Adaptive Clustering for Multiple Evolving Streams

Bi-Ru Dai, Jen-Wei Huang, Mi-Yen Yeh, and Ming-Syan Chen, *Fellow*, *IEEE*

Abstract—In the data stream environment, the patterns generated at different time instances are different due to data evolution. As time progresses, the behavior and members of clusters usually change. Hence, clustering continuous data streams allows us to observe the changes of group behavior. In order to support flexible clustering requirements, we devise in this paper a *Clustering on Demand framework*, abbreviated as *COD framework*, to dynamically cluster multiple data streams. While providing a general framework of clustering on multiple data streams, the COD framework has two advantageous features, namely, *one data scan for online statistics collection* and *compact multiresolution approximations*, which are designed to address, respectively, the time and the space constraints in a data stream environment. The COD framework consists of two phases, i.e., the *online maintenance phase* and the *offline clustering phase*. The online maintenance phase provides an efficient mechanism to maintain summary hierarchies of data streams with multiple resolutions in time linear in both the number of streams and the number of data points in each stream. On the other hand, an *adaptive clustering algorithm* is devised for the offline phase to retrieve approximations of desired substreams from summary hierarchies according to clustering queries. We propose two summarization techniques, based on wavelet and regression analyses, to construct the summary hierarchies. The regression-based summary hierarchy approximates the data stream more precisely and provides better clustering results, at the cost of slightly longer time than and twice the storage space as the wavelet-based one. An adaptive version of COD framework is designed to make a selection between a wavelet-based model and a regression-based model for building the summary hierarchy. By the adaptive COD, we can obtain clustering results with almost the same quality as the regression-based COD while using much less storage space for the summary hierarchy. As shown in the complexity analyses and also validated by our empirical studies, the COD framework performs very efficiently in the data stream environment while producing clustering results of very high quality.

Index Terms—Data mining, clustering of multiple data streams, time-series clustering.

1 Introduction

In recent years, several query problems and mining capabilities have been explored for the data stream environment [1], [2], including those on the summarization and statistics [3], [4], [5], data selection [6], change detection [7], [8], sampling [9], data clustering [10], [11], [12], [13], and data classification [14], [15], [16], 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 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.

The clustering problems have been studied for a data stream environment recently [10], [11], [12], [13]. In this paper, the problem of clustering multiple data streams is addressed. It is assumed that at each time stamp, data points from individual streams arrive simultaneously, and the data points are highly correlative to previous ones in the same stream. Unlike that of prior studies, the objective in this work is to partition these data streams, rather than their data points, into clusters. Note that the data streams are not of a fixed length. Instead, they are still evolving when the clustering results are observed at users' requests. For the sake of simplicity, the model of one-dimensional data streams is explored, which can also be regarded as multiple streaming time-series.

In the data stream environment, the patterns generated at different time instances are different due to data evolution. The frequency of change for rules and patterns is different for different applications. For example, the streams gathered from adjacent sensors may always be in the same cluster and rarely change as time progresses. On the other hand, for stock prices, some companies are probably within the same cluster during several months but in different clusters afterward. The clusters obtained hence change frequently. In order to deal with various types of multiple data streams in the same stream mining system, an important question arises: "*Can we design a scheme for modeling both fast and slow evolving patterns adaptively?*"

Consider the stock market data again. Assume that the clustering request is unknown when the data is collected and processed. After the time range of clustering request is given, recommendations for short-term or long-term investments are desired to be offered precisely. For example, some users are interested in the short-term behavior of clusters, such as the daily or hourly behavior. Therefore, we would like to provide daily clusters for one week or hourly clusters for a few hours. On the other hand, for users interested in the long-term behavior of clusters, such as the monthly or annual behavior, we would like to provide

• *The authors are with the Department of Electrical Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan, ROC. E-mail: {brdai, jwhuang, miyen}@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw.*

Manuscript received 5 June 2005; revised 18 Dec. 2005; accepted 13 Apr. 2006; published online 19 July 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0226-0605.

monthly clusters for this year or annual clusters for last five years. This leads to another important issue: "*Can we provide a system to support various clustering requirements at the same time?*" Clustering multiple data streams is very useful in various applications. As time progresses, the behavior and members of clusters usually change. Hence, clustering continuous data streams allows us to observe the changes of group behavior. For example, oceanographic and surface meteorological data consist of measures such as temperature, humidity, surface winds, and so on. These values are recorded continuously and probably change with the ocean currents, rainfall, solar radiation, and other factors. Therefore, by clustering these continuous streams, we are able to observe that some streams are similar in a period and become dissimilar afterward. For example, we can query monthly temperature clusters or annual humidity clusters. Then, the clustering query can be set as follows: the window size is set as one month and several months are observed, or the window size is set as one year and several years are observed, respectively. Then, according to the changes of cluster members between windows, oceanographers can obtain useful knowledge for further analysis such as building prediction models for future trends. Consequently, we devise in this paper a framework of *Clustering on Demand*, abbreviated as *COD framework*, to dynamically cluster multiple data streams. While providing a general framework of clustering on multiple data streams, the proposed COD framework has the following two advantageous features. The first one is one data scan for online statistics collection, and the second one is compact multiresolution approximations, which are designed to address, respectively, the time and the space constraints in a data stream environment. Furthermore, with the multiresolution approximations of data streams, flexible clustering demands can be supported. Note that since the clustering algorithms are only applied to the statistics maintained rather than to the original data streams, the COD proposed is able to work very efficiently in practice.

The COD framework proposed consists of two phases, namely, the *online maintenance phase* and the *offline clustering phase*. The online maintenance phase provides an efficient algorithm to maintain the summary hierarchies of the data streams with multiple resolutions in time linear in both the number of streams and the number of data points in each stream. On the other hand, an *adaptive clustering algorithm* is devised for the offline phase to retrieve the approximations of the desired substreams from the summary hierarchies as precisely as possible according to the clustering queries specified by the users. Note that the data streams are generated continuously in a dynamic environment with a huge volume and infinite flow. To maintain the summaries of the evolving streams, we propose two summarization techniques based on 1) wavelets and 2) regression lines. In general, we keep finer approximations for more recent data and coarser approximations for older data. In the wavelet-based hierarchy, a flat line is used to approximate an interval of a stream. In contrast, a straight fitting line is applied in the regression-based hierarchy. Since a flat line approximation can be calculated more efficiently and maintained with fewer parameters, the wavelet-based hierarchy can be built faster while using less storage space than the regression-based one. However, the straight fitting line utilized in the regression-based hierarchy can approximate the stream more precisely, at the cost of slightly longer time and twice the storage space to build the summary hierarchy. Therefore, we are able to trade the precision with the storage space in practical applications. If the values of the data streams vary slowly, the wavelet-based models may approximate these streams with acceptable precision. In contrast, the regression-based models will provide better approximations for the fast changing streams. An adaptive version of COD framework is designed to address this issue by making a selection between a wavelet-based model and a regression-based model for building the summary hierarchy. By the adaptive COD, we can obtain clustering results with the same quality as the regression-based COD while using much less storage space for the summary hierarchy. We also conduct a series of experiments on these algorithms to investigate their properties. As shown in the complexity analyses and also validated by our empirical studies, the algorithms of the COD framework perform very efficiently in the data stream environment while producing clustering results of very high quality.

The problem studied in this paper is different from the one discussed in [17], which focuses on clustering the windows of a single streaming time series. On the other hand, the clustering of evolving streams is also discussed in [18]. However, the objective in [18] is to continuously report clusters satisfying the specified distance threshold. The clusters are generated according to the values from the beginning of the stream to the current time. Therefore, it does not have the ability to observe clusters in a time range of interest. To further enhance these techniques, clustering requests of flexible time ranges are supported in our framework.

The rest of this paper is organized as follows: Preliminaries and advantages of the COD framework are described in Section 2. The online maintenance phase and the offline clustering phase of the COD framework are presented in Section 3. Then, empirical studies are conducted in Section 4. This paper concludes with Section 5.

2 Clustering on Demand for Multiple Data Streams

The definitions used in the COD framework are provided in Section 2.1. In addition, the advantages of the COD framework, namely, clustering with flexible window sizes and trend identification, are described in Section 2.2.

2.1 Framework Definitions

The COD framework has two phases, i.e., the *online maintenance phase* and the *offline clustering phase*. In the online maintenance phase, data streams are processed and only very brief summaries are maintained. The offline clustering phase deals with the clustering queries. The COD framework supports clustering queries with a flexible window size and a desired number of windows to observe. For example, a clustering query could be 12 windows with the window size of 30 days to observe the clusters of each

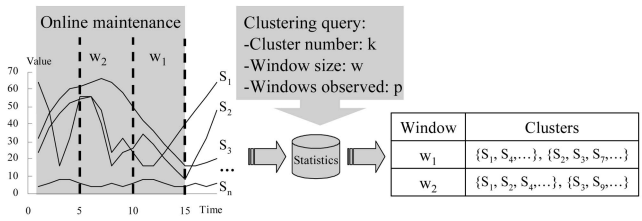


Fig. 1. Clustering of multiple data streams by COD at $t_{now} = 15$, for a query of $k = 2$, $w = 5$ and $p = 2$.

month during a year. Based on the limited space property in the data stream environment, the raw data streams are parsed only once and then discarded. Therefore, the clustering algorithm in our framework is applied to the statistics maintained by the online phase rather than to the original streams, as illustrated in Fig. 1.

At any time stamp, each stream receives a new value simultaneously, and there are totally n data streams. More specifically, we have the n streams $\{S_1, S_2, \ldots, S_n\}$ at time stamp m where $S_i = \{f_i^1, f_i^2, \ldots, f_i^m\}$, for $1 \leq i \leq n$, and f_i^t is the value of stream S_i that has arrived at time t.

2.1.1 The Offline Clustering Phase

Let k denote the number of clusters, and let w be the window size of the clustering query submitted at time stamp t_{now}. The algorithm proposed will generate at most p windows of k-clustering results where

$$Cl(w_i) = \{C_1(w_i), C_2(w_i), \ldots, C_k(w_i)\},$$

for $1 \leq i \leq p$, which minimizes each clustering cost $Cost(Cl(w_i))$ of the substreams in the interval

$$(t_{now} - w \times i, t_{now} - w \times (i-1)].$$

Note that $C_j(w_i)$ is the jth cluster of window w_i with the properties of

$$\bigcap_{j=1}^{k} C_j(w_i) = \phi$$

and $\bigcup_{j=1}^{k} C_j(w_i) = \{S_1(w_i), S_2(w_i), \ldots, S_n(w_i)\}$, where

$$S_q(w_i) = \{f_q^{(t_{now}-w\times i)+1}, f_q^{(t_{now}-w\times i)+2}, \ldots, f_q^{(t_{now}-w\times i)+w}\},$$

for $1 \leq q \leq n$.

Example 1. Consider the first three data streams $\{S_1, S_2, S_3\}$ in Fig. 1 at time stamp $t_{now} = 15$,

$$S_1 = \{64, 48, 16, 32, 56, 56, 48, 24, 32, 24, 16, 16, 24, 32, 40\},$$
$$S_2 = \{24, 38, 46, 52, 54, 56, 40, 16, 24, 26, 34, 28, 20, 14, 8\},$$
and
$$S_3 = \{32, 46, 54, 60, 62, 64, 66, 64, 58, 50, 42, 36, 28, 22, 16\}.$$

Assume that the clustering query is $k = 2$, $w = 5$, and $p = 2$. The algorithm will generate at most 2 windows of 2-clustering results, $Cl(w_1) = \{C_1(w_1), C_2(w_1)\}$ and $Cl(w_2) = \{C_1(w_2), C_2(w_2)\}$. Note that the resulting clusters of window w_1 are $C_1(w_1) = \{S_1\}$ and $C_2(w_1) = \{S_2, S_3\}$ since S_2 is more similar to S_3 in the interval $(15 - 5 \times 1, 15 - 5 \times (1-1)] = (10, 15]$. In window w_2, the clusters are $C_1(w_2) = \{S_1, S_2\}$ and $C_2(w_2) = \{S_3\}$ since S_2 is more similar to S_1 in the interval $(15 - 5 \times 2, 15 - 5 \times (2-1)] = (5, 10]$.

2.1.2 The Online Maintenance Phase

To support various clustering queries in the offline clustering phase, adequate information has to be preserved during the online maintenance phase for discovering fast and slowly evolving patterns. Note that the resolution of statistics maintained could affect the patterns obtained. With a small interval used for summarization, short-term patterns can be observed, but long-term patterns are likely to be neglected. Also, the summaries are possibly affected by noises and oscillations, and the summaries should be updated or reconstructed frequently. On the other hand, if a large interval is used, long-term patterns can be observed while neglecting short-term patterns. Also, the patterns may not catch up with the changes in data streams. Moreover, the patterns could be very rough because of the generalization/summarization of streams. Therefore, we devise a hierarchical structure to store data summaries at different resolutions in the online maintenance phase. Whenever a clustering query is submitted, the offline clustering algorithm will select the approximations at the appropriate levels of the summary hierarchies to support the requirements of the clustering query. The most recent data points are in the first (latest) window and are always approximated by the most accurate fitting models, and new windows are in general more accurate than older ones. Therefore, recent data will not suffer the problem of roughness. Details of the online and offline phases will be described in following sections.

2.2 Advantages of the COD Framework

In this section, we present two advantages of the COD framework, namely, clustering with flexible window sizes and trend detection, which are, in our view, not only of practical importance, but also absent in conventional methods.

2.2.1 Clustering with Flexible Window Sizes

Since the summaries maintained in the summary hierarchies are at multiple resolutions, the COD framework is able to support the clusters for variable window sizes. Applying the conventional time series clustering algorithms to the raw streams with the desired window sizes is not practical in the streaming environment because there is not enough space for the storage of continuous streams. On the other hand, the summarization techniques which maintain the streams with a fixed resolution do not perform well for various window sizes. Although the prior work [18] for clustering of evolving streams continuously reports clusters within the given distance threshold, it does not allow the user to specify a desired window size to be observed. In our COD framework, even though the data streams are collected and summarized into the summary hierarchies before the clustering queries are submitted, the clusters with various window sizes can be obtained directly from the existing summary hierarchies without parsing the data streams again to construct the summaries for the desired

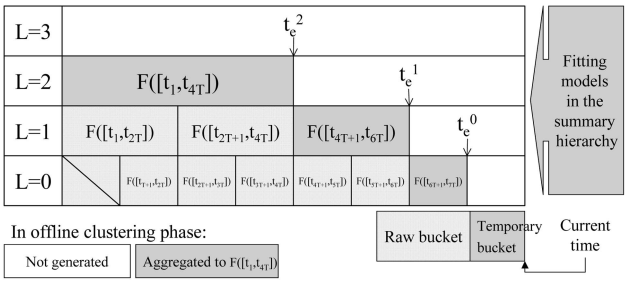


Fig. 2. The illustration of summary hierarchy, where $F(h)$ represents the fitting model in the time interval $h = [t_s, t_e]$. The parameters are set as $B_t = T,\ B_h = 2$, and $\alpha = 6$.

resolutions. For example, suppose that the summary hierarchies of the stock prices have been kept for the prices in 10 years. Then, clusters in one day, one month, or even one year can be observed very efficiently without resorting to the old prices again.

2.2.2 Trend Identification and Change Detection

From the definition of clustering query, at most p windows of clustering results can be obtained for a query. It is very efficient to observe the trends and changes of clusters at one time. The behavior of the clusters, such as the moving paths of clusters, the merges and splits of clusters, and the streams jumping between clusters, can be investigated from the results of a clustering query. Consider the example of stock prices again. Assume that the window size is one month and 12 windows are inspected. We might find out that stock A and stock B are in the same cluster for one month and then stock A jumps to another cluster for the following several months. Such trends and changes within one year can be extended from the results of this clustering query.

3 The Framework of COD

We present the details of the online maintenance phase and the offline clustering phase of COD in Sections 3.1 and 3.2, respectively. Then, we analyze the complexity of the proposed approaches in Section 3.3. Finally, we make some remarks on the adaptive use of COD in Section 3.4.

3.1 Online Maintenance Phase

The main objective of this online maintenance phase is to provide a one scan algorithm of the incoming multiple data streams for statistics collection. A summary hierarchy is maintained incrementally to provide multiresolution approximations for a stream. Multiple levels in the hierarchy correspond to various resolutions.

Definition 1. *A **fitting model** $F(h)$ is defined as an approximation of a raw substream in the interval $h = [t_s, t_e]$ by some summarization techniques. The fitting model on level L is generated by the aggregation of B_h models on level $(L-1)$. A level, say level L, also keeps the time stamp of the latest data point, which has been summarized to that level, as the end time t_e^L of the level.*

Fig. 2 illustrates an example of the summary hierarchy with the parameters $B_t = T$, $B_h = 2$, and $\alpha = 6$. The procedure of generating and updating the summary hierarchy is described in Fig. 3. According to a procedure of the online maintenance phase, each incoming value is put in the temporary bucket first. A new model of level 0 will be generated after B_t values have arrived. At the same time, the end time t_e^0 of level 0 should also be updated. Then, these B_t values are moved from the temporary bucket to the raw bucket, in which the latest α values of the raw stream are maintained. The values in the raw bucket are maintained for queries with very short window sizes, which are smaller than the span of the fitting models in the lowest level. Since they are the newest values in data streams, we would like them to be as precise as possible, i.e., without approximation, when answering clustering queries. When B_h new models are accumulated in level L, a new model of level $(L+1)$ will be generated by aggregating the latest B_h models in level L, and the end time t_e^{L+1} of level $(L+1)$ is also updated.

Procedure of the online maintenance phase (for one stream)

1. For each incoming value, put it in the temporary bucket.
2. If the number of items in the temporary bucket is less than the bucket size B_t, go to Step 1. Else:
   - 2.1 A new fitting model of level 0 is generated according to the values in the temporary bucket.
   - 2.2 Update the end time t_e^0 of level 0.
   - 2.3 Move all the items from the temporary bucket to the raw bucket.
   - 2.4 If the number of items in the raw bucket is more than α, remove the oldest items to keep the number not larger than α.
3. For each level L, if B_h new models are accumulated:
   - 3.1 A new model of level $(L+1)$ will be generated by aggregating the latest B_h models in level L.
   - 3.2 Update the end time t_e^{L+1} of level $(L+1)$.
   - 3.3 If the number of models in level $(L+1)$ is larger than α, remove the oldest model from that level.

Fig. 3. The outlines of procedure of the online maintenance phase.

A summary hierarchy is established for each data stream. Therefore, totally n hierarchies are maintained for n streams. As shown in the procedure, to achieve the space limitation in the streaming environment, only the latest α models are maintained in each level. Note the α should be set to a number not smaller than B_h in order to have enough fitting models for the model generation in a higher level. To capture approximate historical details of data streams with limited resources, various existing techniques for data compression are examined. Among several approaches, the method of wavelet-based transform [19], [20] and regression-based analysis [21], [22], which are selected as our fitting models, are found to be advantageous for their linear computation complexity and the graceful approximation with only a few coefficients.

3.1.1 Wavelet-Based Fitting Model

While the theory of wavelet is extensive, we conform ourselves to the rudimentary wavelet transform in this paper. Namely, the Haar wavelet basis function, which is the simplest wavelet, is explored and utilized for our purposes. In the rest of the paper, we will assume that Haar wavelets are used and a single coefficient representing the average is maintained as a fitting model. This is similar to the strategy of coefficient selection in [3]. To generalize the idea to averaging arbitrary number of values, we have the following definition.

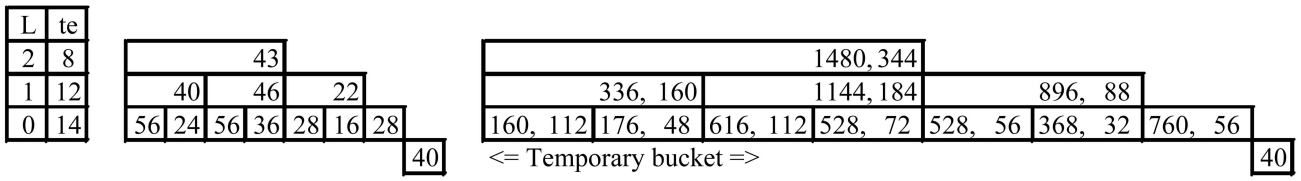


Fig. 4. Example of summary hierarchies. (a) Wavelet-based and (b) regression-based.

Definition 2. *A wavelet-based fitting model in an interval h is represented by*

$$W(h) = \frac{\sum_h f}{|h|}.$$

Lemma 1. *The lossless* aggregation *of q wavelet-based fitting models is denoted by*

$$W^q = \frac{\left(\sum_{i=1}^{q} W(h_i) \times |h_i|\right)}{\left(\sum_{i=1}^{q} |h_i|\right)},$$

where q successive models are merged together.

The concept of a wavelet-based model can be best understood by the following example.

Example 2. The scenario of $B_t = T$, $B_h = 2$, and $\alpha = 6$ is shown in Fig. 2. A model of level 0 is generated from T arrivals, and two new models accumulated in level L will be aggregated into a new model of level $(L+1)$. Also, at most six latest models can be maintained in each level of the hierarchy. Let $B_t = 2$ and consider stream S_1 of Example 1 again. We have $\langle \frac{64+48}{2} \rangle = \langle 56 \rangle$ at time stamp 2 and $\langle \frac{16+32}{2} \rangle = \langle 24 \rangle$ at time stamp 4 for level 0. A fitting model of level 1 is produced by averaging two models of level 0 and, hence, $\langle \frac{56+24}{2} \rangle = \langle 40 \rangle$ is also inserted into level 1 at time stamp 4. The first fitting model of level 2, $\langle \frac{40+46}{2} \rangle = \langle 43 \rangle$, is generated at time stamp 8. By repeating this process, the wavelet-based summary hierarchy can be obtained as in Fig. 4a. Remember that at most six models can be kept in each level. Therefore, when the seventh model $\langle \frac{24+32}{2} \rangle = \langle 28 \rangle$ is generated in level 0 at time stamp 14, the oldest model of that level, which is $\langle 56 \rangle$, will be removed. The latest value arriving at time stamp 15 is still in the temporary bucket waiting to be merged with the future values. The end time t_e^L of each level will be the t_e of the latest model in that level, which is 14, 12, and 8 for level 0, 1, and 2, respectively.

3.1.2 Regression-Based Fitting Model

To meet the space constraint in a data stream environment, we devise a compact stream representation to comprise all the information required for the regression-based approximations. Consequently, only required synopses rather than historical details of data streams are maintained during our discovering process.

A straight-line fit for a stream is a linear estimation function $\widehat{f} = \widehat{\theta} + \widehat{\eta} t$ that conforms to the principle of least squares. Specifically, the regression parameters $\widehat{\theta}$ and $\widehat{\eta}$ are chosen to make the residual sum of squares

$$D = \sum_{t=1}^{n} (f_t - \widehat{\theta} - \widehat{\eta} t)^2$$

minimal, where f_t is the actual value of the tth data point. To perform the calculations for getting best estimates of $\widehat{\theta}$ and $\widehat{\eta}$, the following quantities are maintained, $\bar{t} = \frac{1}{n}\sum t$, $\bar{f} = \frac{1}{n}\sum f$, $S_{tt} = \sum (t - \bar{t})^2 = \sum t^2 - \frac{(\sum t)^2}{n}$, and

$$S_{tf} = \sum (t - \bar{t})(f - \bar{f}) = \sum tf - \frac{(\sum t)(\sum f)}{n}.$$

Then, the least square estimates of $\widehat{\theta}$ and $\widehat{\eta}$ are computed by $\widehat{\eta} = \frac{S_{tf}}{S_{tt}}$, and $\widehat{\theta} = \bar{f} - \widehat{\eta}\bar{t} = \frac{\sum f}{n} - \widehat{\eta} \times \frac{\sum t}{n}$.

Specifically, only three measures are required to maintain during the regression process, i.e., the ending time (t_e), the accumulative product of time and value ($\sum tf$), and the accumulative sum of value ($\sum f$). Consequently, we have the following definition for the regression-based fitting model.

Definition 3. *A regression-based fitting model in an interval h is represented by*

$$R(h) = \left\langle \sum_h tf, \sum_h f \right\rangle.$$

Based on the following lemma, we will prove in Theorem 1 that the least square error linear fit for a substream can be obtained losslessly from the accumulative sum $\langle \sum tf, \sum f \rangle$ and t_e^L of that level. For interest of space, proofs of theorems and lemmas are given in the Appendix.

Lemma 2. *1)* $\sum_{t=t_s}^{t_e} t = \frac{(t_s+t_e)(t_e-t_s+1)}{2}$ *and 2)*

$$\sum_{t=t_s}^{t_e} t^2 = \sum_{t=1}^{t_e} t^2 - \sum_{t=1}^{t_s-1} t^2 = \frac{t_e(t_e+1)(2t_e+1)}{6} - \frac{(t_s-1)(t_s)(2t_s-1)}{6}.$$

Theorem 1. *The least square error linear fit for a substream can be obtained losslessly from the accumulative sum $\langle \sum tf, \sum f \rangle$ and t_e^L of that level.*

Lemma 3. *The lossless* aggregation *of q regression-based fitting models is denoted by*

$$R^q = \left\langle \sum_{i=1}^{q} \left(\sum_{h_i} tf \right), \sum_{i=1}^{q} \left(\sum_{h_i} f \right) \right\rangle,$$

where q successive models are merged together.

Example 3. Consider the stream in Example 2 again. Note that the procedure of generating the regression-based fitting model is the same as that of the wavelet-based one, but the values stored are different. Therefore, we have $\{\langle 160, 112\rangle, \langle 17, 648\rangle\}$ at time stamp 4, where the first two values in the stream, i.e., 64 and 48, generate the accumulative sum

$$\left\langle \sum tf, \sum f \right\rangle = \langle 1 \times 64 + 2 \times 48, 64 + 48\rangle = \langle 160, 112\rangle$$

at time stamp 2, and the second two values 16 and 32 generate the accumulative sum

$$\langle 3 \times 16 + 4 \times 32, 16 + 32\rangle = \langle 176, 48\rangle$$

at time stamp 4. In addition, $\langle 160 + 176, 112 + 48\rangle = \langle 336, 160\rangle$ is also inserted into level 1 at time stamp 4. The first fitting model $\langle 1, 480, 344\rangle$ of level 2, which is the summation of $\langle 336, 160\rangle$ and $\langle 1, 144, 184\rangle$ of level 1, is generated at time stamp 8. By repeating this process, the regression-based summary hierarchy can be obtained as in Fig. 4b. The seventh model $\langle 760, 56\rangle$ generated in level 0 at time stamp 14 also causes the oldest model of that level, i.e., $\langle 160, 112\rangle$, to be removed. The end time t_e^L of each level and the value in the temporary bucket are the same as those of the wavelet-based model.

There are still other possible solutions for estimating data streams, such as adaptive piecewise constant approximation. However, different from the focus of our multi-resolution structure, the adaptive piecewise constant approximation does not provide finer approximation for more recent data and coarser approximation for older data. It will be an interesting research direction for our future works to combine the adaptive piecewise constant approximation in the summary hierarchy.

3.2 Offline Clustering Phase

As the clustering query shown in Section 2.1, the users want to inspect clusters with window size w, and at most p windows will be observed. Note that the window size desired could be different from those maintained in the summary hierarchy. In this situation, we have to select the fitting models from appropriate levels of the hierarchy to approximate the desired windows. We have the following theorems for adaptive level selection.

Theorem 2. *The highest level to approximate a substream with window size w is*

$$L_{max} = \left\lfloor log_{B_h} \left(\frac{w}{B_t} \right) \right\rfloor.$$

Theorem 3. *The lowest level to approximate a substream with window size w is*

$$L_{min} = \min_L \{ w \le (t_{now} - t_e^L) + \alpha_L \times h_L \},$$

where α_L is the exact number of fitting models in level L, and $h_L = B_t \times (B_h)^L$ is the window size of the fitting models in level L.

From the above theorems, fitting models in the levels between L_{min} and L_{max} are able to approximate the substreams in the windows of clustering queries. The fitting models in higher levels span longer intervals and, thus, provide more generalized fitting to the original streams. In contrast, the fitting models in lower levels possess shorter spans and can thus provide more specific and accurate fits to the original streams. However, only α models are maintained in each level. Therefore, we devise an *adaptive clustering algorithm* for the offline clustering phase to approximate each window of data streams with the fitting models as accurately as possible. The statistics to characterize a window of stream is *encapsulated* as an *entry*, which is described in Definition 4. Note that each entry obtained from the hierarchy may include more than one fitting model. As shown in the procedure of the adaptive clustering algorithm, the clustering algorithm is executed according to the entries obtained from the fitting models in the hierarchy.

Definition 4. *An* entry $E_A(w_i)$ *that* encapsulates j *fitting models to represent window w_i of stream S_A is expressed as:*

a. $\{\langle W_A(h_1), |h_1|\rangle, \langle W_A(h_2), |h_2|\rangle, \ldots, \langle W_A(h_j), |h_j|\rangle\}$ *for a wavelet-based model, where h_q represents $[t_s, t_e]$ of the qth fitting model in the entry and*

b. $$\left\{ \left\langle \widehat{\theta}_A(h_1), \widehat{\eta}_A(h_1), h_1 \right\rangle, \left\langle \widehat{\theta}_A(h_2), \widehat{\eta}_A(h_2), h_2 \right\rangle, \ldots, \left\langle \widehat{\theta}_A(h_j), \widehat{\eta}_A(h_j), h_j \right\rangle \right\}$$

for a regression-based model, where $\widehat{\theta}_A(h_d)$ and $\widehat{\eta}_A(h_d)$ for $1 \le d \le j$ are calculated from the corresponding

$$R_A(h_d) = \left\langle \sum_{h_d} tf, \sum_{h_d} f \right\rangle$$

by the equations derived in Section 3.1.2.

The outlines of the adaptive clustering algorithm are shown in Fig. 5. In the first step of the adaptive clustering algorithm, the values of L_{min} and L_{max} are calculated according to the window size w specified by the clustering request. Then, for each data stream, the fitting models, which are covered by the ranges of the desired windows, are retrieved in Step 2 and Step 3. Finally, for each window, the clustering algorithm is executed on the retrieved statistics. A typical cost function for a window of size w is

$$Cost(Cl(w_i)) = \frac{\sum_{C_j(w_i)} \sum_{S_q(w_i) \in C_j(w_i)} dist\big(S_q(w_i) - C_j(w_i).center\big)}{n \times w},$$

Procedure of adaptive clustering algorithm

1. Calculate L_{min} and L_{max}. For each data stream, do Steps 2 and 3.
2. If the end time $t_e^{L_{min}}$ of level L_{min} is not equal to the current time, aggregate the models of lower levels (from L_{min} -1 to 0) and the temporary bucket to generate a temporary model characterizing the interval between $t_e^{L_{min}}$ and the current time. Then, aggregate this temporary model to the latest model in level L_{min}.
3. Encapsulate the fitting models between level L_{min} and L_{max} to generate at most p entries, where each entry represents a window with size w. Set $L = L_{min}$ initially. For the windows from w_1 to w_p, if the range of a desired window is covered by the interval of the fitting models in level L, encapsulate an appropriate number of fitting models into that entry. Else, increase L by one to look for the fitting models with enough coverage. This step stops when p entries have been retrieved or when L exceeds the maximum level L_{max} with $p_r \le p$.
4. Run the clustering algorithm to cluster these sub-streams by the retrieved entries for each window.

Fig. 5. The outlines of the adaptive clustering algorithm.

$$dist\big(S_q(w_i) - C_j(w_i).center\big) = \sum_{f_q}^{t} \in S_q(w_i), f_{C_j}^t \in C_j(w_i).center\Big(f_q^t - f_{C_j}^t\Big)^2,$$

where $C_j(w_i).center$ is the center of cluster $C_j(w_i)$, and $f_{C_j}^t$ are the values of this cluster center in the range of w_i. If more than one window is observed, the total cost will be the average clustering cost of the retrieved windows, as shown in the following equation:

$$Cost(Cl) = \frac{\sum_{i=1}^{p_r} \rho_i Cost(Cl(w_i))}{p_r}, \quad (1)$$

where p_r is the number of windows actually retrieved and ρ_i is the weight of the clustering cost for window w_i. In some applications, users are more interested in the recent data and give higher weights for the costs of more recent windows. Without loss of generality, this cost function will be used to evaluate the clustering results in our performance studies in sections later. Based on the definitions of the entry, the distance measurement between two substreams can be derived by Lemma 4.

Lemma 4. *The* distance measurement D_i *between two substreams* $S_A(w_i)$ *and* $S_B(w_i)$ *of window* w_i*, i.e., between entries* $E_A(w_i)$ *and* $E_B(w_i)$*, is:*

a.

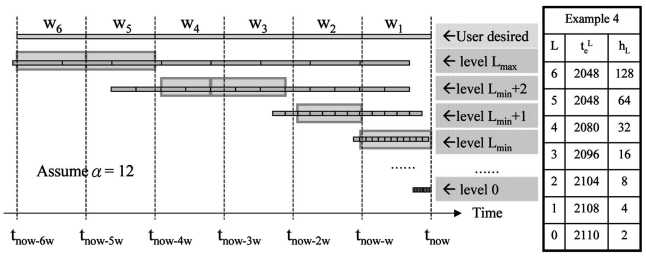
$$\sum_{d=1}^{j}(W_A(h_d) - W_B(h_d))^2 \times |h_d|$$

for a wavelet-based model and

b.

$$\sum_{d=1}^{j}\left[\Big(\widehat{\theta}_A(h_d) - \widehat{\theta}_B(h_d)\Big)^2|h_d| + 2\Big(\widehat{\theta}_A(h_d) - \widehat{\theta}_B(h_d)\Big) \times (\widehat{\eta}_A(h_d) - \widehat{\eta}_B(h_d)) \sum_{h_d} t + (\widehat{\eta}_A(h_d) - \widehat{\eta}_B(h_d))^2 \sum_{h_d} t^2\right]$$

for a regression-based model, where $\widehat{f}_A(h_d) = \widehat{\theta}_A(h_d) + \widehat{\eta}_A(h_d)t$ *and* $\widehat{f}_B(h_d) = \widehat{\theta}_B(h_d) + \widehat{\eta}_B(h_d)t$.



Example 4

L	t_e^L	h_L
6	2048	128
5	2048	64
4	2080	32
3	2096	16
2	2104	8
1	2108	4
0	2110	2

Fig. 6. An illustration of adaptive clustering. Assume that $\alpha = 12$, $B_t = B_h = 2$, $w = 180$, $p = 6$, and $t_{now} = 2,110$.

Let us consider Step 2 of a procedure of the adaptive clustering algorithm. Assume that level $L_{min} = 2$, as shown in Fig. 2. However, since the end time t_e^2 of level 2 is not equal to the current time, we aggregate the models by the lower levels (from level $2-1$ to 0) and the temporary bucket to generate a temporary model characterizing the interval between t_e^2 and the current time. Then, it is aggregated into the latest model in level 2.

The following lemma describes the interpolation method to obtain a fraction of a fitting model.

Lemma 5. *Let* t_i *be a time stamp between* $h = [t_s, t_e]$ *of a fitting model* $W(h)$ *(or* $R(h)$*, respectively). The* interpolation *of the fitting model between* $h' = [t_i, t_e]$ *is:*

a. $W(h') = W(h)$ *for a wavelet-based model and*

b.

$$R(h') = \left\langle \left(\frac{1}{2}\hat{\theta}((t_e)^2 - (t_i)^2) + \frac{1}{2}\hat{\eta}((t_e)^3 - (t_i)^3)\right), \left(\hat{\theta}(t_e - t_i) + \frac{1}{2}\hat{\eta}((t_e)^2 - (t_i)^2)\right)\right\rangle$$

for a regression-based model, where $\widehat{f} = \widehat{\theta} + \widehat{\eta}t$ *is the estimation function of* $R(h)$.

The details of Step 3 are described as follows: We can in fact improve the clustering results by approximating a window with more fitting models in lower levels of the hierarchy. Note that interpolation is possible to bring some distortions into the fitting model. Therefore, we use interpolation only at the highest level L_{max} for each clustering query. An entry, which encapsulates several *whole* fitting models from a level *below* L_{max}, is *approximated* to the window size w in order to avoid the distortion from interpolation. More specifically, a fitting model is inserted into an entry if at least half of its interval is in the range of that window. Note that due to the space limitation of the data stream environment, at most α models are maintained in each level. Therefore, the number of models in lower levels is usually not enough for all the windows queried. In this situation, some fitting models in level L_{max} are aggregated and interpolated for an entry. Each entry generated from the models in level L_{max} is *exactly* with window size w.

Example 4. Consider summary hierarchies in Fig. 6 as an example. Assume that $t_{now} = 2,110$, $\alpha = 12$, $B_t = B_h = 2$, and the clustering query indicates that $w = 180$ and

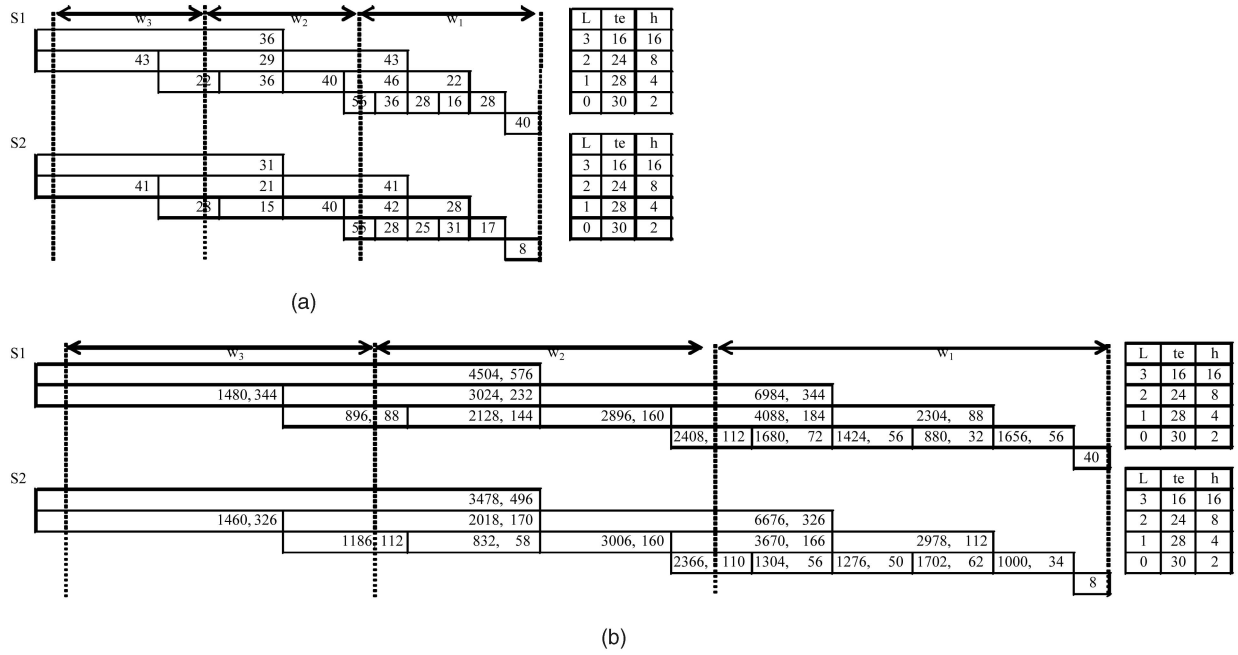


Fig. 7. Example of applying the adaptive clustering algorithm on summary hierarchies of two streams S_1 and S_2, where $\alpha = 5$, $B_t = B_h = 2$, $w = 10$, $p = 3$, and $t_{now} = 31$. (a) Wavelet-based and (b) regression-based.

$p = 6$. According to Theorem 2 and Theorem 3, $L_{min} = 3$ because

$$(t_{now} - t_e^L) + \alpha_L \times h_L = (2110 - 2096) + 12 \times 16 = 206 \geq 180$$

for $L = 3$, and

$$(t_{now} - t_e^L) + \alpha_L \times h_L = (2,110 - 2,104) + 12 \times 8 = 102 < 180$$

for $L = 2$.

$$L_{max} = \left\lfloor log_{B_h}\left(\frac{w}{B_t}\right)\right\rfloor = \left\lfloor log_2\left(\frac{180}{2}\right)\right\rfloor = 6.$$

Since the end time $t_e^{L\min} = 2,096$ of level $L_{min} = 3$ is not equal to the current time $t_{now} = 2,110$, the models of lower levels (from level 2 to 0) are aggregated to the latest model in level L_{min}. Then, 11 fitting models in level 3 are encapsulated in the entry for window w_1, five fitting models in level 4 are encapsulated in the entry for window w_2, three fitting models in level 5 are encapsulated in the entry for window w_3, and 2 fitting models in level 5 are encapsulated in the entry for window w_4. Note that a fitting model is inserted into an entry if at least half of its interval is in the range of that window. As shown in Fig. 6, the entries representing windows w_1 to w_4 are approximated to the window size w by several whole fitting models (without interpolation) from level L_{min} to $L_{min} + 2$. Older windows such as windows w_5 and w_6 are not able to be approximated by several small models. Therefore, we have to aggregate and interpolate fitting models in level $L_{max} = 6$ to generate entries exactly with window size w.

Assume that the interval covered by an entry is w'. We observe that the difference between w' and the window size w of the clustering query is limited in $\frac{w}{B_h}$, which can be verified by the following lemma.

Lemma 6. *The difference between the interval of the retrieved entry, denoted as w', and the window size w of the clustering query is limited in $\frac{w}{B_h}$, i.e., $|w\prime - w| \leq \frac{w}{B_h}$.*

Example 5. Consider summary hierarchies in Fig. 7 as an example. Assume that $t_{now} = 31$, $\alpha = 5$, $B_t = B_h = 2$, and the clustering query indicates that $w = 10$, and $p = 3$. According to Theorem 2 and Theorem 3, $L_{min} = 0$ because $(t_{now} - t_e^L) + \alpha_L \times h_L = (31 - 30) + 5 \times 2 = 11 \geq 10$ for $L = 0$ and $L_{max} = \left\lfloor log_{B_h}(\frac{w}{B_t})\right\rfloor = \left\lfloor log_2(\frac{10}{2})\right\rfloor = 2$. As shown as follows, these three windows of streams S_1 and S_2 can be approximated by entries $E_1(w_1)$, $E_1(w_2)$, $E_1(w_3)$, and $E_2(w_1), E_2(w_2), E_2(w_3)$, respectively, according to Definition 4.

a. Wavelet-based: Entry

$$E_1(w_1) = \Big\{\langle 56, 2\rangle, \langle 36, 2\rangle, \langle 28, 2\rangle, \langle 16, 2\rangle, \left\langle \frac{28 * 2 + 40 * 1}{2 + 1}, (2 + 1)\right\rangle\Big\} = \{\langle 56, 2\rangle, \langle 36, 2\rangle, \langle 28, 2\rangle, \langle 16, 2\rangle, \langle 32, 3\rangle\}$$

is obtained from the temporary bucket and level 0, entry $E_1(w_2) = \{\langle 36, 4\rangle, \langle 40, 4\rangle\}$ is obtained from level 1, and entry $E_1(w_3) = \{\langle \frac{43*7+29*3}{10}, 10\rangle\} = \{\langle 38.8, 10\rangle\}$ is interpolated and aggregated by fitting models in level 2. Then, entries

$$E_2(w_1) = \{\langle 55, 2\rangle, \langle 28, 2\rangle, \langle 25, 2\rangle, \langle 31, 2\rangle, \langle 14, 3\rangle\}$$
$$E_2(w_2) = \{\langle 14.5, 4\rangle, \langle 40, 4\rangle\},$$

and $E_2(w_3) = \{\langle 34.95, 10\rangle\}$ of stream S_2 can be obtained similarly. The distance between w_1 of streams S_1 and S_2 will be

$$D_1 = \sum_{d=1}^{5} (W_1(h_d) - W_2(h_d))^2 \times |h_d|$$
$$= (56-55)^2 \times 2 + (36-28)^2 \times 2 + (28-25)^2 \times 2 + (16-31)^2 \times 2 + (32-14)^2 \times 3 = 1,570.$$

D_2 and D_3 can be calculated similarly.

b. Regression-based: Entry

$$E_1(w_1) = \{\langle 56, 0, [21, 22]\rangle, \langle 600, -24, [23, 24]\rangle, \langle 232, -8, [25, 26]\rangle, \langle 16, 0, [27, 28]\rangle, \langle -208, 8, [29, 31]\rangle\}.$$

Note that $\widehat{\eta}_1(h_1) = \frac{S_{tf}}{S_{tt}} = \frac{0}{0.5} = 0$ and $\widehat{\theta}_1(h_1) = \bar{f} - \widehat{\eta}_1(h_1)\bar{t} = 56 - 0 \times 21.5 = 56$ can be calculated from

$$S_{tf} = \sum tf - \frac{(\sum t)(\sum f)}{n} = 2,408 - \frac{(21+22) \times 112}{2} = 0$$

and

$$S_{tt} = \sum t^2 - \frac{(\sum t)^2}{n} = (21^2 + 22^2) - \frac{(21+22)^2}{2} = 925 - 924.5 = 0.5.$$

Similarly, $\widehat{\eta}_1(h_2) = \frac{-12}{0.5} = -24$,

$$\widehat{\theta}_1(h_2) = 36 - (-24) \times 23.5 = 600,$$

and so on. Entry

$$E_1(w_2) = \{\langle -80, 8, [13, 16]\rangle, \langle 276.8, -12.8, [17, 20]\rangle\}$$

is obtained from level 1, and entry $E_1(w_3) = \{\langle 59.492, -3.41049, [2, 11]\rangle\}$ is interpolated and aggregated by fitting models in level 2. Then, entries

$$E_2(w_1) = \{\langle 12, 2, [21, 22]\rangle, \langle 592, -24, [23, 24]\rangle, \langle -26, 2, [25, 26]\rangle, \langle 196, -6, [27, 28]\rangle, \langle 194, -6, [29, 31]\rangle\},$$

$$E_2(w_2) = \{\langle 40.6, -1.8, [13, 16]\rangle, \langle -130.2, 9.2, [17, 20]\rangle\},$$

$E_2(w_3) = \{\langle 59.0467, -3.5189, [2, 11]\rangle\}$ of stream S_2 can be obtained similarly. The distance between w_1 of streams S_1 and S_2 will be $D_1 = 2,032$ according to Lemma 4(b). D_2 and D_3 can be calculated similarly.

Based on adaptive clustering algorithm, we can also observe clustering patterns in the intervals prior to the current time with slight modification on the algorithm. In other words, we are able to observe p windows of clustering results starting from t_{before} rather than from t_{now}, i.e., window w_i is in the interval $(t_{before} - w \times i, t_{before} - w \times (i-1)]$ rather than in $(t_{now} - w \times i, t_{now} - w \times (i-1)]$. Because the newest window to be observed is in the interval $(t_{before} - w, t_{before}]$, the lowest level to approximate a substream with window size w will become

$$L_{min} = \min_L \{(t_{now} - t_{before}) + w \le (t_{now} - t_e^L) + \alpha_L \times h_L\},$$

which is the lowest level whose fitting models are still enough to illustrate the pattern within the interval $(t_{before} - w, t_{before}]$. In the original design of the adaptive clustering algorithm, the range of the clustering query starts from t_{now}. However, in order to support the clustering query for arbitrary historical ranges, Step 2 of the procedure of adaptive clustering algorithm needs a little modification to handle the newest window w_1. Step 2 is modified as follows:

2. If the end time $t_e^{L_{min}}$ of level L_{min} is smaller than t_{before}, aggregate the models of lower levels and the temporary bucket (if needed) to generate a temporary model characterizing the interval between $t_e^{L_{min}}$ and t_{before}. Then, aggregate this temporary model to the latest model in level L_{min}.

With these modifications mentioned above, we can observe clustering patterns in any interval. Note that the offline clustering phase is not designed for a specific clustering algorithm. Therefore, users can adopt any traditional clustering algorithm with minor modification if so necessary. Without loss of generality, the *k-means clustering algorithm* [23] is applied in the offline clustering phase for the complexity analysis and performance evaluation.

3.3 Complexity Analyses

The complexities of the online and offline phases of the COD framework are shown in the following theorems, where n is the number of streams and m is the number of data points in each stream.

Theorem 4. *The time complexity of the online maintenance phase is* $O\big(n\big(m + \frac{m}{B_t}\big)\big)$.

Theorem 5. *The space complexity of the online maintenance phase is* $O\big(n\big(\alpha log_{B_h}(\frac{m}{B_t})\big)\big)$.

Theorem 6. *The time complexity of the offline clustering phase is* $O(kn\alpha)$ *for a window.*

According to Theorem 5, the total number of fitting models maintained is relatively modest. For example, suppose that $B_h = B_h = 2$ and $\alpha = 20$, for a data stream running for 100 years with a clock time granularity of 1 second, the total number of fitting models maintained for a stream is at most $20 \times log_2(100 \times 365 \times 24 \times 60 \times 60/2) \approx 600$. This is quite a modest storage requirement. Furthermore, the dimension of the data for clustering is usually highly reduced by the summarization techniques. Thus, the COD framework is very efficient in the clustering phase.

3.4 Adaptive Use of COD

The framework COD proposed can be further extended to an adaptive version by integrating the wavelet-based and

the regression-based models together. Although the wavelet-based COD and the regression-based COD have the same complexities in time and space, we can still trade the precision with the storage space in practical applications. If the values of the data streams change slowly, a wavelet-based model may approximate these streams with acceptable precision while achieving shorter maintaining time and smaller storage space. Also, the regression-based model will provide better approximations for the faster changing data streams with a bit longer maintaining time and twice of the storage space. Therefore, it will be efficient in the usage of space to apply a wavelet-based model when the stream is stable and to employ a regression-based model as the stream changes dramatically. Based on the temporal locality, we may expect that the behavior of a stream is similar in a nearby interval. The following criteria provide guidelines for selecting fitting models adaptively according to the current behavior of a stream.

Criterion 1. If the slope of a regression-based model does not exceed a threshold δ, i.e., $|\hat{\eta}| \leq \delta$, start to apply a wavelet-based model.

Criterion 2. If the variation of a wavelet-based model is larger than a threshold δ, i.e., $\left|\frac{W(h_i)-W(h_{i-1})}{|h|}\right| > \delta$, where $|h| = |h_i| = |h_{i-1}|$, start to employ a regression-based model. Note that if the previous one is a regression-based model, the value of $W(h_{i-1})$ can be calculated from the second component of

$$R(h_{i-1}) = \left\langle \sum_{h_{i-1}} tf, \sum_{h_{i-1}} f \right\rangle$$

by

$$W(h_{i-1}) = \frac{\sum_{h_{i-1}} f}{|h_{i-1}|}.$$

Since a wavelet-based model can be obtained from a regression-based model by

$$W(h) = \frac{\sum_{h} f}{|h|},$$

and a regression-based model can be approximated from a wavelet-based model by

$$R(h) = \left\langle \sum_{h} tf, \sum_{h} f \right\rangle = \left\langle W(h) \times \sum_{h} t, W(h) \times |h| \right\rangle,$$

all the theorems and lemmas are still applicable to the adaptive COD. The fitting model in a higher level of the summary hierarchy is selected adaptively according to the following criterion.

Criterion 3. If the slope of the aggregated model is larger than a threshold δ, a regression-based model is maintained. Otherwise, a wavelet-based model is produced.

Although a wavelet model can be constructed exactly from a regression model, a regression model requires twice of the space used by a wavelet model. Therefore, if the storage space is very limited, users can take advantage of the adaptive COD to store steady data in wavelet models, which spend less space, without having much influence on the clustering quality. Consequently, we can trade the precision with the storage space adaptively according to the behavior of a stream at present.

4 Experimental Results

To assess the performance of our framework, we have conducted a series of experiments. In Section 4.1, we introduce the simulation model. Next, we investigate the sensitivity analyses of parameters in Section 4.2. Then, we perform the adaptivity analyses of COD framework in Section 4.3. After that, the performance of the offline clustering phase of COD framework on the real data set is evaluated in Section 4.4. Finally, we examine the scalability of the online maintenance phase of the COD framework in Section 4.5.

4.1 Simulation Model

All of our experiments are conducted on a PC with Intel Pentium III processor and 512 MB memory, which runs Window XP professional operating system. In order to evaluate our framework, both the synthetic data set and the real data set are used as our data sets.

1. Fast/Slow changing data: This data is designed to provide different fractions of fast/slow changing data sets for examining the adaptive use of framework COD. For a given fraction of fast data, λ, each series is the concatenation of several segments with length LEN. Samples of the jth segment are generated as follows:

$$g_j(t) = (S + \eta) \times \chi_j(t) + \epsilon(t),$$

where η and $\epsilon(t)$ are drawn from a standard normal distribution $N(0,1)$, t is in the range $[0, LEN-1]$, and $\chi_j(t)$ is decided according to the following rules.

$\chi_j(t)$	$\lambda_j > \lambda$	$\lambda_j \leq \lambda$
$state_j = 0$	0	$\frac{(t \bmod LEN)}{LEN}$
$state_j = 1$	1	$1 - \frac{(t \bmod LEN)}{LEN}$

The slope of the jth segment is determined by λ_j and $state_j$, where λ_j is a value drawn uniformly from $[0,1]$, and $state_j$ is either 0 or 1. Initially, $state_0$ is set to 0. Then, each $state$ value is set according to its value in the previous segment and the current λ_j value. More specifically, $state_j = state_{j-1}$ if $\lambda_j > \lambda$, and $state_j = 1 - state_{j-1}$ if $\lambda_j \leq \lambda$. By this setting, about $(1-\lambda)$ fraction of segments in a series are slowly changing segments with the slopes being close to 0 (for those segments with $\lambda_j > \lambda$). On the other hand, about λ fraction of segments in a series are fast changing segments with the slopes being close to $\pm\frac{S}{LEN}$ (for those segments with $\lambda_j \leq \lambda$). In our experiments, S and LEN are set as 30 and 50, respectively. Therefore, the slopes of fast changing

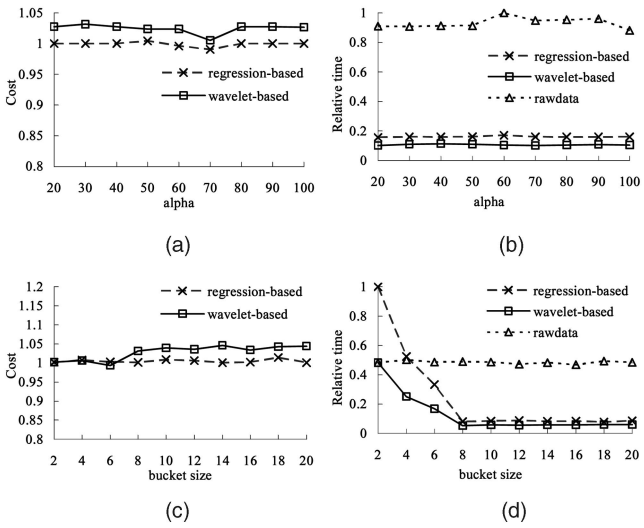


Fig. 8. The execution time and clustering costs of the clustering phase of COD framework. (a) Cost versus alpha. (b) Time versus alpha. (c) Cost versus bucket size. (d) Time versus bucket size.

segments are about ± 0.6. This data is used in our adaptivity analyses, and also supports the need of varying the number of data points and the number of data streams for the scalability analyses.

2. Real weather data: We obtain average daily temperatures of 290 cities around the world from Temperature Data Archive[1] of the University of Dayton. The daily average temperatures of each city are recorded since 1 January 1995 to present. Each city is regarded as a data stream and each stream has 3,416 points.

This work provides a framework to support clustering requests on multiple data streams with flexible time ranges. Unlike that of prior studies [10], [11], [12], [13], the objective in this work is to partition these data streams, rather than their data points, into clusters. On the other hand, although the objective of the study [18] is to partition data streams, it does not support clustering with flexible time ranges. As a result, the problem studied in this paper is intrinsically different from those of prior works. In the following experiments, we use the average squared error as the evaluation function for clustering results, as shown in (1). The costs of COD are measured as the ratio of the costs obtained by running the k-means algorithm on the raw data streams. Because the choice of initial cluster centers in a k-means algorithm affects the quality of results, the best set of clusters from multiple restarts is employed to alleviate this effect [23]. Note that parameter B_t represents the number of values to be merged into a fitting model while parameter B_h represents the number of fitting models to be merged into a new fitting model in a higher level. Both of these two parameters imply approximating several values (or fitting models) with a coarser summary. Therefore, they are set as the same value in our experiments to avoid presenting two experiments with similar effects. Without loss of generality, we assume that the bucket size $B_t = B_h = B$, and all the windows observed are regarded as of equal importance in the following experiments. That is, the weight ρ_i of window w_i is assumed to be one for each window.

1. The real data can be obtained from the Web site: http://www.engr.udayton.edu/weather/.

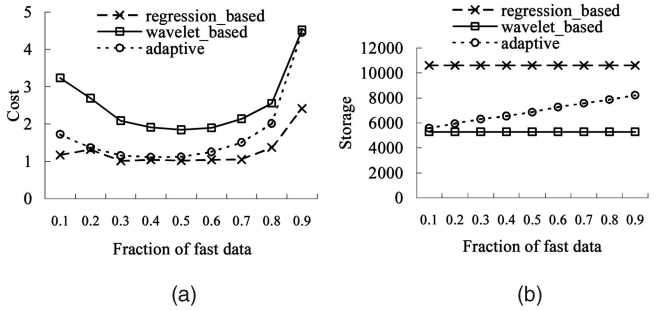


Fig. 9. The clustering costs and storage units of summary hierarchies while varying the fraction of fast data. (a) Cost versus fraction of fast data. (b) Storage versus fraction of fast data.

4.2 Sensitivity Analyses

In this section, two parameters of the summary hierarchy, which are the number of fitting models maintained in each level (α) and bucket size (B), are investigated on the weather data with a fixed window size $w = 50$. In the complexity analysis, the worst case, which retrieves at most α fitting models for clustering, is discussed. Note, however, that one does not encounter the worst case very often in practice. Therefore, the clustering time is mostly dominated by the execution time of the k-means clustering algorithm. As shown in Figs. 8a and 8b, α does not have much influence on the costs and execution time of clustering. Note that since the clustering algorithms are only applied to the statistics maintained, both wavelet-based and regression-based COD are much more efficient than the clustering on the raw data streams, though with slight additional time for calculation from the summary hierarchy. Therefore, when the bucket size is too small, the execution time of framework COD will be larger. As shown in Fig. 8c and Fig. 8d, clustering on the summary hierarchies of framework COD is able to achieve almost the same quality as that on the raw data streams efficiently. Besides, regression-based COD, with slightly longer execution time, is more accurate than wavelet-based COD.

4.3 Adaptivity Analyses

In this section, the fast/slow changing data is used to investigate the performances of proposed methods when varying the fraction of fast data. As described in the simulation model, the fast/slow changing data contain data sets with different fractions of fast data λ. We generate data sets with the fraction of fast data (λ) from 0.1 to 0.9. More specifically, nine data sets are generated, where the value of λ ranges from 0.1 to 0.9. In each data set, 100 streams, each contains 20 segments, were generated. In the data set of $\lambda = 0.1$, each stream contains 10 percent of fast changing segments, in the data set of $\lambda = 0.2$, each stream contains 20 percent of fast changing segments, and so on. The threshold δ is set to be 0.5 in this experiment (because the slope of fast data is about ± 0.6 in the simulation model). As

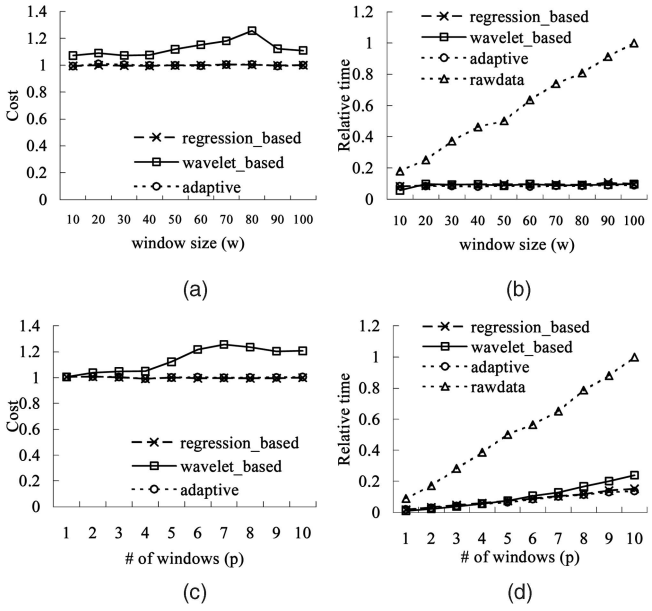


Fig. 10. Performance of the offline clustering phase of COD framework on the real data set. (a) Cost versus window size. (b) Time versus window size. (c) Cost versus windows observed. (d) Time versus windows observed.

shown in Fig. 9a, the clustering quality of the regression-based COD is better than the wavelet-based COD. The adaptive COD, which adaptively selects a wavelet-based model or a regression model for building the summary hierarchy, achieves similar clustering quality as the regression-based COD for data sets with fewer fast data, and the clustering cost is increased when the fraction of fast changing segments increases. We observed that when the data stream changes too fast, it is possible that some changes of a segment are smoothened (regressed) to a line with a small slope and are stored by a wavelet-based fitting model. In this extreme case, the clustering quality of adaptive COD will tend to be similar to the wavelet-based COD. On the other hand, we also investigate the storage space needed to store the summary hierarchies. Note that the wavelet-based hierarchy keeps one value in each model while the regression-based hierarchy keeps two values in each model. We use the number of values stored in a summary hierarchy to approximate the storage space of COD framework. As shown in Fig. 9b, the space needed for the regression-based COD is twice as much as that needed for the wavelet-based COD. Note that the storage space of the adaptive COD is increased along with the increasing amount of fast changing segments. Therefore, when the data sets do not contain too much fast changing data, the adaptive use of framework COD can achieve similar clustering quality as the regression-based COD while spending much smaller storage space for the summary hierarchy.

4.4 COD on Real Data Set

The performance of the COD framework on real data set is next evaluated with a fixed bucket size $B = 12$. As shown in Fig. 10a and Fig. 10c, while varying the window size and the number of windows observed, the framework COD (both of the regression-based COD and the adaptive COD) is able to attain the same good clustering quality as the clustering on the raw data streams, with significantly shorter execution time, as shown in Fig. 10b and Fig. 10d. Note that by maintaining more parameters, the regression-based COD approximates the streams more precisely than the wavelet-based one, and achieves the same quality as that on the raw data streams. In this experiment, we tested several settings of the δ value, and observed that these results are consistent with one another: The regression-based COD is better than the wavelet-based COD, and the adaptive COD can achieve similar quality as the regression-based COD. For these settings, there are segments with slopes larger than the threshold and segments with slopes smaller than the threshold. Without loss of generality, the parameter δ is set as 1, and the storage space for the regression-based COD, the wavelet-based COD and the adaptive COD are 42,920, 21,460, and 22,344, respectively.

It is interesting to see that the adaptive COD attains the same good clustering quality as the regression-based COD while requiring almost as little storage space as the wavelet-based COD. There is no significant variance on the execution time of these three versions of framework COD when changing the window size because the total execution time is mainly dominated by the iterations that k-means algorithm actually performs until it reaches a stable clustering result. However, the wavelet-based COD is faster than the regression-based COD and the adaptive COD when the number of windows observed (parameter p) is smaller, but is slower when p is larger. In our observation, although the calculation of distances in the wavelet-based COD is faster than that in the regression-based COD and the adaptive COD, the wavelet-based COD has to run more iterations to reach a stable clustering result when p is larger. The reason is that the wavelet-based COD approximates data streams by average values, and these average values are closer in older windows (larger p). In other words, the distances between streams are smaller in older windows. Therefore, more streams will move between clusters when the cluster centers change. Consequently, more iterations, also more execution time, of the k-means algorithm are required for the wavelet-based COD when a larger value of p is specified. These experiments show that the summarization techniques in our framework are very accurate and efficient for various clustering queries in a data stream environment. Furthermore, the adaptive COD can achieve the same good quality as the regression-based COD with fewer resources.

4.5 Scalability

To evaluate the scalability of the online maintenance phase, the scale-up experiments on both the number of data points and the number of data streams are conducted. The fast/slow changing data with 50 percent of fast data is used to illustrate how much space can be saved by the adaptive COD. As shown in Fig. 11a, as the number of data points in each stream increases from 1,000 to 10,000, the execution time grows linearly. This property also holds when the number of data streams varies from 100 to 1,000, as shown in Fig. 11c. These results conform to our analyses in Section 3.3 which state that the time complexity of the online maintenance phase of framework COD is linear in both the number of streams and the number of data points

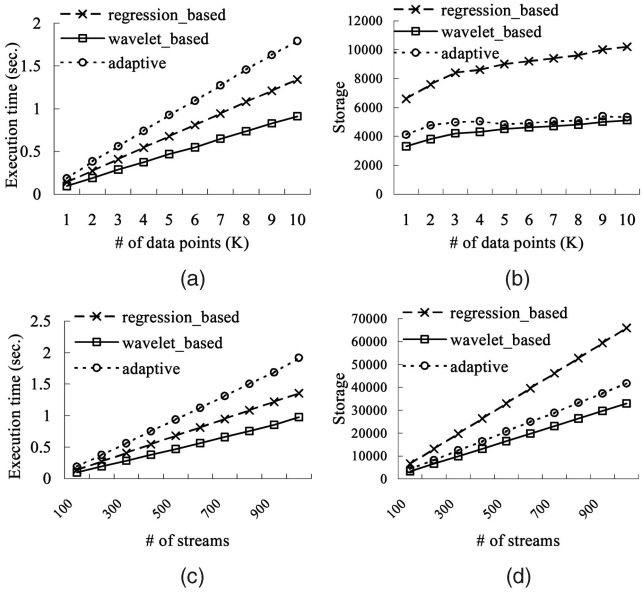


Fig. 11. The scalability analyses of the online maintenance phase. (a) Time versus data number. (b) Storage versus data number. (c) Time versus stream number. (d) Storage versus stream number.

in each stream. Note that the wavelet-based COD is more efficient in the online phase because of fewer parameters being calculated and maintained, while the adaptive COD needs more time to decide which model should be kept. On the other hand, as shown in Fig. 11b and Fig. 11d, the storage space is proportional to $O(\log m)$ and $O(n)$, where m is the number of points in each stream and n is the number of streams. These results do conform to the space complexity analyzed in Theorem 5. In addition, although the adaptive use of framework COD spends more time to build the summary hierarchy, it reduces the storage space significantly. Therefore, the adaptive COD has the ability to generate clusters with the same quality as the regression-based COD, while using the storage space almost as few as the wavelet-based COD, as shown in Fig. 11b.

5 Conclusions

In order to deal with various types of multiple data streams and to support flexible mining requirements, we devised a COD Framework to dynamically cluster multiple data streams. While providing a general framework of clustering on multiple data streams, COD framework had two major advantages, namely, one data scan for online statistics collection and compact multiresolution approximations, which can address, respectively, the time and the space constraints in a data stream environment. Furthermore, with the multiresolution approximations of data streams, flexible clustering demands can be supported. The online maintenance phase of COD provided an efficient algorithm to maintain the summaries of the data streams with multiple resolutions in time linear in both the number of streams and the number of data points in each stream. On the other hand, an adaptive clustering algorithm was devised for the offline phase of COD to retrieve the approximations of the desired substreams from the summary hierarchies as precise as possible according to the clustering queries. We proposed two summarization techniques to maintain summaries of the evolving streams, i.e., 1) wavelets and 2) regression lines. An adaptive version of COD framework was also designed to obtain clustering results with the same quality as the regression-based COD while using much less storage space for the summary hierarchy. As shown in the complexity analyses and also validated by our empirical studies, the algorithms of the COD framework performed very efficiently in the data stream environment while producing clustering results of very high quality.

Appendix

Proofs of Theorems and Lemmas

Proof of Theorem 1. From the equations derived in Section 3.1.2, $\widehat{\theta}$ and $\widehat{\eta}$ can be calculated by $\sum t$, $\sum t^2$, $\sum f$, $\sum tf$, and n. Note that n is the number of data points represented by a fitting model, and can be calculated by $B_t \times (B_h)^L$. The interval $[t_s, t_e]$ of the ith fitting model on level L is equal to $\left[t_e^L - ni + 1, t_e^L - n(i-1)\right]$, where the first fitting model is the latest one. According to the above lemma, $\sum t$, and $\sum t^2$ are both available. Therefore, the least square error linear fit for a substream can be obtained losslessly from the accumulative sum $\langle \sum tf, \sum f \rangle$ and t_e^L of that level. □

Proof of Theorem 2. Let h_{max} be the largest window size on the hierarchy which is not larger than the desired clustering window size w. We have the following equations:

$$h_{max} = B_t \times (B_h)^{L_{max}}, \text{ and } h_{max} \leq w < h_{max} \times B_h. \quad (2)$$

As shown in the above equations, the window size of the fitting models in level L_{max} is h_{max} and the window size of the fitting models in level $(L_{max}+1)$ is $(h_{max} \times B_h)$. The desired window size w is between these two levels. A model in level $(L_{max}+1)$ merges more than w data points into a fitting model. Thus, level L_{max} will be the highest level whose fitting models indicate the pattern changes with window size w. □

Proof of Theorem 3. Let h_{min} be the window size of the fitting models in level L_{min}. We have the following equations:

$$h_{min} = B_t \times (B_h)^{L_{min}},$$

and

$$\begin{aligned} &(t_{now} - t_e^{(L_{min}-1)}) + \alpha_{(L_{min}-1)} \times \frac{h_{min}}{B_h} \\ &< w \\ &\leq (t_{now} - t_e^{L_{min}}) + \alpha_{L_{min}} \times h_{min}. \end{aligned}$$

As shown in the above equations, the total interval covered by the fitting models in level L_{min} is $\alpha_{L_{min}} \times h_{min}$. The data points in the interval $(t_{now} - t_e^{L_{min}})$ can be described by a temporary model which will be described later. Then, the models in level L_{min} and temporary model can cover the whole range of the latest window w queried by a user. However, the total interval in level $(L_{min}-1)$ and its corresponding temporary model $(t_{now} - t_e^{(L_{min}-1)})$ cannot cover the whole range of the

latest window w. Thus, level L_{min} will be the lowest level whose fitting models are still enough to illustrate the pattern with window size w. Note that only the latest window can be approximated by the fitting models from level L_{min}, older windows should be approximated by the models from higher levels. □

Proof of Lemma 4. (a) follows from the definition. The distance between two fitting models $\widehat{f}_A(h_d) = \widehat{\theta}_A(h_d) + \widehat{\eta}_A(h_d)t$ and $\widehat{f}_B(h_d) = \widehat{\theta}_B(h_d) + \widehat{\eta}_B(h_d)t$ is

$$\begin{aligned}
&\sum_{t\in h_d}\Big(\widehat{f}_A^t(h_d) - \widehat{f}_B^t(h_d)\Big)^2 \\
&= \sum_{t\in h_d}\Big[\Big(\widehat{\theta}_A(h_d) + \widehat{\eta}_A(h_d)t\Big) - \Big(\widehat{\theta}_B(h_d) + \widehat{\eta}_B(h_d)t\Big)\Big]^2 \\
&= \Big(\widehat{\theta}_A(h_d) - \widehat{\theta}_B(h_d)\Big)^2 \times |h_d| \\
&\quad + 2\Big(\widehat{\theta}_A(h_d) - \widehat{\theta}_B(h_d)\Big)(\widehat{\eta}_A(h_d) - \widehat{\eta}_B(h_d))\sum_{t\in h_d} t \\
&\quad + (\widehat{\eta}_A(h_d) - \widehat{\eta}_B(h_d))^2 \sum_{t\in h_d} t^2,
\end{aligned}$$

where $\sum t$ and $\sum t^2$ can be calculated efficiently by Lemma 2. Consequently, for (b), the total distance of j fitting models is represented by the summation of each model, i.e., $D_i = \sum_{d=1}^{j}\sum_{t\in h_d}\Big(\widehat{f}_A^t(h_d) - \widehat{f}_B^t(h_d)\Big)^2$. □

Proof of Lemma 5. (a) follows from the definition. By the integration of tf and f in the interval $[t_i, t_e]$, we have

$$\begin{aligned}
\int_{t_i}^{t_e} tf\,dt &= \int_{t_i}^{t_e} t(\widehat{\theta} + \widehat{\eta}t)dt \\
&= \int_{t_i}^{t_e} (\widehat{\theta}t + \widehat{\eta}t^2)dt = \Big(\frac{1}{2}\widehat{\theta}t^2 + \frac{1}{3}\widehat{\eta}t^3\Big)|_{t_i}^{t_e}, \\
\int_{t_i}^{t_e} f\,dt &= \int_{t_i}^{t_e} (\widehat{\theta} + \widehat{\eta}t)dt = \Big(\widehat{\theta}t + \frac{1}{2}\widehat{\eta}t^2\Big)|_{t_i}^{t_e}.
\end{aligned}$$

Consequently, for (b), $R(h') = \Big\langle \sum_{t_i}^{t_e} tf, \sum_{t_i}^{t_e} f\Big\rangle$ can be calculated by the following equations:

$$\begin{aligned}
\sum_{t_i}^{t_e} tf &= \Big(\frac{1}{2}\hat{\theta}((t_e)^2 - (t_i)^2) + \frac{1}{2}\hat{\eta}((t_e)^3 - (t_i)^3)\Big), \\
\sum_{t_i}^{t_e} f &= \Big(\hat{\theta}(t_e - t_i) + \frac{1}{2}\hat{\eta}((t_e)^2 - (t_i)^2)\Big).
\end{aligned}$$

□

Proof of Lemma 6. The proof is divided into two cases, which are the entries from (a) level L_{max}, and (b) level below L_{max}. For case (a), each entry generated is *exactly* with window size w, which means that $w' = w$. For case (b), let h_i be the interval of a fitting model on the ith level of the summary hierarchy. We have the following equation:

$$h_i = B_t \times (B_h)^i.$$

As mentioned above, a fitting model is inserted into an entry if *at least half* of its interval is in the range of that window. Therefore, the worst case occurs at the situations that the first and the last fitting models of the entry are entirely inserted (or removed) because exactly half of their intervals are in (or just more than half of their intervals are not in, respectively) the range of a queried window. That is,

$$|w' - w| \le \frac{h_i}{2} + \frac{h_i}{2} = h_i.$$

Note that h_i decreases exponentially if we retrieve fitting models from lower levels, and the largest difference appears at level $(L_{max} - 1)$. According to (2),

$$|w' - w| \le h_{(L_{max}-1)} = \frac{h_{L_{max}}}{B_h} = \frac{h_{max}}{B_h} \le \frac{w}{B_h}.$$

□

Proof of Theorem 4. For each data stream, every B_t data points generate a fitting model. Therefore, $\frac{m}{B_t}$ models of level 0 have been created in time $O(m)$ after m points arrive. Every B_h models are aggregated to a model in a higher level in time $O(B_h)$ because the average of B_h values is calculated for a wavelet-based fitting model (and, respectively, two summations of B_h values are calculated for a regression-based fitting model). Accordingly, totally

$$\begin{aligned}
\frac{m}{B_t} \times \left(\frac{1}{B_h} + \frac{1}{{B_h}^2} + \ldots + \frac{1}{B_h^{\left(log_{B_h}(\frac{m}{B_t})\right)}}\right) &\le \frac{m}{B_t} \\
\times \left(\frac{\frac{1}{B_h}}{1 - \frac{1}{B_h}}\right) &= O\Big(\frac{m}{B_t(B_h - 1)}\Big)
\end{aligned}$$

fitting models, which are in the levels larger than 0, have been created for a stream. The time required for building the $O(\frac{m}{B_t(B_h-1)})$ models will be $O((B_h \times \frac{m}{B_t(B_h-1)}) = O(\frac{m}{B_t})$. Consequently, the time complexity of the online maintenance phase for n streams is $O\Big(n\Big(m + \frac{m}{B_t}\Big)\Big)$. □

Proof of Theorem 5. For each data stream, every B_t data points generate a fitting model. Therefore, $\frac{m}{B_t}$ models of level 0 have been created after m points arrived. Every B_h models are aggregated to a model in a higher level. Accordingly, the height of the summary hierarchy is $log_{B_h}(\frac{m}{B_t})$ and each level maintains at most α fitting models. Note that each fitting model contains a constant number of parameters, which is one for the wavelet-based model and two for the regression-based model. Consequently, totally $n\Big(\alpha log_{B_h}(\frac{m}{B_t})\Big)$ fitting models are maintained for n streams, which means that the space complexity of the online maintenance phase is $O\Big(n\Big(\alpha log_{B_h}(\frac{m}{B_t})\Big)\Big)$. □

Proof of Theorem 6. The complexity of k-means clustering algorithm is $O(knd)$, where d is the number of dimensions. Since each window of the substream is represented by at

most α fitting models with constant number of parameters, therefore, the time complexity of the offline clustering phase is $O(kn\alpha)$ for a window. □

REFERENCES

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems," *Proc. ACM Symp. Principles of Database Systems (PODS)*, June 2002.

[2] M.R. Henzinger, P. Raghavan, and S. Rajagopalan, "Computing on Data Streams," *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*, vol. 50, pp. 107-118, 1999.

[3] A. Bulut and A.K. Singh, "Swat: Hierarchical Stream Summarization in Large Networks," *Proc. Int'l Conf. Data Eng.*, pp. 303-314, Mar. 2003.

[4] A. Bulut and A.K. Singh, "A Unified Framework for Monitoring Data Streams in Real Time," *Proc. Int'l Conf. Data Eng.*, pp. 44-55, 2005.

[5] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining Stream Statistics over Sliding Windows," *Proc. SIAM Symp. Discrete Algorithms*, pp. 635-644, Jan. 2002.

[6] W. Fan, "Systematic Data Selection to Mine Concept-Drifting Data Streams," *Proc. ACM SIGKDD*, pp. 128-137, 2004.

[7] C.C. Aggarwal, "On Change Diagnosis in Evolving Data Streams," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 5, pp. 587-600, May 2005.

[8] C.C. Aggarwal and P.S. Yu, "Online Analysis of Community Evolution in Data Streams," *Proc. ACM SIAM on Data Mining (SDM '05)*, 2005.

[9] T. Johnson, S. Muthukrishnan, and I. Rozenbaum, "Sampling Algorithms in a Stream Operator," *Proc. ACM SIGMOD Conf.*, pp. 1-12, 2005.

[10] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, "A Framework for Clustering Evolving Data Streams," *Proc. Very Large Data Bases Conf.*, Sept. 2003.

[11] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, "A Framework for Projected Clustering of High Dimensional Data Streams," *Proc. Very Large Data Bases Conf.*, pp. 852-863, 2004.

[12] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering Data Streams," *Proc. Symp. Foundations of Computer Science*, pp. 359-366, Nov. 2000.

[13] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-Data Algorithms for High-Quality Clustering," *Proc. Int'l Conf. Data Eng.*, 2002.

[14] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, "On Demand Classification of Data Streams," *Proc. ACM SIGKDD*, pp. 503-508, 2004.

[15] P. Domingos and G. Hulten, "Mining High-Speed Data Streams," *Proc. ACM SIGKDD*, pp. 71-80, Aug. 2000.

[16] G. Hulten, L. Spencer, and P. Domingos, "Mining Time-Changing Data Streams," *Proc. ACM SIGKDD*, pp. 97-106, Aug. 2001.

[17] E. Keogh, J. Lin, and W. Truppel, "Clustering of Time Series Subsequences is Meaningless: Implications for Past and Future Research," *Proc. IEEE Int'l Conf. Data Mining*, Nov. 2003.

[18] J. Yang, "Dynamic Clustering of Evolving Streams with a Single Pass," *Proc. IEEE Int'l Conf. Data Mining (ICDE '03)*, pp. 695-697, Mar. 2003.

[19] T. Li, Q. Li, S. Zhu, and M. Ogihara, "A Survey on Wavelet Applications in Data Mining," *SIGKDD Explorations*, vol. 4, no. 2, pp. 49-68, 2003.

[20] W.-G. Teng, M.-S. Chen, and P.S. Yu, "Using Wavelet-Based Resource-Aware Mining to Explore Temporal and Support Count Granularities in Data Streams," *Proc. SIAM Int'l Conf. Data Mining*, Apr. 2004.

[21] Y. Chen, G. Dong, J. Han, B.W. Wah, and J. Wang, "Multi-Dimensional Regression Analysis of Time-Series Data Streams," *Proc. Very Large Data Bases Conf.*, pp. 323-334, 2002.

[22] W.-G. Teng, M.-S. Chen, and P.S. Yu, "A Regression-Based Temporal Pattern Mining Scheme for Data Streams," *Proc. Very Large Data Bases Conf.*, pp. 93-104, Sept. 2003.

[23] P.S. Bradley and U.M. Fayyad, "Refining Initial Points for k-Means Clustering," *Proc. Int'l Conf. Machine Learning*, pp. 91-99, July 1998.

Bi-Ru Dai received the BS and PhD degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2000 and 2006, respectively. Her research interests include data mining, data clustering, data stream management, and bioinformatics.

Jen-Wei Huang received the BS degree in electrical engineering from the National Taiwan University, Taiwan, in 2002, and is currently working toward the PhD degree. He is majoring in computer science and is familiar with the data mining area. His research interests include data mining, mobile computing, and bioinformatics. Among these, the Web mining, incremental mining, mining data stream and time series, and sequential pattern mining are his special interests. In addition, some research are on mining general temporal association rules, sequential clustering, data broadcasting, progressive sequential patterns, and bioinformatics.

Mi-Yen Yeh received the BS degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2002. She is currently a PhD candidate in the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan, R.O.C. Her research interests include data mining, data clustering, and data stream management.

Ming-Syan Chen received the BS degree in electrical engineering from National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, in 1985 and 1988, respectively. Dr. Chen is currently a professor and the chairman of the Graduate Institute of Communication Engineering, a professor in the Electrical Engineering Department, and also a professor in the Computer Science and Information Engineering Department, National Taiwan University, Taipei, Taiwan. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 200 papers in his research areas. In addition to serving as a program committee member in many conferences, is currently on the editorial board of the *Very Large Data Base (VLDB) Journal*, the *Knowledge and Information Systems (KAIS) Journal*, the *Journal of Information Science and Engineering*, and the *International Journal of Electrical Engineering*, and is a Distinguished Visitor of IEEE Computer Society for Asia-Pacific from 1998 to 2000, and also from 2005 to 2007 (invited twice). He served as an international vice chair, program chair, program cochair, program vice chairs for numerous conferences. He holds, or has applied for, 18 US patents and seven ROC patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He is a recipient of the NSC (National Science Council) Distinguished Research Award, Pan Wen Yuan Distinguished Research Award, and K.-T. Li Research Penetration Award for his research work, and also the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He also received numerous awards for his research, teaching, inventions and patent applications. He coauthored with his students for their works which received ACM SIGMOD Research Student Award and Acer Long-Term Thesis Awards. Dr. Chen is a fellow of the IEEE and a member of the ACM.