Clustering multivariate climate data streams using fractal dimension

SBBD 2015

XXX SIMPÓSIO BRASILEIRO DE BANCO DE DADOS

13 A 16 DE OUTUBRO DE 2015

PETRÓPOLIS

LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA - LNCC

PROCEEDINGS OF THE 30TH BRAZILIAN SYMPOSIUM ON DATABASES

VANESSA BRAGANHOLO, FABIO PORTO, EDUARDO OGASAWARA, JAVAM C. MACHADO (ORG.).

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

Realização

Laboratório Nacional de Computação Científica

CEFET/RJ

30th BRAZILIAN SYMPOSIUM ON DATABASES

October 13th to 16th, 2015
Petrópolis – RJ – Brazil

PROCEEDINGS

Promotion
Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization
National Laboratory for Scientific Computing - LNCC
Federal Center of Technological Education of Rio de Janeiro - CEFET/RJ

Program Committee Chair
Vanessa Braganholo (UFF)

Steering Committee Chair
Mirella M. Moro (UFMG)

Local Organization Chairs
Fabio Porto (LNCC) and Eduardo Ogasawara (CEFET/RJ)

ISSN: 2316-5170

Brazilian Symposium on Databases (30. 2015, out. 13-16 : Petrópolis, RJ)
B823p Proceedings of the 30th Brazilian Symposium on Databases [recurso eletrônico] / Vanessa Braganholo [et al.] (org.). Petrópolis, RJ. : Laboratório Nacional de Computação Científica, 2015.
xvi, 164p. : il.

ISBN: 978-85-9996-120-9

1. Banco de dados - Encontros 2. Gerência de informações I. Brazilian Symposium on Databases II. Braganholo, Vanessa. (Org.) III. Título.

CDD – 005.74

30th Brazilian Symposium on Databases

full

October 13-16, 2015
Petrópolis – RJ – Brazil

FULL PAPERS

PROCEEDINGS

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

National Laboratory of Scientific Computing – LNCC
Federal Center of Technological Education of Rio de Janeiro – CEFET/RJ

SBBD Program Committee Chairs

Vanessa Braganholo, UFF

Table of Contents (SBBD Proceedings)

sbbd

Clustering Multivariate Climate Data Streams using Fractal Dimension*

Christian C. Bones[1], Luciana A. S. Romani[2], Elaine P. M. de Sousa[1]

[1]University of São Paulo - USP
São Carlos – SP – Brazil

[2]Embrapa Agriculture Informatics
Campinas – SP – Brazil

{chris,parros}@icmc.usp.br, luciana.romani@embrapa.br

Abstract. *A data stream is a flow of data produced continuously along the time. Storing and analyzing such information become challenging due to exponential growth of the data volume collected. In this context, some methods were proposed to cluster data streams with similar behavior along the time. However, those methods have failed on clustering data flows with more than one attribute, i.e., multivariate flows. This paper introduces a new method to cluster multivariate data streams, based on fractal dimension, reading the data only once. We evaluated our method over real multivariate data streams generated by climate sensors. Not only was our method able to cluster the flows of data, but also identified sensors with similar behavior during the analyzed period.*

1. Introduction

The increasing number of devices and sensors that continuously generate a huge amount of data leads to new challenges and applications. For instance, sensors have been used to monitor the pollution in cities, the level of rivers (to prevent flooding) and the meteorological conditions. This flow of data, generated ad infinitum at a high speed rate, is called data stream.

In this scenario, clustering of data streams becomes an active research topic [Bifet and De Francisci Morales 2014, Zhang et al. 2014, Pereira and de Mello 2014, Widiputra et al. 2011] with applications in several contexts. Its goal is grouping in the same cluster data streams that have similar properties and behavior over time, whereas data streams of different clusters must present dissimilar characteristics. For instance, clustering tecniques could be applied to cluster data streams from climate sensors that have similar behavior along a period of time.

Some of the main challenges on extracting knowledge from data streams are: read data only once; capture and represent data evolution along the time; provide answers as soon as the user demand them. It is also desirable to deal with multidimensional data, considering all the variables involved on each stream, i.e., multivariate data streams. Some methods have been developed to work with multiple data streams [Rodrigues et al. 2008, Chaovalit and Gangopadhyay 2009, Widiputra et al. 2011, Pereira and de Mello 2014]. Their approaches in data streams clustering fall into two main categories that are: *Clustering by example*, in which all data points are clustered independently, regardless of the

*The authors are greatful to CAPES, CNPQ and FAPESP for their financial support.

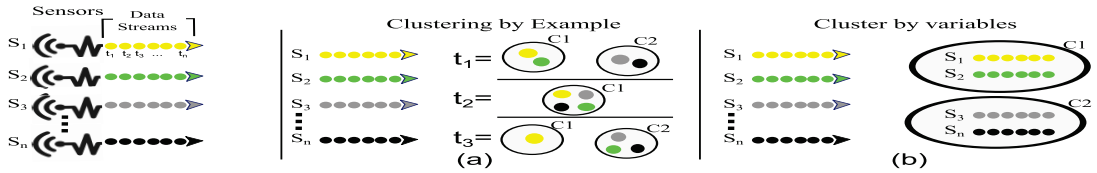


Figure 1. Difference between clustering by examples and clustering by variables.

data sources these data points are from; *Clustering by variable* or *Clustering the entire data stream*, in which a data stream is compared with other data streams and those similar will be clustered into the same cluster. Fig. 1 shows the difference between these two approaches: Fig. 1(a) illustrates clustering by examples at different times t_i such that clusters C_1 and C_2 are created considering the properties of individual data points at time t_1 and, therefor, points from different streams can be at the same cluster. On the other hand, the clustering by variable approach in Fig. 1(b) shows that each cluster is built considering the general properties of the entire flow of data from each sensor.

Although, several online clustering methods have been proposed to process and analyze flows of data in real time [Rehman et al. 2014, Pereira and de Mello 2014, Zhang et al. 2014, Widiputra et al. 2011, Chaovalit and Gangopadhyay 2009, Rodrigues et al. 2008], only few of them try to extract valuable information and group the entire data streams based on their similar behavior over the time [Pereira and de Mello 2014, Widiputra et al. 2011, Chaovalit and Gangopadhyay 2009, Rodrigues et al. 2008]. Furthermore, these methods have failed on clustering data streams that have more than one attribute, i.e., multivariate flows, because they only consider the similarity of attributes independently.

In this work, we are interested in clustering the entire data stream, as we focus on clustering climate sensors that behave similarly along the time. Moreover, we are especially interested in clustering sensors that generate multivariate data streams, taking into account the correlation among the attributes. As climate data streams usually have more than one attribute (e.g. temperature and precipitation) and, in such way, those attributes may have some correlation, considering each attribute as an independent flow is not the best approach for clustering climate sensors.

In this context, this paper proposes the Fractal-based Clustering of Data Streams framework (FCDS), whose the main module is a novel method for clustering multivariate data streams, based on the calculation of fractal dimension. Our method utilizes the fractal dimension, calculated for data streams that have more than one attribute, to cluster data streams that behave similarly along the time. Also, our method keeps the evolution of the data stream by checking cluster membership whenever a new value of fractal dimension is obtained. In other words, our method checks if the data stream is still belonging to the same cluster or if it should be allocated in another one that better describes its behavior in that period of time.

In order to evaluate our method, we used a dataset provided by EMBRAPA Agriculture Informatics - Empresa Brasileira de Pesquisa Agropecuária - Campinas. The dataset includes multivariate data streams from climate sensors of different Brazilian regions. Our results not only indicated that our approach can be a useful method to assist

specialists in analyzing large amounts of climate data, but also helps to identify regions with the same behavior along the time.

The rest of this paper is organized as following. Section 2 presents background concepts and related work. Section 3 describes our approach to cluster data streams. Experimental results are discussed in Section 4 and Section 5 presents final remarks and future work.

2. Background and Related Work

Data stream is an ordered collection of data $s_1, s_2, ...$, generated continuously by one or more sources, that can be read only once [Guha et al. 2003]. In addition, a data stream could have more than one attribute per datum, being called a multivariate data stream. Formalizing these concept, let $\mathbb{S} = \{S_1, \ldots, S_n\}$ be a set of data streams sources where each $S_i = \left\{\vec{d_1}, \ldots, \vec{d_\infty}\right\}$ is a multivariate data stream. Also, each S_i is assumed to contain f attributes such that $\vec{d_i} = [a_1, \ldots, a_f]$ is the set of attributes.

Clustering data stream is a technique used in data mining to perform the grouping of flows of data, so that objects within the same group must have very similar characteristics and properties [Aggarwal et al. 2006]. These characteristics should differ as much as possible from group to group. One of the most common ways to cluster objects is to measure the distance between them [Gama 2010]. However, clustering similar characteristics can be very costly [Guha et al. 2003]. Also, clustering data streams is used to overview the data distribution and as a pre-processing for other algorithms[Gama 2010]. In order to achieve this goal, some basic requirements must be presented in data stream cluster algorithms [Barbará 2002]: (i) Representation of compact size; (ii) Quickly and incremental processing of new data items; (iii) Traceability of changes in groups; (iv) Quick and clear identification of outliers.

Therefore, any new data stream clustering method must be adapted to perform clustering in a continuous, concise and evolving online way entry of the observed sequence. Furthermore, the temporal characteristic of the data stream must also be considered using a small amount of storage space [Guha et al. 2003] and processing time. Such requirements are imposed to contemplate the continuous manner in which the data arrives and the need for analyzing them in real time [Gama 2010].

2.1. Clustering by Variable

Most of the methods proposed in the literature to cluster data streams deal with flows that have only one attribute. Those methods that support more than one attribute do not consider the correlation among attributes to cluster the data streams [Chaovalit and Gangopadhyay 2009, Rodrigues et al. 2008, Rehman et al. 2014, Miller et al. 2014]. They only consider to cluster if all the attributes are similar with all the attributes of other data stream. Leading to results that do not correspond to the behavior of the data streams along the time.

Beyond that, the number of methods that cluster data streams that behave similarly over the time is even more restricted such as the ECM method [Widiputra et al. 2011] and POD-Clus method [Chaovalit and Gangopadhyay 2009]. Thus, this two method will be

detailed in the next sections and they also will be used to compare the achievements of our approach.

ECM

The Evolving Clustering Method (ECM) performs predictions using univariate data streams [Widiputra et al. 2011]. ECM builds local models in two main steps, which are the extraction of relationships between data streams profiles and the detection and clustering recurrent trends when a particular profile emerges. ECM calculates the relationship among data streams profiles using Pearson's correlation [Rodrigues et al. 2008], extracting only the most significant coefficients obtained through tests of statistical significance. Given two data streams a and b the Pearson's correlation is:

$$corr(a,b) = \frac{P - \frac{AB}{n}}{\sqrt{A_2 - \frac{A^2}{n}}\sqrt{B_2 - \frac{B^2}{n}}} \tag{1}$$

where $A = \sum a_i$, $B = \sum b_i$, $A_2 = \sum {a_i}^2$, $B_2 = \sum {b_i}^2$, $P = \sum a_i b_i$, that can be easily upgradeable as soon as each new datum arrives. Then, the ECM calculates the dissimilarity between the data streams a and b by RNOMC (Rooted Normalized One-Minus-Correlation) equation, developed in the ODAC method [Rodrigues et al. 2008] and presented in Equation 2. However, unlike the ODAC, the ECM requires that the whole time series is previously available to perform the calculations offline. Based on a set of decision rules, the algorithm decides how to group the series opting to create, remove, or join groups.

$$rnomc(a,b) = \sqrt{\frac{1 - corr(a,b)}{2}} \tag{2}$$

Because ECM uses the same measure of dissimilarity used by ODAC [Rodrigues et al. 2008], it only detects linear relationships. Another point that limits the application of ECM is how the algorithm decides the union, creating or adding a new element to the cluster, for example, as an element d_j will only be added to the group of d_i if d_j is correlated with all existing elements in d_i, so if d_i has thousand elements and d_j has only one element no correlated with d_i, then d_j will not be added to d_i's cluster. Another disadvantage is that ECM clusters data streams with only one attribute.

POD-Clus

The POD-Clus algorithm (*Probability and Distribution-based Clustering*) [Chaovalit and Gangopadhyay 2009] has four approaches: *clustering by examples* and *clustering the entire data streams* without catching the evolution of the clusters; *clustering by examples* and *clustering the entire data streams* monitoring the evolution of the clusters. Only the latter one is of interest in the context of this work.

The POD-Clus seeks to maintain summaries and discard detailed information of the data points, using normal distributions for this purpose. Each POD-Clus' cluster k receives n data points and stores some average statistics: such as the average μ, standard deviation σ and the updated covariance matrix, whenever new data arrives. These statistics are used to measure the similarity between data streams considering each attribute according to the equation 3. The distance of a muiltivariate data stream S to a cluster

center C is defined as [Kumar and Patel 2007]:

$$D_{SC} = \sum_{f=1}^{F} \frac{(\mu_{Sf} - \mu_{Cf})^2}{\sigma_{Sf}^2}, \quad (3)$$

when S denotes a data stream, C denotes a cluster, f denotes a data feature, μ_{Sf} is the mean of data stream S's feature f, σ_{Sf} is the standard deviation of data streams S's feature f, and μ_{Cf} is the mean of the cluster C's feature f.

POD-Clus also monitors the progress of each cluster, identifying the emergence, the disappearance, the union and the division of the clusters. To detect the appearance of a new group, it maintains a cluster of outliers and when one of these clusters reaches the minimum amount of data streams defined, this cluster becomes a new effective cluster. A cluster disappears when it stops receiving new data by a certain amount of time and old data receives less importance according to a fading factor. To join a cluster to another, it is checked the amount of data which may be in both clusters, generating an overlap between them. If the amount of overlapping data is greater than a predetermined threshold, then the two clusters are merged. To split a group POD-Clus monitors its density and compares it with the normal distribution. If there is significant difference, the group is divided.

The POD-Clus assumes that the whole data stream is derived from a normal distribution that does not guarantee a good representation of the clusters. Another POD-Clus' drawback is the fact that to join a data stream to a cluster all its attributes f have to be similar to the C's features f, disregarding the correlation between the attributes.

2.2. Fractal Dimension

In this work we propose a clustering data stream method based on the calculus of fractal dimension that is used to identify the correlation among the attributes of a data stream in order to capture the data streams' behavior over the time. Then, this behavior will be used to measure the similarity among data streams in order to achieve better clusters results.

A fractal is characterized by the self-similarity property, i.e., it is an object that presents roughly the same characteristics when analyzed over a large range of scales. From the Fractal Theory, the Correlation Fractal Dimension D_2 is particularly useful for data analysis, since it can be applied to estimate the intrinsic dimension of real datasets that exhibit fractal behavior, i.e., exactly or statistically self-similar datasets. The Correlation Fractal Dimension D_2 measures the non-uniform behavior of real data considering both linear and nonlinear attribute correlations [de Sousa et al. 2007]. Therefore, D_2 represents the dimensionality of the dataset regardless of the dimension E of the space defined by its attributes. For instance, a set of points defining a line $z = ax + by + c$ embedded in a three-dimensional space with dimensions $[X; Y; Z]$ (and thus E = 3) has $D_2 = 1$, as there is a linear correlation between its attributes [de Sousa et al. 2007].

A method to measure the fractal dimension of datasets embedded in E-dimensional spaces is the Box-Counting method, which defines D_2 as:

$$D_2 = \frac{\partial log(\sum_i C_{r,i}^2)}{\partial log(r)} \quad r \in [r_1, r_2] \quad (4)$$

where r is the side of the cells in a (hyper) cubic grid that divides the address space of the dataset and $C_{r,i}$ is the count of points in the ith cell.

The work presented in [de Sousa et al. 2007] proposes a technique to detect changes in multidimensional, evolving data streams based on the information of intrinsic behavior provided by the fractal dimension D_2. The authors also present the algorithm SID-meter to continuously measure D_2 over time aimed at monitoring the evolving behavior of the data, such that significant variations in successive measures of D_2 can indicate changes in the intrinsic characteristics, as well as in attribute correlations in the data.

3. Fractal-based Clustering of Data Streams

Our goal is to partitioning the set $\mathbb{S}$, defined in Section 2, in a collection $P = \{C_1, \ldots, C_m\}$ of m disjoint clusters regarding to the fractal dimension analysis of each S_i. Each cluster C is composed of the data stream sources considered similar to each other if they do not exceed an user-defined maximum standard deviation parameter. Our proposal clusters data streams with a similar behavior in an interval of time T_i and also takes into account the correlation among the attributes measured by the fractal dimension D_2, calculated by Equation 4. Our method also follows the evolution of the data streams, where clusters can disappear or be created.

Fig. 2 shows the idea of data stream sources (e.g. sensors) generating new data continuously and send them to the proposed *Fractal-based Clustering of Data Streams* framework (FCDS), which produces evolutive clusters. Notice that for each interval of time T_i, the gathered data are clustered following the fractal dimension of the available data. As new information are received, the clusters are created or rearranged so as to ensure the similarity among their elements and the correlation among the attributes along the time. For instance, from period T_1 to T_2 it is possible to notice a cluster disappearance and from period T_2 to T_3 two new clusters was created.

Let us now illustrate the main components of our proposed framework. The process initiates when a defined number of data stream sources are chosen to be analyzed. As the sources are producing data, they can be directly forwarded to the FCDS framework in order to be processed, as shown in Fig. 3.

The input component of the FCDS is the Sliding Window Module. This module receives the data streams and bounds their information by a sliding window. The sliding window specifies the amount of data buffered for the fractal dimensional calculation. T he window is divided into counting periods (t), and each period has a defined number of events (e) such that each e represents $\vec{d}$ data points. Therefore, $t \times e$ defines the window size l and e also represents its movement step. The window took a default size l, which usually considers either domain experts or the seasonality of the data. But it is possible to define l with different values to achieve different granularities, allowing the incremental calculation of the fractal dimension D_2, as showed in Fig. 4.

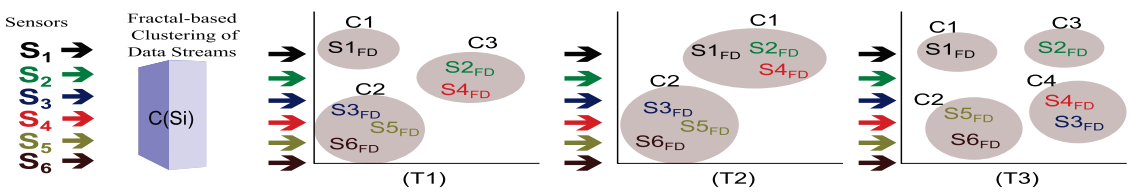


Figure 2. General idea of clustering data streams.

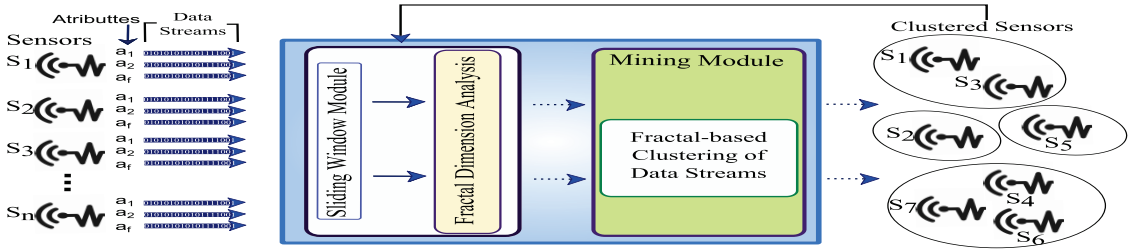


Figure 3. Method of clustering data streams.

Figure 4. Sliding Window of a sensor, counting periods $t = 4$, events $e = 3$.

As soon as there are enough information fulfilling the amount of events e, those data can be now unbuffered for their unique reading. In such case, the sliding window is shifted forward in e units and the region storing the already processed piece of information is released in order to store the continuous incoming data. Additionally, the previously windowed information is dispathed to the module responsible for performing the Fractal Dimension analysis.

The Fractal Dimension Analyzer considers an analog approach applied by SID-Meter [de Sousa et al. 2007] to perform the incremental calculus of D_2. Unlike the methods already proposed in the literature, this kind of technique enables to iterate over data containing more than one attribute. Thus, applying techniques such as the box-counting (Section 2.2), this module transforms a piece of multivariate data stream of size e, which represents the measures, to exactly one point of fractal correlation, summarizing that amount of data and the correlation among their attributes. Our intuition is that using a technique which correlates the involved attributes along the time leads to a better recognition of similar data stream sources than using the raw values of the attributes.

Subsequently, the reduced amount of fractal points is sent to the data mining step. The Mining Module is the main component of the FCDS framework and aims at clustering the correlated points according to their similarity and a user-defined maximum standard deviation (σ_{MAX}). In order to perform the clustering step, let us introduce the Algorithm 1.

The Algorithm 1 iterates over the set X comprising the points of fractal correlation (fd) such that each point is labeled with its respective data stream S_i. Initially, it is verified whether or not the data stream source S_i already belongs to some existing cluster C_j composing the partition P (line 2). Supposing S_i was not clustered yet, it is necessary to find for a cluster to insert S_i into. For this purpose, for each cluster $C \in P$, the boolean procedure `findCluster` (line 8) looks for the cluster C_j with the lowest difference between the centroid of C_j and the analyzed point x. This process follows the model

Algorithm 1: Fractal-based Clustering of Data Streams (FCDS)

Input: The set X containing pairs of streams S_i and its fractal dimension fd;
The partition P of clusters;
The maximum standard deviation σ_{MAX}.
Output: The partition P of clusters with similar data streams.

```
1  foreach x ∈ X do
2      if x.S_i is in a cluster C_j ∈ P then
3          if updateCluster(C_j, x, σ_MAX) = false then
4              if findCluster(x, σ_MAX) = false then
5                  C_new ← createNewCluster(x);
6                  rearrangeCluster(C_j, C_new);
7      else
8          if findCluster(x, σ_MAX) = false then
9              C_new ← createNewCluster(x);
10 return P;
```

introduced in Equation 5, where $C = \{c_1, \ldots, c_k, \ldots, c_n\}$.

$$\Delta(C, x) = \left(\frac{1}{n}\sum_{k=1}^{n} c_k.fd\right) - x.fd \tag{5}$$

Once the cluster C which minimizes the Equation 5 regarding to the element x was obtained, x is assigned to C iff the condition expressed in Equation 6 holds. Otherwise, the new cluster C_{new} containing the element x is created (line 9) and included in the partition P.

$$\sqrt{\frac{1}{n+1}\left(\left(x.fd - \left(\frac{1}{n}\sum_{p=1}^{n} c_p.fd\right)\right)^2 + \sum_{k=1}^{n}\left(c_k.fd - \left(\frac{1}{n}\sum_{p=1}^{n} c_p.fd\right)\right)^2\right)} \leq \sigma_{MAX} \tag{6}$$

Considering the data stream S_i is already part of some cluster, it is necessary to check whether or not the data stream source S_i remains in the previous assigned cluster. In order to employ such verification (line 3), the Mining module re-execute the calculus of Equation 6 regarding to the new value of the fractal dimension $x.fd$. If so, the statistic (function Δ) of the considered cluster C_j is updated. In the case where the left-hand side of Equation 6 exceeds the user-defined maximum standard deviation σ_{MAX}, the algorithm tries to reallocate the element x in an existing cluster C_k so as to minimize the value computed in Equation 5 (line 4) and also hold the condition defined in Equation 6. If there is not such cluster C_k satisfying both conditions, a new one (C_{new}) is created to include the element x (line 5). Now, when creating a new cluster to x, the elements already present in C_j are checked if they are better included in C_{new} (minimizing Equation 5) and inserted in it if they do (line 6). Notice that the `rearrangeCluster` procedure is able to suppress existing clusters if all of their elements are moved and they become empty.

At the end of one iteration over the set X, the partition P is composed of clusters containing the similar data streams sources in relation to the fractal correlation (line 10). As the sources are continuously generating measures, the sliding window shifts forward and, as soon as there are enough data to repeat the process, the Mining module is re-invoked. Therefore, the obtained clusters evolute to reflect the changes in the gathered information along the time.

4. Results

This paper reports a novel method to cluster data streams that behave similarly over the time, based on the correlation of their attributes by the calculus of fractal dimension. For this purpose, we introduced the FCDS framework.

In order to evaluate our proposal, we applied the following methodology: 1) test our approach using a real dataset; 2) assess the quality of the obtained groups by the use of the Silhouette measure; 3) apply the same methodology to the baselines methods, previously described in Section 2.1, in order to compare them with our approach. This methodology was used to analyze three key points: (i) the cohesion of the obtained partition of clusters, (ii) the ability of the FCDS framework to capture similar data streams behavior along the time and (iii) the quality of the generated clusters.

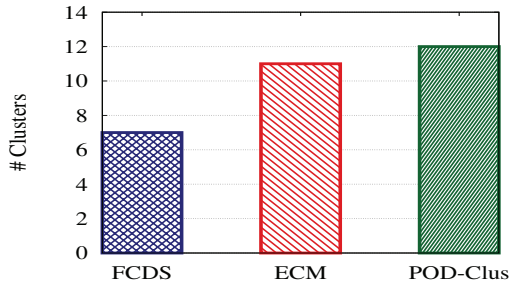
So, we employed a dataset composed of 145 climate sensors (sources of data streams) collecting 3 distinct daily measures (minimum—maximum temperature and precipitation) each one. The considered measures belong to the South and Southeast Brazilian regions, in a period from January 1990 to December 1994. This dataset was obtained in cooperation with the Embrapa Agriculture Informatics (Campinas, SP, Brazil).

In order to measure clusters' quality, we apply the well-known Silhouette. This is a measure commonly employed to evaluated how well the elements are disposed in their respective clusters. The values of Silhouette range from -1 to 1, such that clusters that obtain values above 0.5 are considered of good quality.

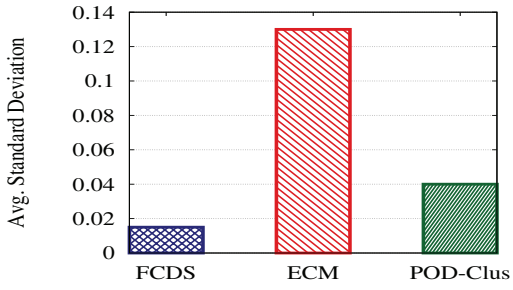
Finally, we compare our results with ECM and POD-Clus methods (both in Section 2.1) once they are the techniques presented in the literature aiming at solving the mainly issues: compact representation; fast data processing; traceability of changes; outliers identification. As mentioned in Section 2.1, only POD-Clus support multivariate data stream. But it only measure the dissimilarity of the attributes without considering the correlation among them. Therefore, in order to deal with the correlation among multiple attributes, presenting in the data streams, we calculate the fractal dimension to each sensor, creating in this way an unidimensional flow to use as input for ECM and POD-Clus. The parameter employed in ECM was correlation threshold equal 0.4. While the parameters for POD-Clus were outlier threshold equal 3 and boundary size equal 0.2. The main results are depicted in Fig. 5.

Fig. 5(a) presents the amount of clusters generated by each considered method. The proposed FCDS technique was able to identify the similarity among the data streams and partitioning them in a fewer number of clusters than the other algorithms. That result suggests our framework better summarized the same amount of information keeping the quality of the clusters, as can be seen in Fig. 5(b).

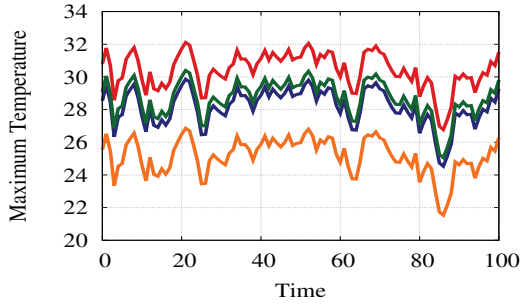
Fig. 5(b) shows the average of the standard deviations among the total number



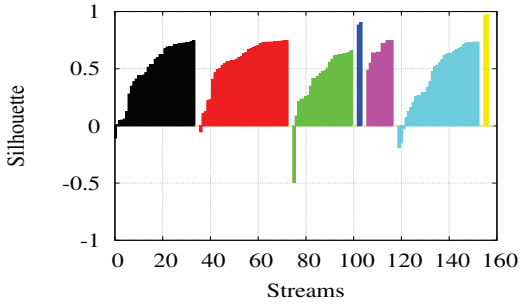
(a) Quantity of clusters generated by each method



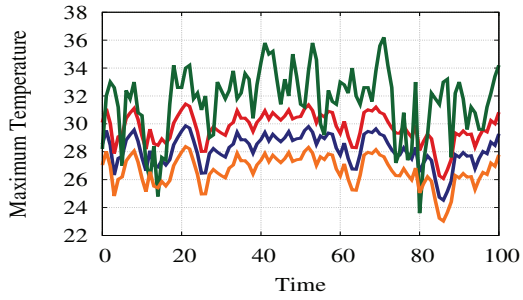
(b) Average of clusters standard deviation



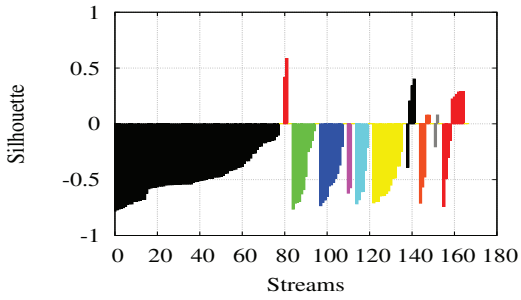
(c) Plot data streams raw data of the FCDS' cluster with the greatest standard deviation



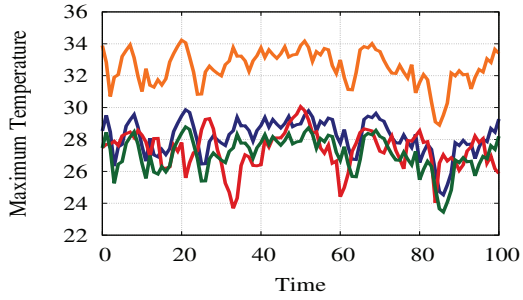
(d) Sillouette of FCDS' clusters



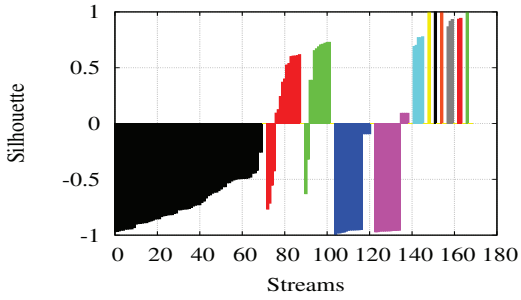
(e) Plot data streams raw data of the ECM' cluster with the greatest standard deviation



(f) Sillouette of ECM' clusters



(g) Plot data streams raw data of the POD-Clus' cluster with the greatest standard deviation



(h) Sillouette of POD-Clus' clusters

Figure 5. Comparative results

of clusters. In that study, the small is the standard deviation the better the elements are clustered. The average of the FCDS's clusters standard deviation was 88.46% less than the result of the ECM method and 62.50% regarding to the POD-Clus. It means that

clustering using FCDS, a data streams is belonging to a cluster that it probably should be in. In addition, it is important to highlight that the POD-Clus method achieved a low average standard deviation due to many generated clusters contain a single data stream, as will be seen further in Fig. 5(h).

Figs. 5(c), 5(e) and 5(g) illustrate the behavior found in the worst cluster (maximum standard deviation) generated by the each considered method. As sampled in Fig. 5(c), the data stream sources were correct included in the same cluster by the FCDS regarding to the correlation among their attributes. Unlike, the ECM method (Fig. 5(e)) was unsuccessful in correct capturing the correlation among the data streams, including streams in clusters which they are not part of. This fact can be confirmed by the two distinct behaviors observed in the same figure. Analogous, the POD-Clus method also misplace data streams in wrong clusters, recurring to the same behavior observed for the ECM method, as depicted in Fig. 5(g).

Fig. 5(d) shows the Sillhoute results obtained by the FCDS technique. The FCDS built clusters with 65% of the data streams Silhouette measures upper to 0.5. Although our method produced fewer clusters than the considered others, that measure indicates that small amount was good enough to represent the behavior of the analyzed dataset.

Fig. 5(f) presents the Silhouette measure to the clusters computed by the ECM method. In this analysis, only one stream was included in the correct cluster. Those results indicate the ECM method produced bad quality clusters when compared to FCDS.

Fig. 5(h) show the the Silhouette values to the clusters generated using the POD-Clus method. When compared with the ECM, the former achieve better results than the latter, correct clustering 29 data streams of a total amount of 145. However, POD-Clus in those 29 data streams, 4 were recognized to be outliers composing a single-element cluster each one. Thus, such behavior affected the standard deviation presented in Fig. 5(b).

Thus, considering the obtained results, the FCDS framework better represented the behavior of different data streams along the time. The approach based on fractals showed promising results to deal with multivariate data streams, helping to capture the behavior of climate streams over time and obtaining fewer clusters with good quality.

5. Conclusion and Future Work

Clustering of data streams is one of the most employed approach to analyze data that is potentially endless and evolves over the time. Nevertheless, the literature provides few methods for clustering the entire data streams. However, such proposals often deal with data streams composed of a single attribute or do not apply appropriate strategies for multivariate flows.

In this paper we introduced a framework to cluster a set of multivariate data streams considering the fractal correlation among their attributes, also complying the basic requirements to cluster data streams. It also takes into account the behavior of distinct streams along the time. The proposed Fractal-based Clustering of Data Streams framework is composed of minor modules, each one responsible for a specific processing of the data. The core of the FCDS framework lies in the computation of the fractal dimension of a piece of the data streams and its subsequent clustering. The use of the fractal dimension allowed to better identify the correlation among the attributes of the data streams. Also,

our method performs the clustering of multivariate data streams supporting their evolution in an incremental way, thereby helping to improve the quality of the clusters.

We performed a set of experimental evaluation of the FCDS framework and compared it againt two recent correlate methods, ECM and POD-Clus, in order to verify the capacity of representing similar data streams behaviors along the time and keeping the quality of the produced clusters.

As an ongoing direction, we intend to analyze other kind of data but the climate ones and improve the recognition of outliers so as to avoid clusters with a single element.

References

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2006). A framework for on-demand classification of evolving data streams. *IEEE TKDE*, 18(5):577–589.

Barbará, D. (2002). Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, 3(2):23–27.

Bifet, A. and De Francisci Morales, G. (2014). Big data stream learning with samoa. In *2014 IEEE Int. Conf on Data Mining Workshop*, pages 1199–1202.

Chaovalit, P. and Gangopadhyay, A. (2009). A method for clustering transient data streams. In *Proc. ACM SAC '09*, pages 1518–1519, New York, NY, USA. ACM.

de Sousa, E. P., Traina, A. J., Jr, C. T., and Faloutsos, C. (2007). Measuring evolving data streams behavior through their intrinsic dimension. *New Generation Computing*, 25(1):33–60.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman and Hall Book, Boca Raton, Florida, USA.

Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. (2003). Clustering data streams: Theory and practice. *IEEE TKDE*, 15(3):515–528.

Kumar, M. and Patel, N. R. (2007). Clustering data with measurement errors. *Computational Statistics & Data Analysis*, 51(12):6084–6101.

Miller, Z., Dickinson, B., Deitrick, W., Hu, W., and Wang, A. H. (2014). Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73.

Pereira, C. M. M. and de Mello, R. F. (2014). Ts-stream: clustering time series on data streams. *Journal of Intelligent Information Systems*, 42(3):531–566.

Rehman, M. Z., Li, T., Yang, Y., and Wang, H. (2014). Hyper-ellipsoidal clustering technique for evolving data stream. *Knowledge-Based Systems*, 70(0):3–14.

Rodrigues, P. P., Gama, J., and Pedroso, J. P. (2008). Hierarchical clustering of time-series data streams. *IEEE TKDE*, 20(5):615–627.

Widiputra, H., Pears, R., and Kasabov, N. (2011). Multiple time-series prediction through multiple time-series relationships profiling and clustered recurring trends. In Huang, J., Cao, L., and Srivastava, J., editors, *Advances in Knowledge Discovery and Data Mining*, pages 161–172. Springer, Shenzhen, China.

Zhang, X., Furtlehner, C., Germain-Renaud, C., and Sebag, M. (2014). Data stream clustering with affinity propagation. *IEEE TKDE*, 26(7):1644–1656.