A Regression-Based Temporal Pattern Mining Scheme for Data Streams

Wei-Guang Teng
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, ROC
eev@arbor.ee.ntu.edu.tw

Ming-Syan Chen
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, ROC
mschen@cc.ee.ntu.edu.tw

Philip S. Yu
IBM T. J. Watson Research Center
P.O.Box 704
Yorktown, NY 10598
psyu@us.ibm.com

Abstract

We devise in this paper a regression-based algorithm, called algorithm FTP-DS (Frequent Temporal Patterns of Data Streams), to mine frequent temporal patterns for data streams. While providing a general framework of pattern frequency counting, algorithm FTP-DS has two major features, namely *one data scan for online statistics collection* and *regression-based compact pattern representation*. To attain the feature of one data scan, the data segmentation and the pattern growth scenarios are explored for the frequency counting purpose. Algorithm FTP-DS scans online transaction flows and generates candidate frequent patterns in real time. The second important feature of algorithm FTP-DS is on the regression-based compact pattern representation. Specifically, to meet the space constraint, we devise for pattern representation a compact ATF (standing for Accumulated Time and Frequency) form to aggregately comprise all the information required for regression analysis. In addition, we develop the techniques of the segmentation tuning and segment relaxation to enhance the functions of FTP-DS. With these features, algorithm FTP-DS is able to not only conduct mining with variable time intervals but also perform trend detection effectively. Synthetic data and a real dataset which contains network alarm logs from a major telecommunication company are utilized to verify the feasibility of algorithm FTP-DS.

Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

1 Introduction

The discovery of temporal relationship among a huge database has been known to be useful in selective marketing, decision analysis, and business management. An important application area of mining temporal relationship is the market basket analysis, which studies the buying behaviors of customers by searching for sets of items that are frequently purchased in a given temporal order.

In recent years, several query problems and mining capabilities have been explored for a data stream environment, including those on the statistics [7], the aggregate query [8, 12, 25], association rules [20], data clustering [13, 22], and data classification [9, 14], to name a few. For data stream applications, the volume of data is usually too huge to be stored on permanent devices or to be scanned thoroughly for more than once. It is hence recognized that both approximation and adaptivity are key ingredients for executing queries and performing mining tasks over rapid data streams. With the computation model presented in Figure 1 [11], a stream processor and the synopsis maintenance in memory are two major components for generating results in the data stream environment. Note that a buffer can be optionally set for temporary storage of recent data from data streams.

For time-variant databases, there is a strong demand for developing an efficient and effective method to mine various temporal patterns [6]. However, most methods which were designed for a traditional database cannot be directly applied to a dynamic data stream due not only to the high complexity of mining temporal patterns but also to the pass-through nature of data streams. Without loss of generality, a typical market-basket application is used in this paper for illustrative purposes. The transaction flow in such an

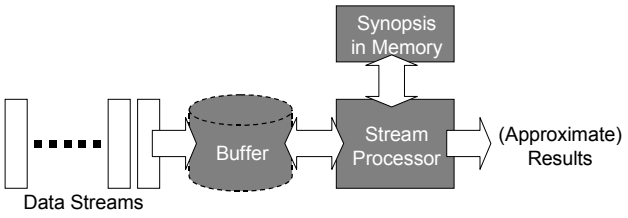

Figure 1: Computation model for data streams

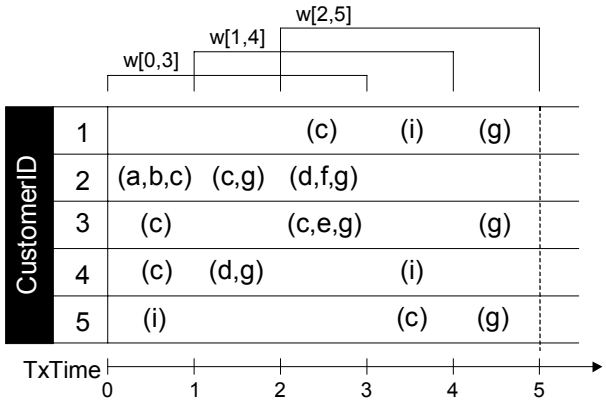

Figure 2: An example of online transaction flows

application is shown in Figure 2 where items a to g stand for items purchased by customers. For example, the third customer bought item c during time t=[0, 1), items c, e and g during t=[2, 3), and item g during t=[4, 5). It can be seen that in such a data stream environment it is intrinsically very difficult to conduct the frequent pattern identification due to the limited time and space constraints.

Consequently, we devise in this paper an algorithm FTP-DS (standing for Frequent Temporal Patterns of Data Streams) to mine frequent temporal patterns for data streams. While providing a general framework of pattern frequency counting, algorithm FTP-DS has two major features, namely *one data scan for online statistics collection* and *regression-based compact pattern representation*, which are designed to address, respectively, the time and the space constraints in a data stream environment. To attain the feature of one data scan, the occurrence frequency of a temporal pattern is first defined in accordance with the time constraint of sliding windows. Specifically, the data segmentation and the pattern growth scenarios are explored for this frequency counting purpose. Algorithm FTP-DS then scans online transaction flows and generates candidate frequent patterns in real time. With the downward closure property [24], longer patterns are gradually formed from their subsets as time advances. As such, frequent patterns are incrementally discovered and recorded by only one database scan. Note, however, that since a pattern is not deemed frequent until all its subsets are found frequent, the efficient frequent pattern identification with one data scan by FTP-DS is in fact at the cost of having some patterns recognized with delays due to the candidate pattern generation process. This phenomenon is referred to as delayed pattern recognition in this paper.

The second important feature of algorithm FTP-DS is on the regression-based compact pattern representation which is designed to address the space constraint of a data stream environment. Note that the data stream is generated continuously in a dynamic environment with huge volume, infinite flow, and fast changing behaviors. To maintain the synopsis of frequency variations for frequent temporal patterns, the regression analysis is utilized. After being transformed into a time series, the data stream is segmented and represented by one or more segments. Each segment of a time series is identified through the regression process. This regression analysis is employed to capture the trends of frequent patterns. Specifically, to meet the space constraint, we devise for pattern representation a compact ATF (standing for Accumulated Time and Frequency) form to comprise all the information required for regression analysis. In fact, it can be shown that using ATF forms to represent patterns, not only is the amount of storage space significantly reduced but also the trends can be efficiently detected. Consequently, only required synopses rather than historical details of frequent patterns are maintained during our discovering process, thereby leading to the ideal result that the required space is bounded.

With these two important features, algorithm FTP-DS is equipped with the adaptivity to answering queries and performing mining tasks with variable time periods, which is, in our opinion, very difficult to achieve by conventional methods. Moreover, this capability enables FTP-DS to perform trend detection effectively. Specifically, through the help of approximation results of regression analysis, the occurrence frequencies of frequent patterns can be retrospectively investigated. In addition, two enhanced segmentation techniques are employed for performance improvement. The first enhanced technique is the segmentation tuning which aims to reducing the error of constructing fit lines. Segmentation tuning can result in more precise segments, which in turn leads to more accurate mining results. The other enhanced technique is the segment relaxation which refers to the adjustment of segment periods along the time dimension in accordance with their corresponding importance to save the storage required. For example, people are usually more interested in recent changes than old ones. In view of this, one may want to model recent changes at a fine scale and old changes at a coarse scale. That is, the time is expected to be investigated at different levels of granularity [4, 25]. Therefore, as time advances, segments identified at prolonged time can be relaxed and merged to form an integrated segment, leading

to the guaranteed space bound for maintaining these historical segments.

Extensive experimental studies have been conducted to provide many insights into algorithms proposed. It is shown by empirical results that algorithm FTP-DS can meet both time and space limitations in a data stream environment. A real dataset which contains network alarm logs from a major telecommunication company is utilized to verify the feasibility of algorithm FTP-DS. As shown by our experimental results, while allowing of one data scan to meet the time constraint in a data stream environment, the delayed pattern recognition phenomenon in fact barely compromises the quality of mining results. In addition, the compact ATF form proposed for pattern representation and the segmentation techniques adopted by algorithm FTP-DS are shown to be very effective to limit the storage required, and lead to the efficient generation of temporal patterns of good quality. Moreover, sensitivity analysis on the lift of support threshold has also been conducted to provide more insights into algorithm FTP-DS.

The rest of the paper is organized as follows. The framework of frequency counting is presented in Section 2. The regression-based framework and the algorithm FTP-DS proposed are described in Section 3. Advantages of algorithm FTP-DS showing its practical usefulness are described in Section 4. Empirical studies are conducted in Section 5. This paper concludes with Section 6.

2 Preliminaries

2.1 Types of Temporal Patterns

In a temporal database, frequent patterns are usually targets of mining tasks. In many applications, a time-constraint is usually imposed during the mining process to meet the respective constraint. Specifically, the sliding window model is employed in this study, i.e., data expires after exactly N time units after its arrival where N is the user-specified window size. Consequently, a temporal pattern is frequent if its support, i.e., occurrence frequency, in the current window is no less than the threshold. Prior works have developed several models of temporal patterns, including the inter-transaction association rule [19], the causality rule [18], the episode [21] and the sequential pattern [1].

Note that the very difference among the above temporal patterns lies the ordering of occurrences. Mining of sequences corresponds to the one with strict order of events, while mining inter-transaction associations corresponds to the one without limitation on order of events. Between these two extremes, mining of causalities and episodes mainly emphasizes the ordering of triggering events and consequential events. Although the mining procedures may vary when being applied to discover different types of temporal patterns, a typical Apriori framework is commonly adopted. By utilizing the downward closure property in this framework [24], a fundamental issue of mining frequent temporal patterns is the frequency counting of patterns. In this paper, a frequency counting mechanism for a data stream environment is proposed.

2.2 Support Framework for Temporal Patterns

In market-basket analysis, the transaction data consists of records in the form of $<TxTime,\ CustomerID,\ Itemset>$ where *Itemset* is a set of items. In other words, a transaction record maps to a purchasing log generated by a single customer in a specific time. To evaluate the importance of a temporal pattern, the support, i.e., occurrence frequency, is a metric commonly used. However, the definition of support for a pattern may vary from one application to another. Consider again the market-basket database as an example. In mining sequential patterns [1], all the transactions of a customer can be viewed as a sequence together and the support for a sequential pattern is the fraction of customers whose purchasing sequences contain that pattern. Analogously, we have the model of frequency counting in mining causality rules [18]. On the other hand, in mining inter-transaction association rules [19], the repetitive occurrences of a pattern from an identical customer are counted cumulatively. Moreover, when the sliding window constraint is introduced in mining episodes [21], the support is defined to be the fraction of windows in which an episode occurs.

To deal with data streams, problems arise due to different support definitions. Specifically, since it is not possible to store all the historical data in the memory, to identify repetitive occurrences of a pattern is difficult. As a result, it is very important to properly formulate the support of temporal patterns. With the sliding window model, we define the support of a temporal pattern as follows.

Definition 1 *The support or the occurrence frequency of a temporal pattern X at a specific time t is denoted by the ratio of the number of customers having pattern X in the current time window to the total number of customers.*

Given the window size N=3, three sliding windows, i.e., w[0,3], w[1,4] and w[2,5], are shown in Figure 2 for the transaction flows. For example, according to Definition 1, supports of the inter-transaction itemset $\{c,\ g\}$ from TxTime t=1 to t=5 are obtained as in Table 1. Accordingly, the support variations can be presented as a time series as shown in Figure 3. For simplicity, the total number of customers is a constant in this example, and could be a variable as time advances in real applications.

TxTime		Occurrence(s) of {c, g}	Support
t=1	w[0,1]	none	0
t=2	w[0,2]	CustomerID={2, 4}	2/5=0.4
t=3	w[0,3]	CustomerID={2, 3, 4}	3/5=0.6
t=4	w[1,4]	CustomerID={2, 3}	2/5=0.4
t=5	w[2,5]	CustomerID={1, 3, 5}	3/5=0.6

Table 1: The support values of the inter-transaction itemset $\{c, g\}$

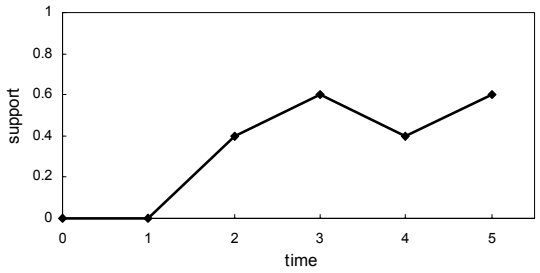

Figure 3: Support variations of the inter-transaction itemset (c, g)

As shown in Definition 1, the support variation of a specific temporal pattern as time advances can be represented as a single time series. Note that the support of a temporal pattern is generally defined to be a ratio of customers having that pattern without referring to any specific type of mining patterns, i.e., either inter-transaction associations, causalities, episodes, or the sequences. Without loss of generality, the temporal pattern of inter-transaction association is considered in this paper. With proper provisions, the same methods can be utilized in this support framework for other types of temporal patterns. Consequently, to conduct mining of the temporal patterns, the primary task is then to handle time series effectively and efficiently, which is in fact the design objective of algorithm FTP-DS proposed in this paper.

3 Mining Temporal Patterns in a Data Stream

The major features of algorithm FTP-DS proposed are described in Section 3.1. Algorithmic forms of FTP-DS are shown in Section 3.2. Section 3.3 presents two enhanced segmentation techniques, i.e., segmentation tuning and segment relaxation.

3.1 Major Features of Algorithm FTP-DS

3.1.1 One Scan for Statistics Collection

In essence, the frequency counting process is similar to that of incremental mining [5, 10, 17] in that new arriving transactions are being dealt with while some obsolete transactions are discarded due to the sliding window constraint, and is, however, different from the latter in that approximate answers in the model of data stream are allowed as a trade-off of having only one data scan.

As pointed out in [20], there are two major approaches to dealing with the frequency counting problem of data streams, i.e., one with a probabilistic error bound and the other with a deterministic error bound. Due to the limitation in processing data streams, both approaches are designed for obtaining approximate answers. The one with a probabilistic error bound is based on the sampling technique and that with a deterministic error bound is based on data segmentation technique. Note that although these approaches work successfully for counting supports of singleton items, as the number of items increases, the rapidly increasing number of temporal patterns can cause severe problems which include prohibitive storage and computing overheads. Explicitly, if the lossy counting scheme proposed in [20] is adopted, patterns with supports no less than ϵ are maintained during the mining process to guarantee the error range to be within ϵ. However, since the threshold ϵ, whose value could be one-tenth of MinSup, is usually too small to filter out uninteresting patterns, the storage space could be quite large. To address this point, in the sliding window model employed, only the occurrences of singleton items are being counted in the first time window. After the counting iteration, frequent items which have supports no less than the specified threshold are identified. These frequent items can be joined to generate candidate patterns of size two, which are then being counted in later iterations. After some patterns of size two are identified frequent, the candidate patterns of size three are generated and counted subsequently. As a result, longer candidate patterns are gradually generated, counted and verified to be frequent during the counting iterations. From the downward closure property [24], it follows that only patterns whose subpatterns are all frequent are taken as candidates and to be counted subsequently.

Example 1: Given the support threshold MinSup=0.4, the window size N=3 and the transaction flows in Figure 2, suppose the frequent inter-transaction associations are being generated. Since the temporal order is not required for inter-transaction associations, we have the frequent temporal itemset generation shown in Table 2. The support calculation of each itemset is the same as the process in Table 1. The averaged support value is represented by (accumulated supports over windows)/(number of recorded windows) in Table 2 where only itemsets with supports no less than MinSup=0.4 are listed. In addition, frequent itemsets generated in previous time window are used to generate longer candidates to be examined later. For example, according to Definition 1, the supports of itemset {d} during t=1 to t=5 are 0, 0.2, 0.4, 0.4 and 0.2, respectively. Not until t=3 does the support value satisfy the threshold, meaning that

itemset {d} is being tracked since t=3 as shown in Table 2(c) and Table 2(d). However, the averaged support of itemset {d} is (0.4+0.4+0.2)/3=1.0/3 which is less than the MinSup=0.4, making this itemset discarded in Table 2(e). Moreover, the inclusion of itemset {d} at t=3 results in the generation of related candidate itemsets, i.e., {c,d} and {d,g}, to be examined at t=4. However, only itemset {d, g} satisfies the support threshold and is included in Table 2(d).

t=1	
{c}	0.6/1

(a)

t=2	
{c}	1.2/2
{g}	0.4/1

(b)

t=3	
{c}	2/3
{d}	0.4/1
{g}	1/2
{c,g}	0.6/1

(c)

t=4	
{c}	2.8/4
{d}	0.8/2
{g}	1.6/3
{i}	0.4/1
{c,g}	1/2
{d,g}	0.4/1

(d)

t=5	
{c}	3.4/5
{g}	2.4/4
{i}	0.8/2
{c,g}	1.6/3

(e)

Table 2: Generation of frequent temporal itemsets (MinSup=0.4)

It can be seen that this approach can generate patterns of various lengths as time advances. However, as pointed out earlier, since a pattern is not taken as a candidate to accumulate its occurrence counts before all its subsets are found frequent, the phenomenon of delayed pattern recognition exists, i.e., some patterns are recognized with delays due to the candidate forming process in the data stream. For example, since items c and g are not both identified frequent until t=2, the candidate itemset $\{c,\ g\}$ is generated and counted at t=3. However, it can be verified from Table 1 that $\{c,\ g\}$ is actually frequent at t=2. Therefore, a delay of one time unit is introduced for discovering this itemset $\{c,\ g\}$. It is worth mentioning that only long transient frequent patterns could be neglected in this pattern generation process. As time advances, patterns with supports near the threshold will be further examined and identified to be frequent if so qualified. This is the very feature of delayed pattern recognition.

With a support threshold MinSup to filter out uninteresting patterns, only new patterns whose frequencies in the current time unit meet this threshold are being recorded. Supports of existing patterns, i.e., patterns which were already being recorded in previous time unit, are updated according to their support values in the current time unit. Note that, as time advances, patterns whose averaged supports fall below the threshold are removed from the records. Therefore, only frequent patterns are monitored and recorded. In practice, since a frequent pattern is not always with a very steady frequency, we can certainly delay the above mentioned removal and allow a pattern whose statistics are already recorded to stay in the system longer with an expectation that this pattern will become frequent again soon. This will be an application-dependent design alternative.

3.1.2 Regression-Based Analysis on Frequent Patterns

In the frequency counting model, the online transaction flows are transformed into numerical time series. Among various techniques used for analyzing time series, the regression analysis is adopted as the basis in our framework to mine temporal patterns in this paper. Through regression analysis, the estimated relationship can be used to predict frequency of patterns from previous experience [4]. A straight-line fit for a time series s(t), which corresponds to the frequency variation of a temporal pattern, is a linear estimation function $\widehat{\mathrm{f}} = \widehat{\alpha} + \widehat{\beta}\mathrm{t}$ that conforms to the principle of least squares. Specifically, the regression parameters $\widehat{\alpha}$ and $\widehat{\beta}$ are chosen to make the residual sum of squares $D = \sum_{i=1}^{n} (\mathrm{f}_i - \widehat{\alpha} - \widehat{\beta}\mathrm{t})^2$ minimal, where f_i is the actual frequency in the i-th recorded point.

To perform the calculations for getting best estimates of $\widehat{\alpha}$ and $\widehat{\beta}$, the following quantities are maintained,

$$\begin{aligned}
\overline{t} &= \frac{1}{n}\sum t,\ \overline{f} = \frac{1}{n}\sum f, \\
S_{tt} &= \sum (t-\overline{t})^2 = \sum t^2 - \frac{(\sum t)^2}{n}, \\
S_{ff} &= \sum (f-\overline{f})^2 = \sum f^2 - \frac{(\sum f)^2}{n}, \text{ and} \\
S_{tf} &= \sum (t-\overline{t})(f-\overline{f}) = \sum tf - \frac{(\sum t)(\sum f)}{n}.
\end{aligned}$$

Then, the least square estimates of $\widehat{\alpha}$ and $\widehat{\beta}$ are computed by

$$\widehat{\alpha} = \overline{f} - \widehat{\beta}\overline{t}, \text{ and } \widehat{\beta} = \frac{S_{tf}}{S_{tt}}.$$

In addition, the strength of a linear relation is measured by

$$r^2 = \frac{(S_{tf})^2}{S_{tt}S_{ff}}$$

which is the square of the sample correlation coefficient r and is also called the coefficient of determination.

Due to the nature of a data stream, some jitters may occur in our support framework which makes the corresponding time series rugged. However, since we are interested in the averaged supports of patterns, the individual occurrence frequency over time of a pattern can be aggregately transformed into the averaged support of that pattern, which will in fact make the time series much smoother and facilitate the subsequent processing work. A time series of averaged support (solid line) for the inter-transaction itemset $\{c,$

$g\}$ in Figure 2 is shown in Figure 4. The starting time of recording this pattern is t=3 since before that point the support constraint, i.e., MinSup=0.4, is not met. Delayed recognition occurs for this pattern $\{c, g\}$ as mentioned earlier. The regression technique is then applied to process the time series corresponding to the frequent patterns.

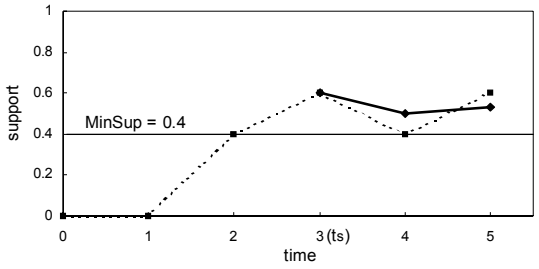

Figure 4: The time series of averaged support for the inter-transaction itemset (c, g)

Note that in a data stream environment, the data amount in the time series to track the frequency variations of individual temporal patterns could be very large. To meet the corresponding space constraint, we devise a compact form representation to maintain the information required for the regression-based analysis on frequent patterns. Specifically, only four measures are required to maintain during the regression process, i.e., the starting time (t_s), the accumulative product of time and frequencies ($\sum tf$), the accumulative sum of pattern frequencies ($\sum f$) and the accumulative squared sum of pattern frequencies ($\sum f^2$). Consequently, the corresponding representation form is referred to as the ATF form where ATF stands for Accumulated Time and Frequency.

Definition 2 *The ATF form of the time series corresponding to the frequency variation of a temporal pattern is (t_s, $\sum tf$, $\sum f$, $\sum f^2$), where t_s is the starting time, $\sum tf$ is the accumulated product of time and support, and $\sum f$ and $\sum f^2$ are, respectively, the sum and the squared sum of pattern frequencies since the pattern is recorded.*

By utilizing this compact ATF form for pattern recording, the storage needed for maintaining statistics of temporal patterns can be significantly reduced, thereby meeting the limited space requirement in a data stream environment. As shown by our experimental results, the memory usage in real applications can be kept very small. With its proof given below, Theorem 1 shows the feasibility of this compact ATF form.

Theorem 1: The least square error linear fit for a frequent temporal pattern can be obtained losslessly from its compact ATF form (t_s, $\sum tf$, $\sum f$, $\sum f^2$).

Proof: In a data stream environment, the current time t_{now} is always known and up-to-date as time advances. Using the starting time t_s, the accumulated values of $\sum t$ and $\sum t^2$ in the ATF form of a pattern can be obtained by $\sum t = \sum_{t_i=t_s}^{t_{now}} t_i$ and $\sum t^2 = \sum_{t_i=t_s}^{t_{now}} t_i^2$, since the corresponding time series for a pattern is composed of the averaged support values at every time unit during t=[t_s, t_{now}]. In addition, the number of recorded points is n=t_{now}-t_s+1.

Together with the other three measures, i.e., $\sum tf$, $\sum f$, $\sum f^2$, the values of S_{tt}, S_{ff} and S_{tf} can be obtained through the equations:

$$\begin{aligned} S_{tt} &= \sum t^2 - \frac{(\sum t)^2}{n}, \\ S_{ff} &= \sum f^2 - \frac{(\sum f)^2}{n}, \text{ and} \\ S_{tf} &= \sum tf - \frac{(\sum t)(\sum f)}{n}. \end{aligned}$$

Consequently, the least square estimates of $\widehat{\alpha}$ and $\widehat{\beta}$ are computed by

$$\begin{aligned} \widehat{\beta} &= \frac{S_{tf}}{S_{tt}}, \text{ and} \\ \widehat{\alpha} &= \overline{f} - \widehat{\beta}\overline{t} = \frac{\sum f}{n} - \widehat{\beta} \times \frac{\sum t}{n}. \end{aligned}$$

Therefore the fit line $\widehat{f} = \widehat{\alpha} + \widehat{\beta}t$ for this pattern can be precisely extracted from the ATF compact form. **Q.E.D.**

Example 2: Given the frequency threshold MinSup=0.4, and the averaged support variations of the inter-transaction itemset $\{c, g\}$ in Figure 4, we have the following measures according to Definition 2,

$$\begin{aligned} t_s &= 3, \\ \sum tf &= 3 \times 0.6 + 4 \times 0.5 + 5 \times 0.5333 = 6.4665, \\ \sum f &= 0.6 + 0.5 + 0.5333 = 1.6333, \text{ and} \\ \sum f^2 &= 0.6^2 + 0.5^2 + 0.5333^2 = 0.8944. \end{aligned}$$

Therefore, the ATF form for itemset $\{c, g\}$ at t=5, is (3, 6.4665, 1.6333, 0.8944). In addition, we have the following quantities $\sum t = 3 + 4 + 5 = 12$, and $\sum t^2 = 3^2 + 4^2 + 5^2 = 50$. Consequently, $S_{tt} = 50 - \frac{(12)^2}{3} = 2$, $S_{ff} = 0.8944 - \frac{(1.6333)^2}{3} = 5.1770 \times 10^{-3}$ and $S_{tf} = 6.4665 - \frac{12 \times 1.6333}{3} = -0.0667$.

Finally, $\widehat{\beta} = \frac{S_{tf}}{S_{tt}} = \frac{-0.0667}{2} = -0.0333$, and $\widehat{\alpha} = \overline{f} - \widehat{\beta}\overline{t} = \frac{1.6333}{3} + 0.0333 \times \frac{12}{3} = 0.6777$. The fit line is then $\widehat{f}$=0.6777-0.0333t.

In practice, the compact ATF form of a frequent temporal pattern can be obtained when this pattern starts to be recorded. According to Definition 2, except for the starting time t_s which is a fixed value, the

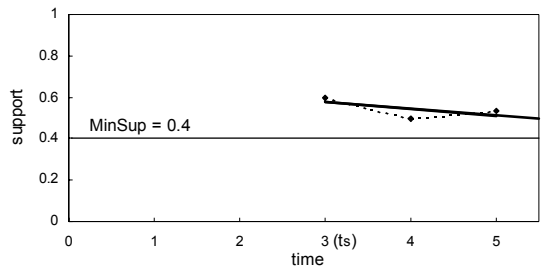

Figure 5: The regression analysis for Example 2 where the dotted line represents the variations of averaged supports for itemset $(c,\ g)$ and the solid line is the corresponding fit line

other three measures are all aggregate values. Consequently, as time advances these three measures mainly change according to the corresponding time and frequency values and the compact ATF form of a frequent pattern can always be precisely obtained.

3.2 Algorithm of FTP-DS

With its two major features described above, the algorithmic form of FTP-DS is next presented.

3.2.1 Piecewise Linear Representation

To efficiently and effectively represent the data of time series, several representations with related approaches have been proposed, including Fourier transforms, wavelets, symbolic mappings and piecewise linear representation. Without loss of generality, we employ the piecewise linear representation in this work. In [16], the problem of transforming a time series into a piecewise linear representation is referred to as a segmentation problem. Among various approaches for time series segmentation, the one utilizing sliding windows is deemed appropriate to meet the requirement of online processing for data streams. To evaluate the quality of a fit line, the measure of the residual error or the sum of squares is commonly used.

In essence, the segmenting algorithm is to cut at some time points where the error for corresponding segment exceeds the user-specified threshold. As such, segments are generated and represented as a list of ATF forms when a mining task is performed. In fact, the actual time series for a temporal pattern can be approximately reconstructed by this ATF list. Since the residual error is bounded by a user-specified threshold, the error resulting from this approximation for the whole time series is hence bounded. Note that since mining over data streams may continue forever, the three summations in Definition 2 could grow indefinitely in theory. We comment that since the summations are reset as a new data segment is created, one can certainly have another segment deliberately created so as to limit the growth of summations if so necessary in practice.

3.2.2 Flow of Algorithm FTP-DS

Given the support framework of temporal patterns for data streams and the regression-based techniques, the algorithmic form of mining frequent temporal patterns can be outlined below.

Algorithm FTP-DS: Frequent Temporal Patterns of Data Streams
Input: The window size N, and the support threshold MinSup
Output: The set of frequent temporal patterns F with their ATF forms

```
1. t=0;
2. buf=NULL;  //buffer for storing recent transactions
3. F={all singleton items};  //initial candidate patterns
4. while(1){
5.    wait until t=t+1;
6.    data_t=transactions in time slot t;
7.    buf=(buf∪data_t)−data_{t−N};  //data_{t−N} is expired
8.    foreach(p∈F){
9.       p.count=0;
10.      p.nWindow+=1;
11.   }
12.   count occurrences of each p∈F from buf;
13.   foreach(p∈F){
14.      p.sup=(p.sup+p.count/nCustomer)/p.nWindow;
15.      if(p.sup<MinSup)  remove p from F;
16.      else              update the ATF forms of p;
17.   }
18.   F=F∪{candidates generated from F};
19.}
```

As mentioned earlier, since a frequent pattern does not always have a very steady frequency, the frequency threshold can be slightly lifted to be higher than the real threshold MinSup. This option can help reducing the overhead for recording unnecessary patterns with the trade-off of a delayed recognition of some frequent patterns.

3.3 Enhanced Techniques for FTP-DS

3.3.1 Compensation for Delayed Pattern Recognition

The basic approach of one data scan adopts a levelwise style to construct candidates for future examination. There could be some drawbacks when potential frequent patterns are long. A direct and significant impact is introduced on the memory required for maintaining all the frequent patterns under this circumstance. Suppose a frequent pattern of size 10 is identified, one would need to track $2^{\mathbf{10}}$ sub-patterns, which is very costly. Therefore, an alternative approach is that only the maximum patterns and some near-maximum patterns are tracked. This will help on saving memory usage while sacrificing a little recovery ability when

patterns become infrequent gradually. A redundant process is then crucial for rebuilding list of frequent patterns.

Another compensation technique for alleviating the phenomenon of delayed pattern recognition is to use depth first or look ahead approaches instead of the levelwise one. Consequently, longer patterns could be identified earlier which reduces the potential time delay. Moreover, the border collapsing technique proposed in [23] can also be employed. These approaches have a common feature that a more complex policy on pattern development is employed while a little more space could be needed for examining extra candidate patterns of longer sizes. To compensate the delayed pattern recognition, these techniques can be seamlessly incorporated into our framework proposed as design alternatives.

3.3.2 Segment Tuning and Relaxation

As noted in [16], to raise the accuracy of segmentation, a process which utilizes the buffer to facilitate the fine-tuning of segments is devised as an enhancement. Specifically, the transition point between two recent segments is adjusted according to the buffered data to raise the accuracy of using these segments to represent the whole time series. This technique is employed in our implementation and shown to be able to greatly improve the effectiveness of the piecewise linear regression representation.

In addition, to ease the problem of memory explosion as time advances, the segment relaxation technique is proposed. It is noted that though the piecewise linear regression is able to improve accuracy of the approximation to a time series, the storage of corresponding ATF form grows as time advances. In fact, with the piecewise linear regression representation employed, we can adjust the level of temporal granularity [4, 25] in accordance with the interestingness of the data period. For example, people are often interested in recent changes at a fine scale, but old changes at a coarse scale. Therefore, as time advances, some old segments can be relaxed and merged, as illustrated by the region between point a and point b in Figure 6, to form an integrated segment covering those time periods, thus achieving the purpose of saving storage for maintaining these historical segments. Though the approximation accuracy is lowered due to this segment relaxation, the storage space required is reduced. With this enhancement, the proposed approach is able to guarantee a bounded storage required for handling data streams. In all, this technique not only exploits the key ingredients, i.e., approximation and adaptivity, in performing data processing over rapid data streams but also supports the model of having implied weights on data segments with different interests, showing another advantage of the regression-based frequency counting framework proposed.

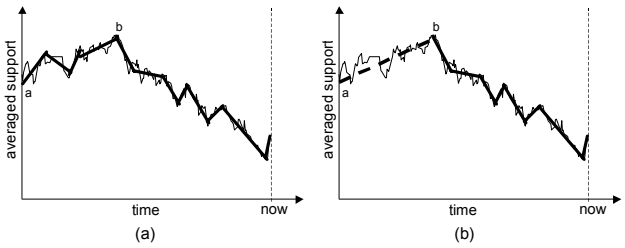

Figure 6: An example of segment relaxation: (a) the original segments; (b) with some old segments merged

4 Advantages of Algorithm FTP-DS

In this section, we present two advantages of algorithm FTP-DS, namely mining with flexible time intervals and trend detection, which are, in our view, not only of practical importance but also difficult to achieve by conventional methods.

4.1 Mining with Flexible Time Intervals

As stated in [3], an example query specification in a data stream environment can be as follows:

SELECT *
FROM DataStream
WHERE (Query Clause)
BEGIN (BeginTime)
END (EndTime).

The BEGIN-END clause can either be constant, relative time to the current system clock, i.e., NOW, or a variable. In the proposed framework, since the frequency variations of frequent patterns are recorded, the answers to queries with variable time intervals are directly supported. Furthermore, this result can be extended to the mining with variable time intervals. Specifically, the support variations of a temporal pattern can be retrospectively investigated through the use of its fit lines extracted from corresponding ATF forms if this pattern is frequent during the time interval queried.

Example 3: Consider the inter-transaction itemset $\{c, g\}$ again. Since the regression-based approach is adopted in algorithm FTP-DS, instead of the actual support values at each time in Figure 4, the ATF compact form of $\{c, g\}$ is the only information maintained. Suppose that one would like to answer a query of averaged supports during the time interval t=[4, 5]. The fit line of $\{c, g\}$ is extracted from its ATF form at the time t=5. The corresponding equation is $\widehat{f}$=0.6777-0.0333t which is derived in Example 2. Thus, we have $\widehat{f}(3)$ =0.5778 and $\widehat{f}(5)$=0.5112. Since t_s=3 and $\widehat{f}(5)$ is the average value of the 3 support values during t=[3,5],

we consequently have the averaged support queried as:

$$\begin{aligned} & \frac{1}{5-4+1}[(5-t_s+1)\cdot\widehat{\mathrm{f}}(5)-(3-t_s+1)\cdot\widehat{\mathrm{f}}(3)] \\ = \; & \frac{1}{2}\times(3\times 0.5112-1\times 0.5778) \\ = \; & 0.4779. \end{aligned}$$

Note that the actual answer should be $\frac{(0.4+0.6)}{2}=0.5$, showing a very small approximation error of about 4%. Using the regression approach, answers to similar queries of different time intervals can be estimated.

Analogous retrospection process can be conducted on performing mining tasks with variable time intervals. Provided that the patterns are previously frequent in requested time interval, the support variations of these patterns can be estimated similarly. Therefore, even if a higher frequency threshold is specified, supports of each pattern in a given time interval can be retrospectively estimated and then verified to see if this pattern is frequent. Consequently, without resorting to any data scans, the set of frequent patterns subject to new constraints can be produced.

4.2 Trend Identification and Change Detection

The mining results in our framework can be reported in two alternative ways. The snapshot reports generated at distinct time are basically the same as those generated in a batch environment, stating the current frequent patterns. The snapshot reports for frequent patterns are obtained through carefully tracking of frequency variations as shown in our examples. However, in practical applications, reporting some patterns constantly is of little importance. On the other hand, as also noted in many recent works [2, 15], a great interest has been developed in identifying trends and detecting changes over time. By utilizing the one data scan approach proposed, the frequent patterns being added or removed from the ATF list in each time window can be obtained in real time. Moreover, the regression-based framework is able to provide insights into the trends of frequent patterns. Specifically, the slope of a segment can clearly represent the trend of occurrence frequencies for a temporal pattern. In addition, the disjunctive points of segments when performing piecewise linear regression are critical for detecting changes.

To evaluate the usefulness of algorithm FTP-DS on these aspects, some changes of item distribution have been deliberately introduced into the synthetic datasets in our empirical studies. Also, the change reports are provided in addition to the snapshot reports for comparison purposes.

5 Experimental Results

The simulation model of our experimental studies is described in Section 5.1. To assess the performance of algorithm FTP-DS, we conduct two empirical studies based on both the synthetic and the real datasets. The feasibility and the scalability of algorithm FTP-DS are examined in Section 5.2. FTP-DS is compared with a batch mode algorithm in Section 5.3. Sensitivity analysis on the lift of support threshold is conducted in Section 5.4. Performance of algorithm FTP-DS on a real dataset is evaluated in Section 5.5.

5.1 Simulation Model

In our experiments, we use two synthetic datasets generated by a randomized transaction generation algorithm in [1]. The synthetic data generation program takes the parameters shown in Table 3, and the values of parameters used to generate the datasets are summarized in Table 4. Both the environments of long transactions with fewer items and short transactions with more items are considered. In addition, a real dataset of network alarm logs provided by a major telecommunication company is employed to verify the feasibility of algorithm FTP-DS. In this real dataset, various alarms generated by a huge number of base station controllers are logged and the corresponding parameters for this dataset are also listed in Table 4.

N	Number of items
T	Average numbers of items per transaction
C	Number of customers
D	Number of transactions

Table 3: Parameters of the synthetic datasets

Name	N	T	C	D
N200T3C1000	200	3	1000	500,000
N100T5C1000	100	5	1000	500,000
AlarmLog	287	1.6	5788	128,815

Table 4: Parameter settings of synthetic datasets

The simulation is coded in C++ and performed in a 1.7GHz IBM compatible PC. The default values of support threshold are 20% (for N200T3C1000) and 25% (for N100T5C1000) respectively. The memory usage reported refers to the memory space consumed for storing frequent patterns. To emulate the processing of data streams, the testing datasets are input to algorithm FTP-DS in a pass-through fashion. Explicitly, except for a limited amount of transaction data buffered in compliance with the sliding window constraint, algorithm FTP-DS will never look over the historical transaction details.

5.2 Feasibility and Scalability of FTP-DS

To evaluate the feasibility and the scalability of algorithm FTP-DS, a scale-up experiment is first conducted. Specifically, in the temporal pattern mining of a market-basket application, the number of customers is increased from 1,000 to 10,000. As shown in Figure 7, the execution time grows smoothly as the dataset

size increases. The major reason incurring the slightly nonlinear growth is that as the dataset size increases, the corresponding time required for processing recent data in the buffer increases.

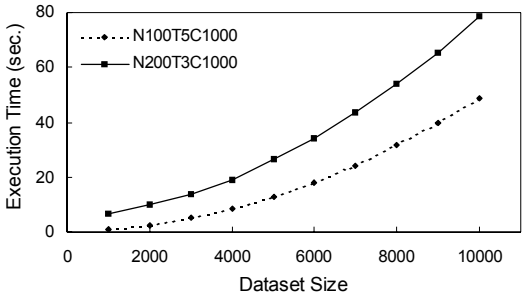

Figure 7: The scale-up results for synthetic datasets

The memory and the execution time required are two primary factors in the mining of a data stream, since both should be bounded online as time advances. Since the transaction data is fed in a pass-through fashion, the memory consumption is recorded as the number of processed transactions increases. This is represented by the ratio to the whole transactional dataset and is labeled as "relative time" on the x-axis. Note that the memory usage in Figure 8(a) for both synthetic datasets is steadily bounded, owing to the advantageous features of using the compact ATF form and also the segment relaxation technique. In addition, the execution time shown in Figure 8(b) is also very stable as time advances, indicating the feasibility of algorithm FTP-DS. Note that the execution time here refers to the time consumed on processing online transactions during a time window.

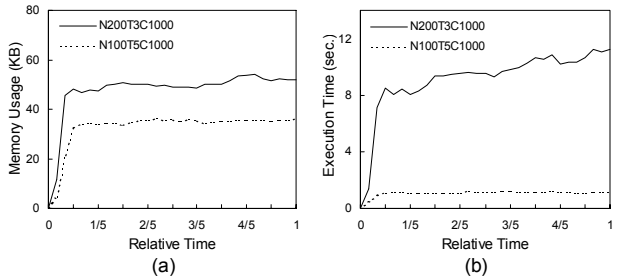

Figure 8: Required resources for synthetic datasets: (a) memory; (b) execution time

5.3 Comparison with Batch Mode Algorithm

Note that FTP-DS which is an online algorithm in nature is intrinsically different from a traditional batch mode algorithm. To illustrate the advantage gained by the online nature of FTP-DS, we implement a typical Apriori algorithm which works in batch mode and is allowed to scan data multiple times to get corresponding intermediate results. At a specific time, only transactions within current window are fed to FTP-DS whereas all transactions generated so far are input to Apriori. Clearly, Apriori will process more transactions as time advances. Consequently, the mining process is performed iteratively and the results of patterns discovered and the corresponding execution times are recorded accordingly. The experimental results on the dataset N100C5T1000 are shown in Figure 9. Note that since a candidate pattern cannot be formed and counted before all its subsets are identified frequent, the number of frequent patterns discovered by the one scan approach will be no greater than the actual number of frequent patterns. In addition, as discussed earlier, the phenomenon of delayed pattern recognition occurs due to the limitation imposed by the one data scan in a data stream environment. To capture the resulting effects from this phenomenon, it is observed from Figure 9(a) that the number of patterns generated by FTP-DS is noticeably smaller at first, but quickly approaches the actual number of frequent patterns. This result shows that though there is some short delays due to the one scan approach, most of the frequent patterns can be quickly and effectively discovered. Moreover, from Figure 9(b) the execution time (with a logarithmic scale on the y-axis) incurred by FTP-DS is quite steady and is much shorter than that of Apriori, i.e., the batch mode approach, showing that FTP-DS performs more efficiently without compromising the quality of results.

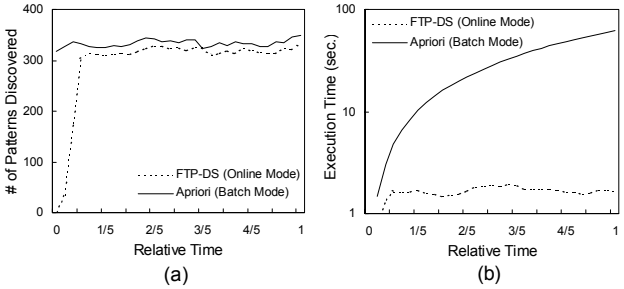

Figure 9: Comparison of online and batch mode algorithms

5.4 Sensitivity Analysis

Note that since a frequent pattern is not always with a very steady frequency in practice, the frequency threshold can be slightly lifted to a higher value to reduce the overhead for recording unnecessary patterns. The lift of a support threshold will mainly filter out transient frequent patterns which are only frequent during short time periods. On the other hand, frequent patterns which last for long periods are essentially not affected by this lifted support threshold. To conduct this sensitivity analysis, experiments on the synthetic dataset N100T5C1000 are performed. From the results

shown in Figure 10, frequent patterns are successfully discovered, meaning that algorithm FTP-DS can generate frequent patterns precisely with less memory and shorter execution time. Note that the lift of a support threshold will unavoidably aggravate the delayed pattern recognition scenario. However, this drawback introduces little extra overhead in our framework since iterative frequency counting is already employed in our scheme for mining a data stream. In addition, the lossy counting scheme proposed in [20] is also tested with error bound ϵ=10%. The execution time is plotted in Figure 10(b), whose value, as indicated in the second y-axis, is in orders of magnitude larger than those by FTP-DS.

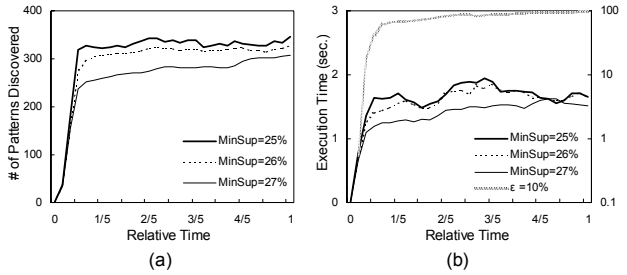

Figure 10: Impact of lifting support threshold: (a) number of patterns discovered; (b) execution time

5.5 Experiments on a Real Dataset

Analogous experiments are conducted on the real dataset, i.e., AlarmLog, to evaluate the performance of algorithm FTP-DS. The alarm records are first divided into windows of hours and the window size is selected to be eight hours in the sliding window mechanism. Experimental results on other granularities of window sizes did not provide additional insight and are omitted here. Unlike the item occurrences for synthetic datasets that are steady over time due to the random generation, those for a real dataset are of a big variance over time, indicating data skew over time. This feature can be observed from Figure 11 where both the memory usage and the execution time consumed in each window are quasi-periodic with the period of exactly a day. Though not being steady over time, the memory consumed and the corresponding execution time are still bounded since there is no increasing trend for either curve in Figure 11, showing the robustness of algorithm FTP-DS.

Moreover, it is interesting to observe the item distribution and the pattern distribution. From Figure 12, the alarm distribution is indeed of the period of a day and the peaks fall on the afternoons. It is also noted that during the weekends, i.e., 2/4, 2/5, 2/11 and 2/12, the alarms are fewer than those on weekdays. These observations confirm to the application nature in this dataset.

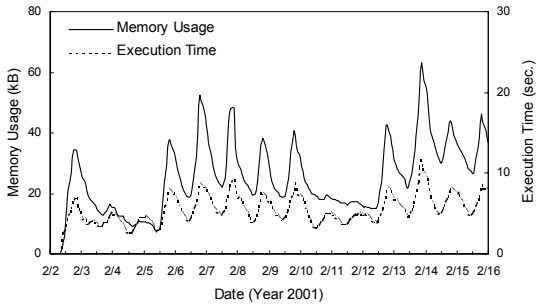

Figure 11: Memory and execution time required for the real dataset AlarmLog

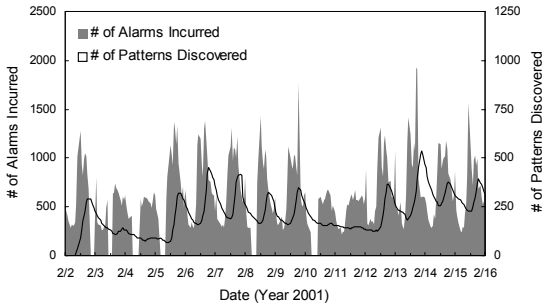

Figure 12: Comparison of alarm and pattern distributions

Furthermore, the number of patterns discovered is also depicted in Figure 12 for comparison purposes. It is observed that the pattern distribution is of about two hour delay to the alarm distribution, which in fact confirms to our expectation and indeed shows a scenario resulting from a combination of the use of a sliding window approach and also the delayed pattern recognition phenomenon. It is interesting to note that if a batch mining task is performed daily, the average delay is up to half a day, i.e., 12 hours, which is significantly larger than the period of 2 hours in our experiments.

6 Conclusions

We have devised in this paper a regression-based algorithm FTP-DS to mine frequent temporal patterns for data streams. Algorithm FTP-DS has two major features, namely one data scan for online statistics collection and regression-based compact pattern representation which are designed to address, respectively, the time and the space constraints in a data stream environment. With these features, algorithm FTP-DS is able to not only conduct mining with variable time intervals but also perform trend detection effectively. As shown by our experimental results, while allowing of one data scan to meet the time and space constraints in a data stream environment, FTP-DS is able to obtain the mining results of very good quality. Moreover,

sensitivity analysis on the lift of support threshold has also been conducted to provide more insights into algorithm FTP-DS.

Acknowledgement

The authors are supported in part by the National Science Council, Project No. NSC 91-2213-E-002-034 and NSC 91-2213-E-002-045, Taiwan, Republic of China.

References

[1] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proceedings of the 11th International Conference on Data Engineering*, pages 3–14, March 1995.

[2] S. Chakrabarti, S. Sarawagi, and B. Dom. Mining Surprising Patterns Using Temporal Description Length. *Proceedings of the 24th International Conference on Very Large Data Bases*, pages 606–617, August 1998.

[3] S. Chandrasekaran and M. J. Franklin. Streaming Queries over Streaming Data. *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 203–214, August 2002.

[4] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-Dimensional Regression Analysis of Time-Series Data Streams. *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 323–334, August 2002.

[5] D. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. *Proceedings of the 12th International Conference on Data Engineering*, pages 106–114, February 1996.

[6] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule Discovery from Time Series. *Proceedings of the 4th ACM SIGKDD*, pages 16–22, August 1998.

[7] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining Stream Statistics over Sliding Windows. *Proceedings of the 2002 Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2002.

[8] A. Dobra, M. N. Garofalakis, J. Gehrke, and R. Rastogi. Processing Complex Aggregate Queries over Data Streams. *Proceedings of the 2002 ACM SIGMOD*, pages 61–72, June 2002.

[9] P. Domingos and G. Hulten. Mining High-Speed Data Streams. *Proceedings of the 6th ACM SIGKDD*, pages 71–80, August 2000.

[10] V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining Data Streams under Block Evolution. *SIGKDD Explorations*, 3(2):1–10, January 2002.

[11] M. N. Garofalakis, J. Gehrke, and R. Rastogi. Querying and Mining Data Streams: You Only Get One Look. *Proceedings of the 2002 ACM SIGMOD*, June 2002.

[12] J. Gehrke, F. Korn, and D. Srivastava. On Computing Correlated Aggregates Over Continual Data Streams. *Proceedings of the 2001 ACM SIGMOD*, pages 13–24, May 2001.

[13] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 359–366, November 2000.

[14] G. Hulten, L. Spencer, and P. Domingos. Mining Time-Changing Data Streams. *Proceedings of the 7th ACM SIGKDD*, pages 97–106, August 2001.

[15] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying Representative Trends in Massive Time Series Data Sets Using Sketches. *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 363–372, September 2000.

[16] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An Online Algorithm for Segmenting Time Series. *Proceedings of the 1st IEEE International Conference on Data Mining*, pages 289–296, November 2001.

[17] C.-H. Lee, C.-R. Lin, and M.-S. Chen. Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining. *Proceeding of the ACM 10th International Conference on Information and Knowledge Management*, pages 263–270, November 2001.

[18] C.-H. Lee, P. S. Yu, and M.-S. Chen. Causality Rules: Exploring the Relationship between Triggering and Consequential Events in a Database of Short Transactions. *Proceedings of the 2nd SIAM International Conference on Data Mining*, pages 403–419, April 2002.

[19] H. Lu, J. Han, and L. Feng. Stock Movement Prediction and N-Dimensional Inter-Transaction Association Rules. *Proceedings of the 1998 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 12:1–12:7, June 1998.

[20] G. S. Manku and R. Motwani. Approximate Frequency Counts over Streaming Data. *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 346–357, August 2002.

[21] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[22] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. High-Performance Clustering of Streams and Large Data Sets. *Proceedings of the 18th International Conference on Data Engineering*, February 2002.

[23] J. Yang, W. Wang, P. S. Yu, and J. Han. Mining Long Sequential Patterns in a Noisy Environment. *Proceedings of the 2002 ACM SIGMOD*, pages 406–417, June 2002.

[24] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. *Proceedings of the 3rd ACM SIGKDD*, pages 283–286, August 1997.

[25] D. Zhang, D. Gunopulos, V. J. Tsotras, and B. Seeger. Temporal Aggregation over Data Streams Using Multiple Granularities. *Proceedings of the 8th International Conference on Extending Database Technology*, pages 646–663, March 2002.