Mining Multidimensional Sequential Patterns over Data Streams

Chedy Raïssi, Marc Plantevit

Mining Multidimensional Sequential Patterns over Data Streams

Chedy Raïssi and Marc Plantevit

LIRMM, University of Montpellier, France
{raissi,plantevi}@lirmm.fr

Abstract. Sequential pattern mining is an active field in the domain of knowledge discovery and has been widely studied for over a decade by data mining researchers. More and more, with the constant progress in hardware and software technologies, real-world applications like network monitoring systems or sensor grids generate huge amount of streaming data. This new data model, seen as a potentially infinite and unbounded flow, calls for new real-time sequence mining algorithms that can handle large volume of information with minimal scans. However, current sequence mining approaches fail to take into account the inherent multidimensionality of the streams and all algorithms merely mine correlations between events among only one dimension. Therefore, in this paper, we propose to take multidimensional framework into account in order to detect high-level changes like trends. We show that multidimensional sequential pattern mining over data streams can help detecting interesting high-level variations. We demonstrate with empirical results that our approach is able to extract multidimensional sequential patterns with an approximate support guarantee over data streams.

1 Introduction

Sequential patterns have been studied for more than a decade [1], with substantial research and industrial applications. Sequence pattern mining allows the discovery of frequent sequences and helps identifying relations between itemsets in transactional database. However, sequential pattern mining is a difficult and challenging task as the search space for this problem is huge. To bypass this problem, researchers have developed mining algorithms based on the *Apriori* property [1] or *pattern growth* paradigm [10]. Lately, these approaches have been extended to mine multidimensional sequential patterns [11, 12, 14]. They aim at discovering more interesting patterns that take time into account and involve several analysis dimensions. For instance, in [12], rules like "*A customer who bought a surfboard together with a bag in New York later bought a wetsuit in San Francisco*" are discovered.

With the constant evolutions in hardware and software technologies, it becomes very affordable for companies and organisations to generate and store very large volume of information from various sources: network monitoring with TCP/IP traffic, financial transactions such as credit card customers operations,

medical records and a wide variety of sensor logs. This large amount of data calls for the study of a new model called data streams, where the data appears as a continuous, high-speed and unbounded flow. Compared to the classical mining approaches from static transaction databases, the problem of mining sequences over data streams is far more challenging. It is indeed often impossible to mine patterns with classical algorithms requiring multiple scans over a database. Consequently, new approaches were proposed to mine itemsets [4, 5, 7, 8]. But, few works focused on sequential patterns extraction over data streams [2] [9] [13]. In this paper, we propose to consider the intrinsic multidimensionality of the streams for the extraction of more interesting sequential patterns. These patterns help detecting high-level changes like trends or outliers and are by far more advantageous for analysts than low-level abstraction patterns. However, the search space in multidimensional framework is huge. To overcome this difficulty, we only focus on the most specific abstraction level for items instead of mining at all possible levels. Furthermore, we mine time-sensitive data streams using a tilted-time frame approach that is more suitable for pattern mining. As a matter of fact, other models, like the landmark or sliding-window models, are often not appropriate since the set of frequent sequences is time-sensitive and it is often more important to detect changes in the sequences than sequences themselves.

The rest of this paper is organized as follows. Related work is described in Section 2. Preliminary concepts, problem description and a motivating example are introduced in Section 3. Section 4 presents our algorithm. The experiments and their results are described and discussed in Section 5. In the last section we give some conclusions and perspectives for future researches.

2 Related Work

Lately, sequential patterns have been extended to mine multidimensional sequential patterns in [11], [14] and [12]. [11] is the first paper dealing with several dimensions in the framework of sequential patterns. The sequences found by this approach do not contain multiple dimensions since the time dimension only concerns products. In [14], the authors mine for sequential patterns in the framework of Web Usage Mining considering three dimensions (pages, sessions, days), those being very particular since they belong to a single hierarchized dimension.In [12], rules combine several dimensions but also combine over time. In the rule *A customer who bought a surfboard together with a bag in NY later bought a wetsuit in SF*, NY appears before SF, and $surfboard$ appears before $wetsuit$. In [13], the authors propose a new approach, called SPEED (*Sequential Patterns Efficient Extraction in Data streams*), to identify maximal sequential patterns over a data stream. It is the first approach defined for mining sequential patterns in streaming data. The main originality of this mining method is that the authors use a novel data structure to maintain frequent sequential patterns coupled with a fast pruning strategy. At any time, users can issue requests for frequent sequences over an arbitrary time interval. Furthermore, this approach produces an

approximate answer with an assurance that it will not bypass user-defined frequency and temporal thresholds. In [9], the authors propose an algorithm based on sequences alignment for mining approximate sequential patterns in Web usage data streams. In [6] is introduced an efficient stream data cubing algorithm which computes only the layers along a popular path and leaves the other cuboids for query driven, on-line computation. Moreover, a tilted-time window model is also used to construct and maintain the cuboid incrementally.

We can denote that the approach defined in [2] is totally different from our proposal even if some notations seem to be similar. Indeed, the authors propose to handle several data streams. However, they only consider one analysis dimension over each data stream.

3 Problem Definition

In this section, we define the problem of mining multidimensional sequential patterns in a database and over a data stream.

Let DB be a set of tuples defined on a set of n dimensions denoted by $\mathcal{D}$. We consider a partitioning of $\mathcal{D}$ into three sets: the set of the analysis dimensions D_A, the set of reference dimensions D_R and the set of temporal dimensions D_t. A *multidimensional item* $a = (d_1, \ldots, d_m)$ is a tuple such that for every $i = 1 \ldots m$, $d_i \in Dom(D_i) \cup \{*\}$, $D_i \in D_A$. The symbol * stands for wild-card value that can be interpreted by ALL.

A *multidimensional itemset* $i = \{a_1, \ldots, a_k\}$ is a non-empty set of multidimensional items such that for all distinct i, j in $\{1 \ldots k\}$, a_i and a_j are incomparable. A *multidimensional sequence* $s = \langle i_1, \ldots, i_l \rangle$ is an ordered list of multidimensional itemsets. Given a table T, the projection over $D_t \cup D_A$ of the set of all tuples in T having the same restriction r over D_R is called a *block*. Thus, each block $\mathcal{B}_r$ identifies a multidimensional data sequences.

A multidimensional data sequence identified by $\mathcal{B}_r$ *supports* a multidimensional sequence $s = \langle i_1, \ldots, i_l \rangle$ if for every item a_i of every itemset i_j, there exists a tuple (t, a'_i) in $\mathcal{B}_r$ such that $a'_i \subseteq a_i$ with respect to the ordered relation (itemset i_1 must be discovered before itemset i_2, etc.). The support of a sequence s is the number of blocks that support s. Given a user-defined minimum support threshold *minsup*, a sequence is said to be *frequent* if its support is greater than or equal to *minsup*.

Given a set of blocks B_{DB,D_R} on a table DB, the problem of mining *multidimensional sequential patterns* is to discover all multidimensional sequences that have a support greater than or equal to the user specified minimum support threshold *minsup* (denoted σ).

Because of wild card value, multidimensional items can be too general. Such items do not describe the data source very well. In other words, they are too general to be useful and meaningful to enhance the decision making process. Moreover, these items combinatorially increase the search space. In this paper, we thus focus on the *most specific* frequent items to generate the multidimensional sequential patterns. For instance, if items $(LA, *, M, *)$ and $(*, *, M, Wii)$

are frequent, we do not consider the frequent items $(LA, *, *, *)$, $(*, *, M, *)$ and $(*, *, M, Wii)$ which are more general than $(LA, *, M, *)$ and $(*, *, M, Wii)$.

Let *data stream* $DS = B_0, B_1, \ldots, B_n$, be an infinite sequence of batches, where each batch is associated with a timestamp t, i.e. B_t, and n is the identifier of the most recent batch B_n. A batch B_i is defined as a set of multidimensional blocks appearing over the stream at the i^{th} time unit, $B_i = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, ..., \mathcal{B}_k\}$. Furthermore, the data model is fixed for the data stream: all batches are defined over the same set of dimensions D. For each block $\mathcal{B}_k$ in B_i we are thus provided with the corresponding list of itemsets. The length L_{DS} of the data stream is defined as $L_{DS} = |B_0| + |B_1| + \ldots + |B_n|$ where $|B_i|$ stands for the cardinality of the set B_i in terms of multidimensional blocks. In our current example, $L_{DS} = |B_{October}| + |B_{November}| = 7$, with $|B_{October}| = 3$ ($\mathcal{B}_{C_1}$, $\mathcal{B}_{C_2}$ and $\mathcal{B}_{C_3}$) and $|B_{November}| = 4$ ($\mathcal{B}_{C_4}$, $\mathcal{B}_{C_5}$,$\mathcal{B}_{C_6}$ and $\mathcal{B}_{C_7}$).

Therefore, given a user-defined minimal support σ, the problem of mining multidimensional sequential patterns over data stream DS is to extract and update frequent sequences S such that: $support(S) \geq \sigma.L_{DS}$.

However, and in order to respect the completeness of our sequential pattern extraction, any data stream mining algorithm should take into account the evolution of sequential patterns support over time: infrequent sequences at an instant t could become frequent later at $t+1$ and a frequent sequence could also not stay such. If a sequence become frequent over time and we did not store its previous support, it will be impossible to compute its correct overall support. Any mining algorithm should thus store necessary informations for frequent sequences, but also for candidate *sub-frequent* patterns that could become frequent [5]. A sequence S is called *sub-frequent* if: $\epsilon \leq Support(S) \leq \sigma$, where ϵ is a user defined support error threshold.

October					
CID	Date	Customer Informations			
C_1	1	*NY*	*Educ.*	*Middle*	*CD*
C_1	1	*NY*	*Educ.*	*Middle*	*DVD*
C_1	2	*LA*	*Educ*	*Middle*	*CD*
C_2	1	*SF*	*Prof.*	*Middle*	*PS3*
C_2	2	*SF*	*Prof.*	*Middle*	*xbox*
C_3	1	*DC*	*Business*	*Retired*	*PS2*
C_3	1	*LA*	*Business*	*Retired*	*Game*

November					
CID	Date	Customer Informations			
C_4	1	*NY*	*Business*	*Middle*	*Wii*
C_4	2	*NY*	*Business*	*Middle*	*Game*
C_5	1	*LA*	*Prof.*	*Middle*	*Wii*
C_5	1	*LA*	*Prof.*	*Middle*	*iPod*
C_6	1	*LA*	*Educ.*	*Young*	*PSP*
C_6	2	*LA*	*Educ.*	*Young*	*iPod*
C_7	1	*LA*	*Business*	*Young*	*PS2*

Fig. 1. Multidimensional tables

3.1 Motivating Example

In order to illustrate our previous definitions, we focus only on the two month tables (also called batches) in Figure 1: October and November. More precisely, these batches describe the customer purchases according to 6 attributes or dimensions: the *Customers id*, the purchase *Date*, the shipment *City*, the customer social group *Customer-group*, the customer age group *Age-group* and the *Product* as shown in Figure 1. Suppose that a company analyst wants to extract

multidimensional sequences for the discovery of new marketing rules based on customers purchasing evolutions from October to November.

The analysts would like to get all the sequences that are frequent in at least 50% of the customers in each batch. Note that each row for a customer contains a transaction number and a multidimensional item. From the October batch, the analyst extracts three different sequences: (i) The first sequence: $\langle\{(*,*,M,*)\}\rangle$ is a single item sequence. This states that at least 2 customers out of 3 buying entertainment products are middle aged. (ii) The second sequence: $\langle\{(LA,*,*,*)\}$ is also a single item one. The analyst can infer that at least 2 customers out of 3 asked for their purchases to be shipped in Los Angeles. (iii) The third sequence: $\langle\{(*,*,M,*)\}\{(*,*,M,*)\}\rangle$, is a 2-item sequence. This is an interesting knowledge for the analyst as it means that 2 customers out of 3 are middle aged and that they bought entertainment products twice in October. Then the analyst extracts sequences for November. He finds 9 sequences but 3 are really interesting: (i) $\langle\{(LA,*,Y,*)\}\rangle$, this is an extremely valuable new knowledge as it informs the analyst that 2 out of 4 customers that ask to be shipped in Los Angeles are young people. This is also a specialization of the second sequence from October's batch. (ii) $\langle\{(LA,*,*,iPod)\}\rangle$, this informs the analyst that 2 out of 4 customers that have asked to be shipped in Los Angeles bought the iPod product. Thus, the analyst can use this knowledge to build, for instance, targeted customers offers for the next month. Notice that this sequence is also a specialization of the second sequence from the previous batch. (iii) $\langle\{(*,*,M,Wii)\}\rangle$, this sequence infers that 2 out of 4 customers are middle-aged and bought a brand new Wii console in November. This is also a specialization of the first sequence of October's batch. Plus, this sequence highlights the appearence of a new product on the market: the Wii console.

4 The *MDSDS* Approach

In *MDSDS*, the extraction of multidimensional sequential patterns is the most challenging step. In order to optimize it we divide the process into two different steps:

1. *MDSDS* extracts the most specific multidimensional items. Most specific items are a good alternative to the potential huge set of frequent multidimensional items that can be usually extracted. Indeed, they allow to *factorize* knowledge, as more general patterns can be inferred from them in a post-processing step. Furthermore, mining most specific multidimensional items allows the detection of generalization or specialization of items with wildcard values appearing or disappearing in an item over time.
2. Using the extracted most specific items, we mine sequences containing *only* these items in a classical fashion using PrefixSpan algorithm[10]

When applying this strategy, frequent sequences with too general items are not mined if there exist some more specific ones. However, this is not a disadvantage since these sequences often represent too general knowledge which is useless

and uninteresting for the analysts (*e.g.* decision maker). $MDSDS$ uses a data structure consisting of a prefix-tree containing items and tilted-time windows tables embedded in each node of the tree to ensure the maintenance of support information for frequent and sub-frequent sequences. We use tilted-time windows table technique in our approach to store sequences supports for every processed batch. Tilted-time windows notion was first introduced in [3] and is based on the fact that people are often interested in recent changes at a fine granularity but long term changes at a coarse one. By matching a tilted-time window for each mined sequence of the stream, we build a history of the support of the sequence over time. The patterns in our approach are divided into three categories: frequent patterns, sub-frequent patterns and infrequent patterns. Sub-frequent patterns may become frequent later (in the next batches from the data stream), thus $MDSDS$ has to store and maintain their support count as for the frequent ones. Only the infrequent patterns are not stored in the prefix-tree. The cardinality of the set of sub-frequent patterns that are maintained is decided by the user and called *support error threshold*, denoted ϵ. The updating operations are done after receiving a batch from the data stream: at the prefix-tree level (adding new items and pruning) and at the tilted-time window table level (updating support values). First, multidimensional sequential patterns are extracted from the batch and projected in the prefix-tree structure. Second, the tilted-time windows table for each multidimensional sequential pattern is updated by the support of the sequence in the current batch. The pruning techniques relative to tilted-time windows table are then applied. Generalization and specialization of sequences are detected during the maintenance.

4.1 Algorithm

We now describe in more details the $MDSDS$ algorithm. This algorithm is divided in four steps :

1. As previously highlighted, mining most specific multidimensional items is the starting point for our multidimensional sequence extraction. We use a levelwise algorithm to build the frequent multidimensional items having the smallest possible number of wild-card values. Each item on the different analysis dimensions is associated with a unique integer value. We then mine the most specific multidimensional items based on this new mapping.

   For instance, let us consider October's batch from Figure 1. Each value for each analysis dimension will be mapped to an integer value: NY in the $City$ dimension will be mapped to 1, LA to 2 and so on until $Game$ in $Product$ dimension which will be mapped to the value 15. With this new representation of the batch, the most specific frequent items extracted are: (2) and (8), in multidimensional representation: $(LA, *, *, *)$ and $(*, *, M, *)$. All the multidimensional items are stored in a data structure called mapping array. After this step, we can add an optional processing step in order to detect the appearance of specialization or generalization according to the previous processed batch. Indeed, a specific item which was frequent over the previous batches could become infrequent later. In this case, we consider some

more general multidimensional items. In order to detect the specialization or generalization, the subroutine compare (by inclusion) each multidimensional item from the current batch with the multidimensional items in the previous batch. Notice that in order to keep the process fast, the items are compared only with the previous batch.

2. In order to have a consistant mining of frequent sequences we have to consider subfrequent sequences which may become frequent in future batches. Thus, the support is set to ϵ and we use the PrefixSpan algorithm [10] to mine efficiently the multidimensional sequences. The new mined sequences are added to the tree which maintain the set of frequent and subfrequent sequences over the data stream.
3. Finally, a last scan into the pattern tree is done in order to check if each node n was updated when batch B_i was mined. If not, we insert 0 into n's tilted-time window table. Eventually, some pruning is done on the tilted-time window as defined in [5].

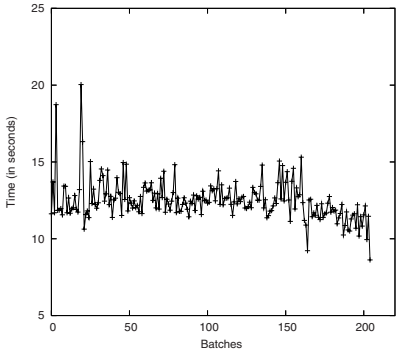
5 Performance

In this section, we present the experiments we conducted in order to evaluate the feasability and the performances of the *MDSDS* approach. Throughout the experiments, we answer the following questions inherent to scalability issues : *Does the algorithm mine and update its data structure before the arrival of the next batch? Does the mining process over a data stream remain bounded in term of memory ?* The experiments were performed on a Core-Duo 2.16 Ghz MacBook Pro with 1GB of main memory, running Mac OS X 10.5.1. The algorithm was written in C++ using a modified Apriori code[1] to mine the most specific multidimensional items and we modified PrefixSpan[2] implementation to enable multidimensional sequence mining. We performed several tests with different real data sets that we gathered from TCP/IP network traffic at the University of Montpellier. The size of the batches is limited to 20000 transactions with an average of 5369 sequences per batch, the time to fill the batch varies w.r.t to the data distribution over the TCP/IP network.

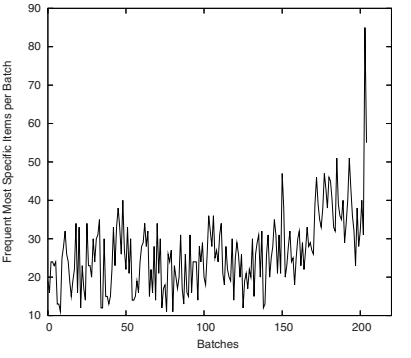
A TCP/IP network can be seen as a very dense multidimensional stream. This is based on the following property: TCP and IP headers contain multiple different informations encapsulated in different formats that can vary drastically depending on the packet destination, its final application or the data it contains. We claim that generalization or specialization of the packets on the network (or even Denial of Services) can be detected by mining multidimensional sequential patterns over a local area network.

*DS*1 is very dense data set composed of 13 analysis dimensions based on the different TCP header options selected for their relevance by a network analyst expert (source port, destination port, time-to-live etc...), the data stream is divided into 204 batches for a total size of 1.58 GB. The results for the different

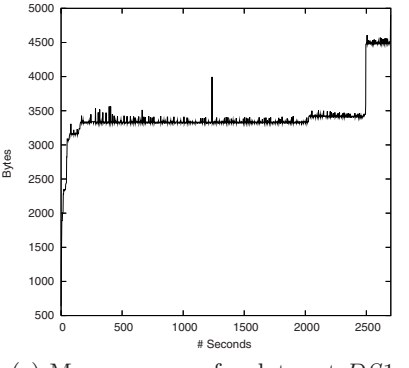
[1] http://www.adrem.ua.ac.be/ goethals/software/
[2] http://illimine.cs.uiuc.edu/



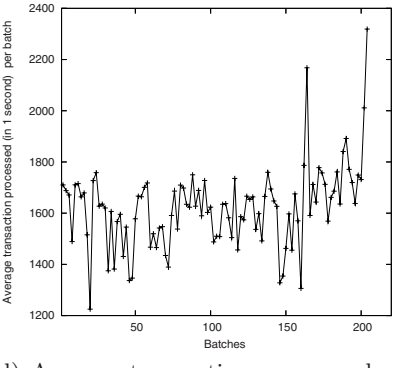
(a) Time needed for each batch processing for data set $DS1$



(b) Number of frequent most specific items per batch for data set $DS1$

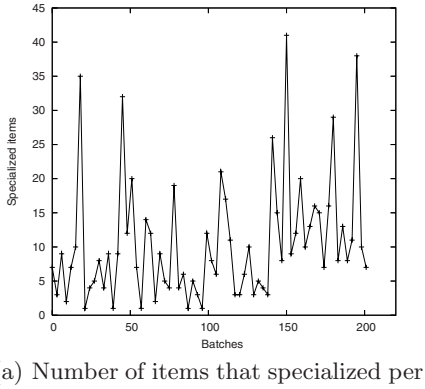


(c) Memory usage for data set $DS1$

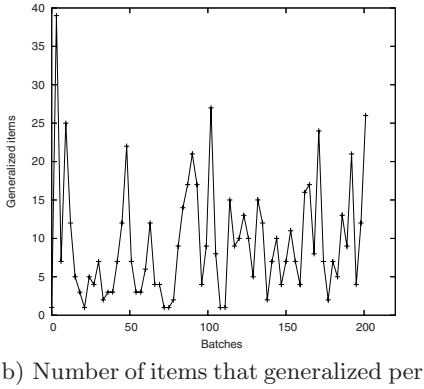


(d) Average transactions processed per second and per batch

Fig. 2. Experiments carried out on TCP/IP network data



(a) Number of items that specialized per batch



(b) Number of items that generalized per batch

Fig. 3. Specialization and generalization for the experiments carried out on TCP/IP network data

experimentations are listed in figures 2(a), 2(b), 2(c) and 2(d). The computation time for a batch over $DS1$ do not exceed 25 seconds, our algorithm never goes beyond this limit when extracting multidimensional sequences, leaving enough time for the data gathering for the next batch. The experimentation total time is 2600 seconds. Figure 2(a) depicts the computation time needed for the batches processing by our algorithm for a support value of $\sigma = 0.15\%$ and an error value of $\epsilon = 0.1\%$. $MDSDS$ is also bounded in term of memory constraint. Figure 2(c) depicts the memory behavior of the algorithm on $DS1$. We see that the space usage is bounded above and stable (less than 6 Mb) with spikes when extracting the sequences and updating the tilted-time windows. The effective processing rate is approximately 1400 transactions per second. Specializations (Figure 3(a)) and generalizations (Figure 3(b)) of the frequent multidimensional items vary between batches but is bounded by a maximum of 41 specializations and 39 generalizations between two batches. From these results, the network analyst expert can deduce some new knowledge on the state of the network traffic, for example between batch 150 and batch 151, the multidimensional item $(VER : 4, IPLEN : 5, TOS : 0, PLEN > 21, IPID : [10K, 20K[, [AF], DF : 1, PROT : 6, TCPHLEN : 5, SEQ\_NUM : +65K)$ which states that most of the packets are based on the TCP protocol with some options on the version, packet length etc gets specialized into $(VER : 4, IPLEN : 5, TOS : 0, PLEN > 21, IPID : [10K, 20K[, [AF], DF : 1, TTL : 128, PROT : 6, TCPHLEN : 5, SEQ\_NUM : +65K)$. Notice the appearance of $TTL : 128$, which states that at batch 150 most of the packets had different time-to-live values but starting from batch 151, most of the packets now have a time-to-live to 128. From a network analysis point of view, this means that all the packets are now topologically concentrated in less than 128 hops from the source to the destination host. Several other specializations or generalizations give some other insights on the traffic trends and evolutions. For example, items that specialize with the value $DSTP : [32K[$, means that the destination ports for the different applications are used for video or music streaming (applications for multimedia streaming usually have their ports number starting from 32771).

6 Conclusion

In this paper, we adress the problem of mining multidimensional sequential patterns in streaming data and propose the first approach called $MDSDS$ for mining such patterns. By considering the most specific multidimensional items, this approach efficiently detects trends. Experiments on real data gathered from TCP/IP network traffic provide compelling evidence that it is possible to obtain accurate and fast results for multidimensional sequential pattern mining. This work can be extended following several directions. For example, we can take hierarchies into account in order to enhance the trend detection and discover trends like "*In october, console sales are frequent whereas a specific console become frequent in November (Wii)*". Besides, to fit to OLAP framework, we should take

approximate values on quantitative dimensions into account with constrained base multidimensional pattern mining.

Finally, our results shows the potential of further work on multidimensional sequential pattern mining and specially in the new challenging data streams model.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proc. 1995 Int. Conf. Data Engineering (ICDE 1995), pp. 3–14 (1995)
2. Chen, G., Wu, X., Zhu, X.: Sequential pattern mining in multiple streams. In: ICDM, pp. 585–588. IEEE Computer Society, Los Alamitos (2005)
3. Chen, Y., Dong, G., Han, J., Wah, B.W., Wang, J.: Multi-dimensional regression analysis of time-series data streams. In: VLDB, pp. 323–334 (2002)
4. Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Moment: Maintaining closed frequent itemsets over a stream sliding window. In: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), pp. 59–66, Brighton, UK (2004)
5. Giannella, G., Han, J., Pei, J., Yan, X., Yu, P.: Mining frequent patterns in data streams at multiple time granularities. In: Kargupta, H., Joshi, A., Sivakumar, K., Yesha, Y. (eds.) Next Generation Data Mining. MIT Press, Cambridge (2003)
6. Han, J., Chen, Y., Dong, G., Pei, J., Wah, B.W., Wang, J., Cai, Y.D.: Stream cube: An architecture for multi-dimensional analysis of data streams. Distributed and Parallel Databases 18(2), 173–197 (2005)
7. Li, H.-F., Lee, S.Y., Shan, M.-K.: An efficient algorithm for mining frequent itemsets over the entire history of data streams. In: Proceedings of the 1st International Workshop on Knowledge Discovery in Data Streams, Pisa, Italy (2004)
8. Manku, G., Motwani, R.: Approximate frequency counts over data streams. In: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002), pp. 346–357, Hong Kong, China (2002)
9. Marascu, A., Masseglia, F.: Mining sequential patterns from data streams: a centroid approach. J. Intell. Inf. Syst. 27(3), 291–307 (2006)
10. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.-C.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions on Knowledge and Data Engineering 16(10) (2004)
11. Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q., Dayal, U.: Multi-dimensional sequential pattern mining. In: CIKM, pp. 81–88 (2001)
12. Plantevit, M., Choong, Y.W., Laurent, A., Laurent, D., Teisseire, M.: M^2SP: Mining sequential patterns among several dimensions. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 205–216. Springer, Heidelberg (2005)
13. Raïssi, C., Poncelet, P., Teisseire, M.: Need for speed: Mining sequential patterns in data streams. In: BDA (2005)
14. Yu, C.-C., Chen, Y.-L.: Mining sequential patterns from multidimensional sequence data. IEEE Transactions on Knowledge and Data Engineering 17(1), 136–140 (2005)