $v, w \in V$ with $v \leq w$, $h(v) < h(w)$.

The occurrence of an episode is taken into account only if events corresponding to event descriptions forming the episode occur close enough in time, i.e. within a user-defined time window. An episode is considered interesting if it occurs frequently enough. The measure of statistical significance of episodes is a number of occurrences of an episode in the whole event sequence. (The approach presented in the paper can easily be adapted for other statistical measures like e.g. frequency, defined as a percentage of time windows in which a given episode occurs. We consider here the number of occurrences as a basic measure, because frequency favors episodes occurring in time spans significantly shorter then the window size.)

Problem Formulation. Given an event sequence *S*, a desired class of episodes (parallel or serial), a user-defined window size *win* and a minimal required number of occurrences for an episode to be called frequent *min_fr*, discover all frequent episodes from *S*.

4 Discovery Process

In this section we present a framework of the discovery algorithm for the problem formulated in Section 3. The discovery of frequent episodes in a sequence of complex events requires two steps:
1) discovery of frequently occurring combinations of event attribute values (event descriptions),
2) discovery of frequent episodes build from frequent event descriptions.
For the second step algorithms introduced in [8] are almost directly applicable. However, in our case the algorithms start with a set of frequent descriptions (not event types as in case of simple events) and they use different criteria for checking whether a given episode occurs in the event sequence (checking is done according to Definition 6). Although implementation details are different for different classes of episodes (parallel or serial), the basic ideas underlying the algorithms from [8] are the same. They start with episodes containing one event description and then iteratively generate and verify candidate episodes of larger sizes. In *k*-th iteration candidate episodes of size *k* (containing *k* event descriptions) are generated from frequent episodes of size *k-1*. In each iteration, occurrences of candidate episodes in the event sequence are counted. Candidates that do not occur frequently enough are filtered out. The process stops when no candidates can be generated or no candidates of a certain size are found to be frequent.

Since we do not introduce any other innovations in the second step of the discovery process, we concentrate here on the first step that consists in discovering all frequent combinations of event attribute values (called event descriptions). A straightforward solution for that problem is counting occurrences of all possible combinations of event attribute values in a single database scan. The only problem is that in many cases the number of such combinations may be too large. (There are $(|D_1| + 1) * (|D_2| + 1) * ... * (|D_m| + 1) - 1$ possible combinations of event attribute values. We add *1* to sizes of attribute domains since a given attribute may not be present in a description, we subtract *1* from the resulting product since we require that at least one attribute is specified in a description.)

To address the above limitation we propose an Apriori-like algorithm which requires several passes over source event sequence but does not have to check all possible combinations of event attribute values. The algorithm starts with a set of all candidate descriptions of size one (the set of all event attribute values). In *k*-iteration ($k>1$) candidate descriptions of size *k* are built from frequent descriptions of size *k-1*, and their occurrences in the database are counted. The algorithm stops if, for some *k*, no candidates of size *k* can be generated or no candidates of size *k* are found to be frequent.

Apriori-like solutions may be inefficient when patterns to be discovered are long but in our case the maximal length of a description is limited by a number of attributes, which is not likely to be large. In fact, knowing the upper bound on the size of a description we know the maximal possible number of iterations in this phase of the episode discovery process. In *k*-th iteration of the Apriori-like algorithm, candidate descriptions of size *k* are analyzed. Since the maximal size of a description is the number of attributes *m*, the maximal number of iterations of the Apriori-like algorithm is *m*. Of course, for a given frequency threshold, the actual number of iterations can be less than *m*, if no candidate descriptions of a certain size (less than *m*) turn out to be frequent.

The detailed Apriori-like algorithm for discovery of frequent event descriptions is presented below.

```
$F_1$ = {frequent (attribute, value) pairs in $S$};
for (k=2; $F_{k-1} \neq \emptyset$ and k <= m ; k++) do
begin
  $C_k$ = cand-desc-gen($F_{k-1}$);
  forall events e $\in$ $S$ do
  begin
    $C_d$ = describing($C_k$, e);
    forall candidates c $\in$ $C_d$ do
      c.count++;
  end;
  $F_k$ = { c $\in$ $C_k$ | c.count $\geq$ $min\_fr$};
end;
Answer = $\cup_k$ $F_k$;
```

The function *cand-desc-gen* is given a set of frequent event descriptions having *k-1* (attribute, value) pairs and returns a set of description candidates having *k* (attribute, value) pairs such that for each description candidate its elements concern different attributes and all of its subsets of size *k-1* are frequent.

The function *describing* has two parameters: a set of event descriptions C_k and an event *e*, and returns a subset of C_k containing descriptions describing event *e*.

Candidate generation is done in two phases:
1) generation of candidates by merging pairs of descriptions of size *k-1* having *k-2* (attribute, value) pairs in common and differing in (attribute, value) pairs concerning different attributes,
2) pruning out candidates having at least one subset that is not frequent.
To guarantee uniqueness of candidates generated in the first of the above phases, we keep (attribute, value) pairs forming a description sorted lexicographically according to attribute names, and merge descriptions having the first *k-2* pairs in common and differing in their last pair. This procedure guarantees completeness of the candidate generation process, and is common for all variants of Apriori [2]. The only thing specific to our problem is that we have to guarantee that elements of a candidate concern different attributes.

Let us consider the following example. We are given a short event sequence presented in Table 1. The event sequence has a form of a log containing notifications of possible malfunction of system modules. Three attributes are used to describe the nature of events: module name, type of notification and severity. For each event, its occurrence time is remembered. Let us assume that a user wants to discover serial episodes that occurred at least two times (*min_fr* = 2). In addition, only occurrences spanning over no more than 10 time units are to be taken into account (*win* = 10).

Table 1. Example event sequence

Time	Module name (M)	Notification type (N)	Severity (S)
105	m10	t27	1
108	m20	t54	2
112	m10	t27	3
113	m20	t30	3
119	m30	t54	2
126	m40	t54	3

The process of discovering frequent episodes starts with the discovery of frequently occurring event descriptions. In this phase, only *min_fr* constraint is used. The first iteration of the Apriori-like algorithm finds all frequent (occurring at least *min_fr* times) attribute values. In our case we get: {(M, m10)}, {(M, m20)}, {(N, t27)}, {(N, t54)}, {(S, 2)}, and {(S, 3)}. In the second iteration, candidates of size 2 are generated: {(M, m10), (N, t27)}, {(M, m10), (N, t54)}, {(M, m10), (S, 2)},{(M, m10), (S, 3)},{(M, m20), (N, t27)},{(M, m20), (N, t54)},{(M, m20), (S, 2)},{(M, m20), (S, 3)},{(N, t27), (S, 2)},{(N, t27), (S, 3)},{(N, t54), (S, 2)}, and {(N, t54), (S, 3)}. Next, in the database pass occurrences of generated candidates are counted. Only two of them turn out to be frequent (describing at least 2 events): {(M, m10), (N, t27)} and {(N, t54), (S, 2)}. In the third iteration, the algorithm tries to generate candidates of size 3. The discovery of frequent descriptions ends here, because no candidates of size 3 can be generated. The resulting frequent event descriptions ({(M, m10)}, {(M, m20)}, {(N, t27)}, {(N, t54)}, {(S, 2)}, {(S, 3)}, {(M, m10), (N, t27)} and {(N, t54), (S, 2)}) are then used to discover frequent episodes. An example of a serial episode that can be found in the event sequence from Table 1 is an episode saying that notification *t27* from module *m10* is followed by notification *t54* with severity level of *2* coming from some module.

5 Performance Analysis

To assess the performance of our Apriori-like algorithm for discovery of frequent event descriptions we performed several experiments on synthetic data using a PC with Pentium 133MHz processor and 64MB of main memory. The data resided in a flat file and were generated so that the maximal size of a frequent description to be discovered was always equal to the number of event attributes. Due to the above constraint, our algorithm in each run had to perform the maximal possible number of iterations (we consider the worst case). In the experiments we measured execution times for various number of attributes, attribute domain sizes and database sizes (expressed as a total number of events forming the source event sequence). We also counted the number of candidates that our Apriori-like algorithm had to verify, and compared it to the number of all possible descriptions that can be build from a given number of attributes and their domain sizes.

The goal of one of the experiments was to find out how our algorithm scales with the size of the database. Since the meaning of our frequency measure *min_fr* (the number of occurrences) depends on the length of the event sequence, in the experiments we expressed the required statistical significance of event descriptions as a ratio of *min_fr* to the length of the event sequence. All the experiments were conducted for two frequency thresholds: 2% of the sequence length and 10% of the sequence length. Results applying to runs of our Apriori-like algorithm with these frequency thresholds in the following table and charts are labeled *Apriori2%* and *Apriori10%* respectively.

Table 2. Numbers of candidates analyzed by the Apriori-like algorithm

	Candidates analyzed by Apriori2%	Candidates analyzed by Apriori10%	The total number of possible event descriptions
NATR = 3, NVAL = 10	331	34	1330
NATR = 3, NVAL = 15	721	49	4095
NATR = 3, NVAL = 20	1261	64	9260
NATR = 3, NVAL = 25	1951	79	17575
NATR = 3, NVAL = 30	2791	94	29790
NATR = 4, NVAL = 10	645	51	14640
NATR = 5, NVAL = 10	1066	76	161050
NATR = 6, NVAL = 10	1602	117	1771560
NATR = 7, NVAL = 10	2269	190	19487170

Table 2 presents numbers of candidates analyzed by the Apriori-like algorithm for different numbers of event attributes (denoted as *NATR*) and their domain sizes (denoted as *NVAL*). For simplicity, in all the experiments we considered attributes having the same domain sizes. The number of possible event descriptions grows exponentially with the number of attributes, which leads to huge numbers of possible descriptions even for relatively small sizes of attribute domains. The number of candidates analyzed by the Apriori-like algorithm remains reasonably small (of course, exact numbers depend on the nature of the input database).

Figures 1 and 2 show execution times of the Apriori-like algorithm for different database sizes. Figure 1 shows results for the sequence of events having 3 attributes, Figure 2 - for the sequence of events having 7 attributes. In both cases the attribute domain size for all attributes was 10. The algorithm scales linearly with the database size. As the database size becomes larger, there are more events to read in each database pass but the number of candidates to process remains the same, because it depends on the distribution of attribute values in the database and not on the database size.

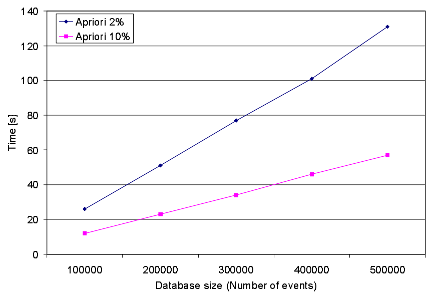

Fig. 1. Execution time for different database sizes (NATR = 3, NVAL = 10)

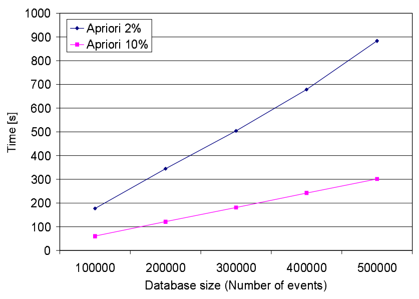

Fig. 2. Execution time for different database sizes (NATR = 7, NVAL = 10)

Figures 3 and 4 show how execution times change with the number of attributes and sizes of their domains for the same database size of 100000 events. The experiments show that the scalability of the algorithm with these two parameters is a bit worse than linear. The lower the frequency threshold, the more rapidly the execution time grows. This is not surprising, because if the frequency threshold is low, in each iteration many candidates are found to be frequent, which in turn leads to a large number of candidates generated in the next iteration.

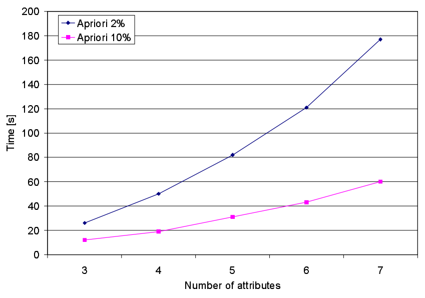

Fig. 3. Execution time for different number of attributes (NVAL = 10)

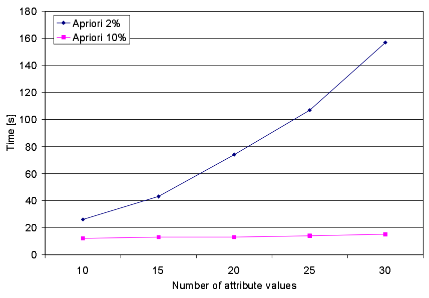

Fig. 4. Execution time for different attribute domain sizes (NATR = 3)

6 Concluding Remarks

We discussed the problem of mining frequent episodes in sequences of events described by sets of attributes. An algorithm for discovery of frequent episodes build from multi-attribute descriptions was presented. The algorithm uses previously proposed methods for episode discovery in sequences of simple events after the initial phase that consists in finding all frequently occurring combinations of event attribute values. Since the straightforward approach to that initial phase by checking occurrences of all possible event descriptions in a single database scan may not be feasible when the number of event attributes and their possible values are large, we proposed an Apriori-like algorithm to address this problem. Experiments show that our Apriori-like algorithm scales linearly with the size of the database and almost linearly with the number of event attributes and sizes of attribute domains.

The approach presented in the paper could be adapted for discovery of sequential patterns, where patterns are mined in a set of data sequences containing sets of items. In that case, items would be described by a set of attributes and patterns would be sequences of elements having the form of sets of item descriptions. The only problem to be solved would be handling situations when within an element of a pattern one item description is a subset of some other item description. One possible solution would be considering descriptions that are subsets of some other item description from the same pattern element as redundant and removing them from the pattern.

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., Srikant R.: Fast Algorithms for Mining Association Rules. Proc. of the 20th Int'l Conf. on Very Large Databases (1994)
3. Agrawal R., Srikant R.: Mining Sequential Patterns. Proc. of the 11th Int'l Conference on Data Engineering (1995)
4. Bayardo R. J.: Efficiently Mining Long Patterns from Databases. Proc. of the 1998 ACM-SIGMOD Conf. on Management of Data (1998)
5. Bettini C., Wang X.S., Jajodia S., Lin J.: Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences. IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 2, March/April 1998 (1998)
6. Guralnik V., Wijesekera D., Srivastava J.: Pattern Directed Mining of Sequence Data. Proc. of the 4th Int'l Conference on Knowledge Discovery and Data Mining (1998)
7. Han J., Fu Y.: Discovery of Multiple-Level Association Rules from Large Databases. Proc. of the 21st Int'l Conf. on Very Large Data Bases, Zurich, Switzerland (1995)
8. Mannila H., Toivonen H., Verkamo A.I.: Discovering frequent episodes in sequences. Proc. of the 1st Int'l Conference on Knowledge Discovery and Data Mining (1995)
9. Mannila H., Toivonen H.: Discovering generalized episodes using minimal occurrences. Proc. of the 2nd Int'l Conference on Knowledge Discovery and Data Mining (1996)
10. Srikant R., Agrawal R.: Mining Generalized Association Rules. Proc. of the 21st Int'l Conf. on Very Large Data Bases, Zurich, Switzerland (1995)
11. Srikant R., Agrawal R.: Mining Sequential Patterns: Generalizations and Performance Improvements. Proc. of the 5th Int'l Conf. on Extending Database Technology (1996)