Revista de Sistemas de Informação da FSMA
n. 15 (2015) pp. 30-39

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

TRILHA ESTUDANTIL

Visualization in Big Data: A tool for pattern recognition in data stream.

Victor Hugo Andrade Soares, *Graduate in Information Systems, UFV*,
Joelson Antônio dos Santos, *Graduate in Information Systems, UFV* and
Murilo Coelho Naldi, *Phd. Adjunct Professor-III, UFV*
{victorhugoasoares, joelsonn.santos}@gmail.com, murilocn@ufv.br

***Abstract*—The development of new technologies is responsible for the generation and storage of continuous and massive amounts of data. Such type of data is known as data stream. The analysis of data streams may be advantageous in many fields, like bioinformatics, medicine, companies and others, as it may result in important information about the data. In this work, we propose a new software tool for Data Visualization that permits the analysis of the evolution of clusters in real time during the data streaming. The proposed visualization tool is add-on for SAMOA, a new variant of MOA (Massive Online Analysis) for massive data streams mining and processing distribution.**

***Index Terms*—Data Mining, Data Streams, Data Visualization.**

I. Introduction

THE constant evolution and usage of Information Technology has changed the way people communicate and solve personal and professional problems. Due to this usage, daily generated data has been gathered from the interaction between customers and organization, social relations in network among friends or bank transactions [1]. The ability to gather and analyze those data in real time is called data stream [2].

The challenge of working with Data Visualization in this context is to analyze the data type quickly, so that we can find patterns and help decision making [3]. Moreover, the data stream may be finite or infinite. Such unpredictability of the data volume makes its analysis even more costly. [4].

Mechanisms that are able to automatize the data analysis processes are necessary and in this scenario we employ data mining techniques. Data mining is made of artificial intelligence and statistical techniques whose aim is to extract useful information from a determined amount of data that is either stored or in a continuous stream [5].

One of the techniques that can help identify patterns in data is data clustering. It is a set of unsupervised methods that intends to group objects according to their characteristics. This can be achieved through dissimilarity or similarity metrics among the analyzed objects [6].

One of the characteristics of data streams is that they can be infinity, that is, it is not possible to store them. After a given analysis, data must be discarded so that new data can be received. This volatility makes it more difficult to recognize new trends in groups, for data cannot be reprocessed. The only stored knowledge is the information on the groups themselves [4].

There are techniques in literature that were created to detect changes in data flows. M-DBScan [7], for instance, compares the entropy, that is, the degree of uncertainty of a new group with the entropy of the previously generated groups. The change in entropy characterizes a change in the data pattern. The goal of this paper is to make it possible to visualize the data behavior during the flow, independently of the occurrence of changes. Therefore, we did not use clustering techniques to detect the changes.

Once the data was clustered, it was necessary to analyze the results in order to recognize the patterns and extract knowledge from them. Nevertheless, the analysis of abstract data is not intuitive. Data visualization has come to facilitate the understanding and the perception of certain characteristics together with the abstract data. The human perception system is sensible to differences in colour, size, shapes and distances among objects. Hence, data visualization uses graphic resources to illustrate the data attributes. This helps the user realize the similarities and differences among them [8].

This paper intends to resent a visualization tool called *2D Data Stream Viewer*, developed to help identify patterns and trends in data gathered in a continuous stream. There are some platforms to analyze data streams, two of which we highlight: Apache Spark Streaming and SAMOA (Scalable Advanced Massive Online Analysis) [9]. Spark Streaming is a distributed analysis platform and a recent extension of the platform Apache Spark [10]. The SAMOA platform, on the other hand, is a distributed platform to analyze continuous data flows and also a data mining library for continuous data flows. As an advantage, SAMOA allows for data analysis in real time. This characteristic impacts on low latency when dealing with huge data amounts. The platform Spark Streaming, on the other hand, stores parts of the data in batch for posterior analysis [10].

The visualization tool proposed here is a complement

to the SAMOA platform. SAMOA contains a distributed and massive version of the *KMeans* clustering algorithm, which is implemented and adapted to the continuous flow model. Differently from other algorithms such as DBSCAN [11], SLINK and its variations [12], which have quadratic computational complexity, the *KMeans* algorithm has linear complexity [5]. Hence, the data processing times for this algorithm is more scalable than others data clustering algorithms. Because of this, we use the *KMeans* algorithms when grouping data streams. The clustering results allow for the gathering and storage of necessary information to illustrate the proposed visualization tool.

Section II presents a theoretical foundation on the development of visualization tools and clustering algorithms. In Section III, we present the development and the functionalities of the proposed tool. Finally, Section IV will present some final considerations.

II. Theoretical Foundations

In this section we present some of the main concepts used in the development of the proposed tool. Among them, concepts on data streams, grouping techniques, and an introduction on data visualization techniques and the usage of human vision to detect patterns.

A. Data Stream

There are applications that when executed demand the data to be gathered and processed in real time. In those cases, storage and queries in persistent conventional databases tables are not viable [13]. It is also not possible to use techniques such as sampling, due to the fact that there is not a defined amount of data to be received. These characteristics define the data stream [5].

Formally, the definition of a data stream consists on a sequence of points *X1,....,Xk*, where k is not known. Each *Xi* arrives at time *T1,...,Tk*, and each point has d dimensions and can be written as *Xi = (Xi1,...,Xid)* [14].

In the data stream processing, we need to deal with transient value tables, which are filled with data collected in a continuous stream. This gathering phase is known as the online phase. After filling the tables, an analysis of the gathered data is performed, a step known as the offline phase. Data analysis begets a knowledge that must be stored for future decision making. Finished the offline phase, data is deleted from the tables so that new data can be gathered. The tables are transient because they store the data only for a small interval, until the data is analyzed and discarded or persisted into a data base. This transience is the difference between flow analysis to the analysis of data stored in persistent tables [4].

This data volatility makes it more difficult for the analysis to find satisfying results. Considering that the data is available for a short span of time and those are discarded or persisted, it becomes impossible to process them again in due time. This makes it more difficult to adapt to new decision models that may come with new characteristics of the received data [15]. In order to decrease those difficulties, we will present some functionalities of the visualization tool *2D Data Stream Viewer* that are used for the perception of new characteristics in a continuous data flow in Section III.

B. Data Clustering

Human beings have the ability to categorize and group objects around them as they see them. One example is an employee at a crop whose goal is to separate fruits according to their sizes. Hence, small fruits will be in a separated group from big and average ones. This task is performed relatively easy by most people. Similarly, data mining has techniques able to group objects according to their similarities, techniques that are knows as data clustering.

The clustering techniques intend to find similarities among objects within a data set [16]. This process is performed through the calculation of distances in the n-dimensional data space, where data dimension is given by the number of attributes contained in the set. Even though it may sound simple, there is subjectivity in the definition of what is a group [5]. For example, Figure 1 shows that there are several ways to group the same data set. Figure 1(a) represents the original data set, Figure 1(b) shows two distinct groups and Figure 1(c) shows three distinct groups. In this example, each point belongs fully to a single group, but there are algorithms that deal with group overlaying, in which a point can belong to more than one group. In this paper we will deal only with strict groups, that is, groups for which there is no overlaying.

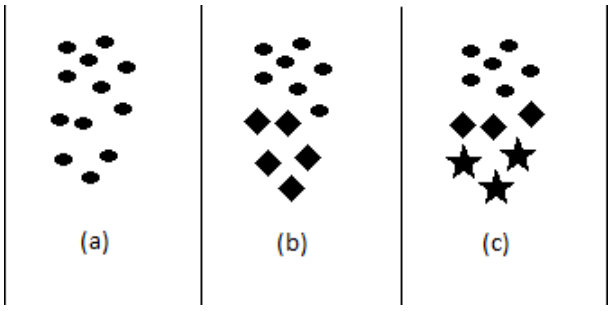

Fig. 1. Different ways to group the same data set.
Source: Adapted from [5].

There are different types of data clustering [17]. Among the most common are the ones based in prototypes, where the objects are more similar to the prototype of its group and less similar to the prototypes of the other groups. There is also density based clustering, in which objects are grouped according to their concentration in the n-dimensional space, that is, the high density of objects forms groups. Another type of clustering is based on graphs, where there are nodes (objects) that are connected by edges that represent the relationship among them.

C. Kmeans

Kmeans is a partitional clustering algorithm based on prototypes [18]. *Kmeans* tries to find the groups through centroids that are formed by the averages of the objects within the same group. Some type of similarity is used to define groups, so that an object is more similar to the objects in its group and less similar to objects in another data cluster.

The *Kmeans* algorithm has simple characteristic and is implemented in many ways in literature. The parameter k, which indicates a priori the number of groups we seek in the clustering process, must be defined by the user of estimated. One of the techniques to estimate the value of k uses evolutionary algorithms [19]. In Table I we present the steps to implement the classic *Kmeans* algorithm, where the attribute k is an input parameter defined by the user. Figure 2 illustrates an example of the execution of the *Kmeans* algorithm for $k = 3$.

TABELA I
CLASSIC *KMeans* ALGORITHM

Classic *KMeans* algorithm
1: Choose k initial centroids randomly defined by the user.
2: Calculate the distance from each object to its centroids.
3: Attribute each object to its closest centroid.
4: Calculate the average for each group and design new positions to the centroids.
5: Repeat steps 2, 3 and 4, until no centroid changes position

Source: Adapted from [18].

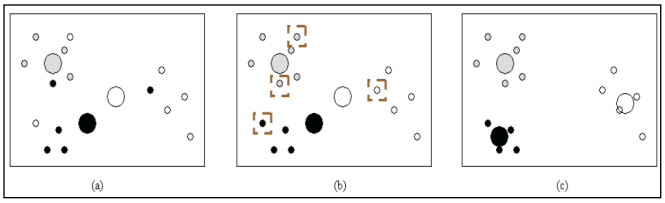

Fig. 2. Example of the execution of the *KMeans* algorithm. (a) Each element was designed to one of three groups randomly and the centroids (larger circles) of each group were calculated (b) The elements were now reassigned to the groups whose centroids were closest to them (c) The centroids were recalculated. The groups are already in their final form. If not, we would repeat steps (b) and (c) until they were.

Source: [6].

Proximity relations or similarity metrics are used to define the real distance between objects and the centroids [18]. The most known and used metrics are a derivation of the Minkowski metric[6], which is given by the following formula:

$$d(x, y) = \sqrt[q]{\sum_{i=1}^{t} (x_i - y_i)^q} \quad (1)$$

A Minkowski metric with $q = 2$ is the widely known euclidian distance, which seeks similarities among objects in euclidian space. When $q = 1$ we have the Manhattan distance (also called block city distance)[5]. In this paper we implemented a *KMeans* algorithm adapted to work with data in a continuous flow in the distributed platform SAMOA, presented in the next section. In the similarity calculations, we used the euclidian distance. The adaptations implemented in the algorithm are described in the Subsection III-A.

D. SAMOA

The *Scalable Advanced Massive Online Analysis* (SAMOA) is a distributed platform that includes a library of data stream mining algorithms. SAMOA is capable of adapting to different distributed processing platforms, including *S4* and *Storm* [9].

SAMOA has some implemented algorithms, among them the *Clustream* clustering algorithm. *Clustream*, which also is referred as a *framework*, has two processing phase for data stream: Online and Offline. The Online phase is the process of summarization of the points that arrive continuously for the analysis phase. The offline phase consists of clustering the collected points in the previous phase [14]. Additionally, SAMOA also guarantees to its users the easy implementation of new algorithms for data analysis and also allows to develop connections to other platforms [9].

Given that the clustering analysis is a complex task, a single technique to solve clustering problems is not enough [5]. Hence, one of the goals of this work was to increase the number of clustering algorithms available for SAMOA. At Subsection III-A we describe the implementation of the traditional *KMeans* algorithm adapted for continuous flow.

The SAMOA platform has plugins for the capture of different synthetic and real data streams. Real data demonstrate a situation or application from the real world. Usually, those data sets are stationary, that is, they are stored in a data repository. When referring to a continuous flow, data come in different time intervals, from one or more sources and are processed in real time. An example of a real application is the one from *Yahoo*, which uses the SAMOA platform to detect spams in e-mails [20].

On the other hand, synthetic data, at least theoretically, simulate a random and undefined sequence of points. This randomness is configurable, which guarantees a flexible data generation and allows us to simulate flows with different configurations [21]. Hence, the synthetic data flexibility became relevant to the experiments performed in this research, allowing us to test the visualization tool proposed with different data configurations.

The difference when using artificial data is that they are generated already normalized. With the algorithm implemented in the SAMOA platform, the plugins for real data are already available. The adaptation of the algorithm to analyze real data depends on the preprocessing and normalization of the raw data gathered. Once preprocessed, the real data can be grouped and analyzed as well as the artificial data.

E. Data Visualization

Data visualization is a field from Graphic Computing that intends to help the user analyze and understand abstract data [22]. These data sets are usually gathered through mining a large data set [23]. Using this techniques takes advantage of the human visual system for the task of identifying patterns and trends present in the data citeCard1999.

Data visualization uses basic elements that the human perception system can quickly assimilate, such as colour, size, shape, proximity and movement [8]. The fact that the human being realizes those characteristics quickly allows us to use them to represent each data attribute. Hence, it is possible not only to assimilate them quickly, but also to represent a large amount of data at once. When large amounts of data are presented visually, it is possible to realize groups, gaps, maxima, minima and several other characteristics [8].

Getting a visual representation of the data set can be generalized into an automated process. The abstract data is gathered, the Data Visualization tool processes this data and generates to the user a representative image [24]. Figure 3 illustrates this process.

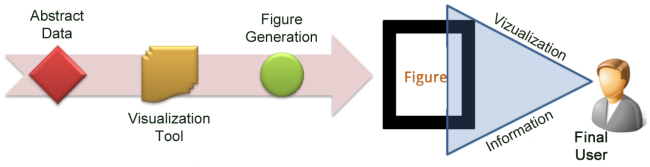

Fig. 3. General process to generate a visual representation of the data.

Source: Adapted from [24].

Several visualization tools are found in the literature with the goal of making the process of analyzing abstracts data sets more intuitive. This includes the usage of different visualization techniques [25], as those oriented to pixels, hierarchical, graph based and others.

Given the existence of several gathering processing and data analysis methods, the visualization tools are not yet capable of illustrating some specific analysis situations, as in the case of data stream analysis.

Considering the main 2D graphic visualization tools found in literature, as *Weka* [26], *MGV* [27] and Polaris [28], the three tools use data stores into persistent tables for processing and visualization. *Weka* implements several algorithms for data mining, such as classification, clustering, association and result validation, but does not allow the gathering and visualization of data stream.

For visualization of data streams, [29] proposes a multidimensional scaling technique whose goal is to decrease the computational time necessary to process the gathered data and draw the result of the continuous flow into a 2D graphic. Nevertheless, the technique does not allow the visual following of the stream in order to visualize changes and trends in the time line in which the stream was gathered. This work develops a 2D Data Visualization tool that is a complement to SAMOA. This tool allows the analysis data that was gathered and clustered in a data stream. In Subsection III-B we present the details of the development of this tool.

III. Development and Presentation

In this section we will describe the steps needed to adapt the *KMeans* algorithm to the data stream model and the necessary data storage that will be used in the visualization tool. We will also present the development of the visualization tool *2D Data Stream Viewer* and the functionalities implemented in this tool.

A. Clustering algorithm

We chose the SAMOA platform to develop the *KMeans* algorithm adapted to a continuous flow and this gave us some advantages. The first one is that the platform has the function of keeping a network communication abstraction that is need in conventional distributed systems. Hence, the developers need only to elaborate the application logic without worrying about the necessary network protocols. The second advantage is that the platform is developed with the *Java* programming language. Like all languages that support the reusability concept of the object oriented paradigm, it is possible to develop new applications in *Java* without code redundancy.

Table II shows the steps of the *Kmeans* adapted to data stream. While the classic algorithm requires the user to predefine the parameter k, which represents the amount of groups to be found, this version of the algorithm expects three different parameters: k, x and t.

As in the classic algorithm, parameter k is the number of groups to find. Parameter x is the maximum amount of data that can be clustered at once before being discarded. An idea is to limit the value of x to the size of the available working memory, but the value can be defined arbitrarily. Parameter t limits the maximum amount of time the algorithm will wait while gathering data. If the amount x of data is not gathered in time t, the clustering phase starts using only the data that has already been gathered.

These parameters have important roles in the context of the data stream analysis. Since we do not know the amount of data which will be received, parameter x prevents a memory overflow or simply divides the flow into standard size batches. Parameter t solves the potential idleness of the algorithm. If the data stream terminates or goes into a long pause, limiting the time t prevents the algorithm from going idle for a long time. Hence, if the algorithm is idle for a time t, the clustering process is initiated.

In this paper, the instances are generated by SAMOA's *RandomRBFGeneratorEvents* class and its attributes are normalized within the interval [0,1]. The instances are generated simulating a data stream but the number of instances may be defined by the user. Defined by the

TABELA II
KMeans Algorithm for Data Flow

***KMeans* Algorithm for Data Flow**
1: Gathers the data from the stream until the table of size x be filled or time t is reached.
2: If this is the first iteration, choose k initial medoids randomly. From the second iteration on, consider the centroids from the previous iteration.
3: Calculate the distance from each object to the centroids.
4: Attribute each object to the closest centroid.
5: Calculate the average for each group and attribute new positions to the centroids.
6: Repeat steps 3, 4 and 5 until no centroid changes position.
7: Calculate the *SSE* from each generated group and the total clustering *SSE*.
8: Store into a file the final centroid positions, the amount of objects in each group, the *SSE* for each group and the total clustering *SSE*.
9: Deletes the gathered data, return to step 1 and repeat all the algorithm until the stream is over.

developers of the platform, each object generated by this class always has two dimensions.

Besides the traditional steps that are similar to the class *KMeans* algorithm, in the first execution of the algorithm the position of the k centroids are chosen by the position of k random medoids, that is, by the position of k elements stored in the database. After executing the *KMeans*, we store the final centroids for each cluster, the number of objects, *SSE* (*sum squared error*) of each dimension of the individual clusters, and the *SSE* of the total clustering. After that, the objects are deleted so that new ones can be gathered and a new clustering process can start.

SSE is one of the most common metrics in the evaluation of clustering algorithms [5]. The calculation of the *SSE* of a group is given by Equation 2.

$$SSE_{group} = \sum_{x \in C_i} dist^2(m_i, x) \tag{2}$$

In this formula, x is a point that belongs to group C_i, and m_i is the position of this groups's centroid. The calculation of the *SSE* of the entire clustering is given by the sum of the k *SSE* found in the groups, as shown by Equation 3.

$$SSE_{total} = \sum_{j=1}^{k} \sum_{x \in C_i} dist^2(m_i, x) \tag{3}$$

In our application, the *SSE* is calculated also for each attribute dimension. For instance, for two dimensional data, we store the *SSE* for each group, as shown in Equation 2 and also the *SSE* of the dimension 1 for each group and the *SSE* for dimension 2. Equation 4 shows how the calculation of the *SSE* is performed for a single dimension. The *SSE* for dimension t is the sum of the squared distances of dimension t to the point x until the dimension t of the centroid m_i. This calculation is performed for all d dimensions of the data.

$$SSE_{Dim_t} = \sum_{x \in C_i} dist^2(m_{i_t}, x_t) \quad \forall \quad t \in d \tag{4}$$

The metric is important to evaluate the quality of the performed clusterings. The smaller the *SSE* for a group, the smaller the points dispersion and the better is the the group. The *SSE* for each dimension are also calculated because they allow us to verify in each dimension the data is more scattered. Besides the *SSE*, there are other cluster quality metrics such as homogeneity, group density, intragroup maxima and minima values and others.

From the second clustering of the data stream, the initial centroids are no longer random. We will use the final positions of the centroids in the last clustering performed. This is done so that there are no exchange in positions among two or more centroids. This makes it possible, when using the visualization tool, to realize the change of a specific cluster during the data stream. The information on the clusters are stores sequentially in a history log. At the end of each clustering process, the text file with extension ".vjm" is updated with information on the new clustering performed. This extension was created specially for the visualization tool, allowing us to differentiate the files supported by the tool from the traditional text files.

Each data stream generates a single ".vjm" file. During the data stream, several clusterings are performed, each one of them with a piece of the data stream. The final information on each clustering process is stored sequentially in the ".vjm" file. The file is used in the *2D Data Stream Viewer* tool, allowing the user to visually analyze the development of the clusters, since it is possible to identify patterns and trends in the data. Figure 4 shows how the data must be saved in the ".vjm" text file. The texts that come after the character '%' are comments.

```
% "vjm" file with information of a Data Stream
% Number k (groups)
4;

% Cluster 1

% X and Y coordinates of the 4 centroids
[[0.225, 0.104], [0.723, 0.256], [0.343, 0.876], [0.978, 0.768]];
% SSE of the X axis of the 4 groups
[1.341, 0.945, 2.549, 1.023];
% SSE of the Y axis of the 4 groups
[0.984, 0.835, 2.232, 1.254];
% SSE of the 4 groups
[3.451, 5.768, 6.295, 3.275];
% Number of objects in each groups
[233, 167, 356, 244];

% Cluster 2

% X and Y coordinates of the 4 centroids
[[0.297, 0.209], [0.700, 0.290], [0.402, 0.850], [0.923, 0.723]];
% SSE of the X axis of the 4 groups
[1.112, 1.254, 2.654, 0.997];
% SSE of the Y axis of the 4 groups
[0.820, 1.332, 2.101, 0.980];
% SSE of the 4 groups
[3.002, 5.997, 6.100, 2.978];
% Number of objects in each groups
[218, 180, 348, 254];
```

Fig. 4. ".vjm" text file format. The first piece of data is the number k of groups. Afterwards come the coordinates for the k centroids of the first clustering process. Next come the *SSE*'s of the first dimension of the k and afterwards, the *SSE*'s from the second dimension of the k groups. Then come the *SSE* from the k group and finally, the number of objects per group.

B. 2D Data Stream Viewer

The *Java* programming language was chosen to develop the visualization tool because of its portability and the fact that it was already used in the SAMOA platform. The tool has three main goals: make result analysis more intuitive; help identify the changes in data and; visualize the areas with higher data density.

In order to make result analysis more intuitive, we used the concepts of data visualization, where the colours and formats of graphic representations trigger the human perception system, making it easier to realize the differences among groups. The changes in the data may be seen by the visualization of the changes in the groups along the data stream. This was implemented through an automation of visualization of several groups sequentially. Visualization of areas with higher data density was possible through the functionality of global data plotting, that is, the plotting of all points of data stream in a single space.

The functionalities of the tool can be accessed in a single window, as illustrated in Figure 5. The tool works with text files with the extension ".vjm", which are created during the execution of the *KMeans* algorithm inside SAMOA, where each data stream generates a different ".vjm" file, each of which contains a timestamp at which the data stream began.

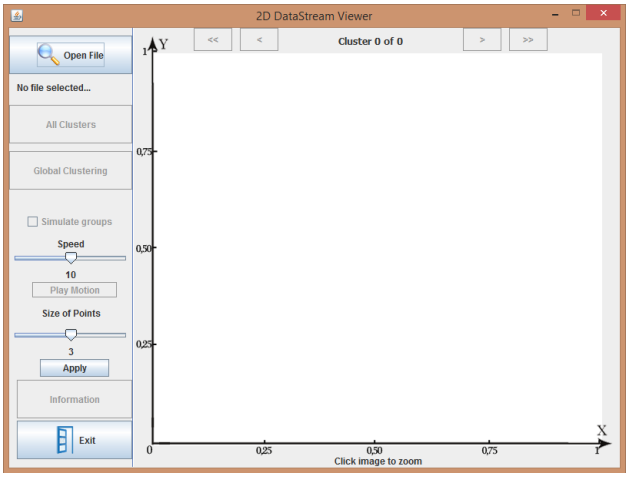

Fig. 5. Window of the *2D Data Stream Viewer* tool.

We developed a functionality to open ".vjm" files, which is done by the button *"Open File"*. It is possible to open files from data streams already finished or from active data stream. In the case of active ones, a *Thread* is activated to verify modifications in the open files so that new data can be loaded into the working memory. With the open file and the stored data loaded into the working memory, the coordinates X and Y of the centroids are normalized for the graphic size, which consists of an image of 600x600 pixels. After coordinates normalization, we plot the data into the graphic with a distinct color for each one of the k groups. Figure 6 shows the plotting of six final centroids from the first clustering from a *Kmeans* algorithm execution. On the top of the screen it is possible to visualize the number of clustering illustrated in the graphic and the total number of clusterings available for visualization.

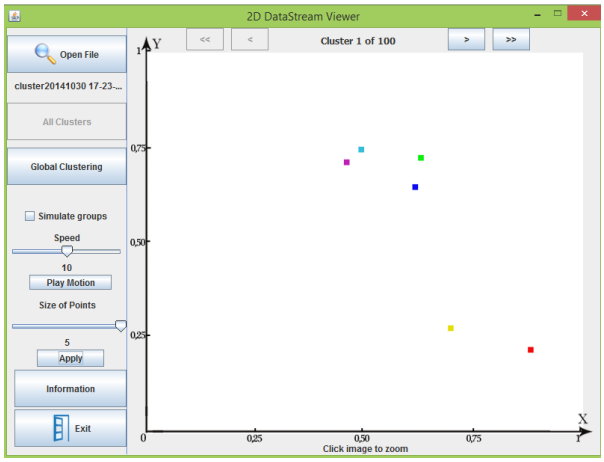

Fig. 6. Example of the plotting of 6 centroids from clustering 1 out of 100 available clusterings.

The arrows in the top part of the window were developed to navigate through the clustering history. Given that the algorithm groups batches of data, several clusterings are made during a single data stream. With the navigation arrow, it is possible to go forward or backward one clustering or access directly the first or last clustering of the history.

The button *"Play Motion"* works as an automated sequence of clustering visualizations. The speed the clusterings are changed in the graphic is controlled by the scrollbar *"Speed"*. The minimum speed between each frame is two second and the maximum, 0.1 second. When visualizing the clusterings sequentially, it is possible to have an idea of how they change. This helps identify changes and realize new trends in the data flow. The visualization tool makes the changes more explicit, allowing for easier perception than when the analysis was performed only on the numerical results of the clusterings.

The bar *"Point size"* changes the number of *pixels* used to represent one piece of data in the graphic. The bar uses normalized values from 1 to 5 and in the image, each data can be represented by a point from 1 to 81 pixels. When the amount of data to show is relatively small, larger points make it easier to visualize. When the amount of data is relatively large, smaller points represent better the real data density. Figure 7 illustrates the difference between using smaller and bigger points in the data visualization.

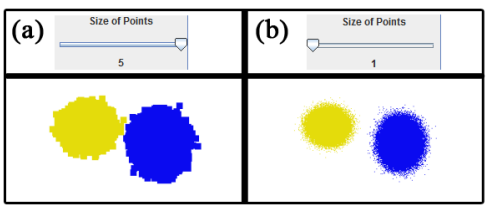

Fig. 7. Example of using the bar *Point size*.

The button *"Global Clustering"* is used to show the centroids existing in the clustering history. this functionality offers the visualization of the final densities in the data stream. With this functionality it is possible to realize the real tendency of the data throughout a long period, elevating the reliability of the analysis for pattern recognition. Figure 8 demonstrates the global clustering for a full history.

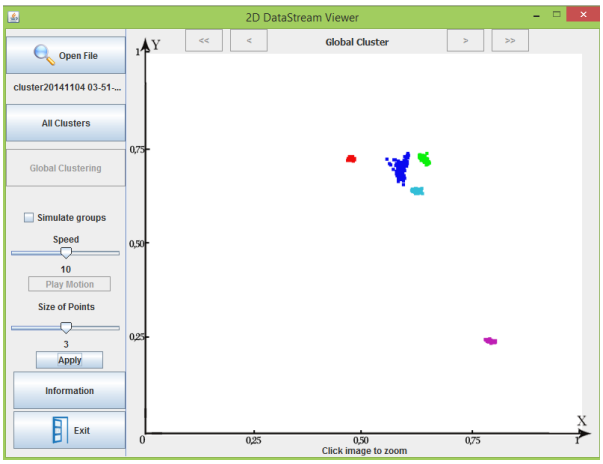

Fig. 8. Global clustering for a full history.

The button *"Information"* shows the informations on the clusters. If it is pressed while the tool is showing a global cluster, a message window will exhibit the amount of clusterings in the history, the number k and the total amount of data clustered in the flow. In Figure 9 we see the global information of a history.

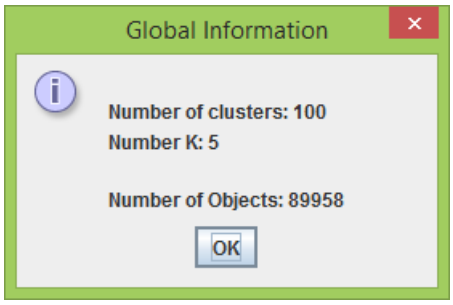

Fig. 9. Information on the global clustering of a history.

When the button *"Information"* is pressed when the tool is showing a single clustering, a window appears showing the information on each centroid, the amount of data and the variance of each group and the total variance of the clustering. Figure 10 shows the information of a single clustering in the history.

The visualization of the centroids as data representatives may not express the real trend and density of the data. This happens when the groups have discrepant amounts of data or variances. Since all centroids are plotted with the same dimensions, it is not possible to identify these differences through the image. As a solution, we created the checkbox titled *"Simulate groups"*. When activated,

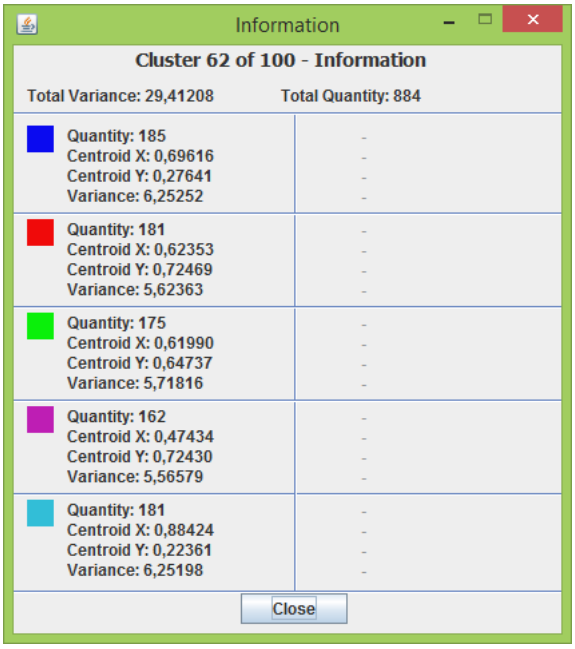

Fig. 10. Information on a clustering in the history that has 5 groups.

this functionality uses the statistical concept of the normal distribution to simulate group data. The points are generated randomly obeying the limits of a normal distribution, which is given by the average (position of the centroids) and the variance of each group. The simulation can be applied in an individual group or in the global clustering. Figure 11(a) shows a simulation of five groups in an individual clustering and Figure 11(b) shows a simulation of the global clustering of the full history.

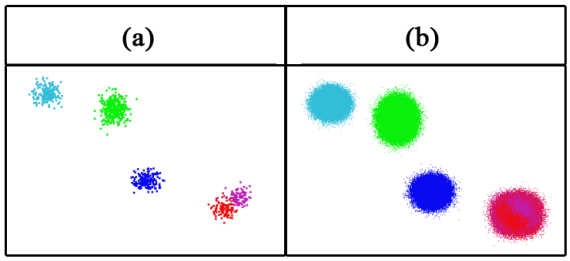

Fig. 11. Simulation of a simple and global clustering with$k$=5

As can be seen in Figure 11, the groups represented by colours red and purple have become intertwined in the simulation of the global clustering. Due to this unification and considering the data density, we can assume that these two groups are actually a single one. Hence, the amount of groups (k) considered should drop to only 4 different groups. Since the value of k is defined by the user prior to data gathering, there may be differences in the clusters representation due to this predefined parameter.

In a first case, when the value of k is low, there may be clusters with high variances, which is due to the presence of hidden subclusters that are merged into a single cluster. In a second case, when the value of k is high, there may occur the existence of clusters that are too close and that

could be merged into one. The tool *2D Data Stream Viewer* does not allow for the perception of subgroups for the correction of the first case, but the simulation of the groups and the global clustering allow for the visualization of the intersections among clusters, allowing the use to consider the possibility of joining them.

Figure 12 shows the simulation of a global clustering for a single data stream for the values k=1 and k=5. In Figure 12(a) it is possible to see a single group in the data, without gaps or spaces among them. In Figure 12(b) it is possible to realize that the 5 groups have become only two, with a large gap between them. Even though it is the same dataset, only the second case allowed for the identification of the second group. We can say that the higher the value of k, the higher the precision on the identification of different groups. Even though similar objects are represented by different colors, visualization allows for the perception of similarity due to their proximity.

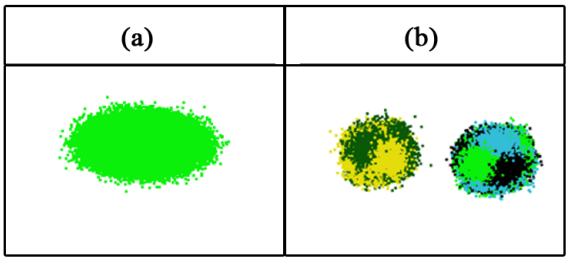

Fig. 12. Visualization of the same data base with k=1 and k=5.

IV. Final Considerations

GIVEN that platforms and algorithms for data stream analysis are still evolving, adapting the *KMeans* algorithm is a contribution to the study of new forms of clustering of data streams. Adapting the algorithm allowed us to store the history of the clustered data that are represented by the centroids of the groups. This functionality had the goal of helping future decision making based on the behavior of data streams analyzed by the algorithm.

The *2D DataStream viewer* tool made it easier to analyze clusters and identify the changes during the data stream generated by SAMOA. With it, it is possible to visualize trends and simulate the data in the streaming using only the centroids calculated during the data stream development. Given that the clustering algorithms are developed to deal with a large volume of data, find knowledge on them without actually storing them is a satisfactory development. Since the tool only uses the final numerical data from the clusterings, any clustering algorithm can be adapted so that the results can be visualized by the tool.

As future work, we intend to perform practical usability tests, so that the tool can become more intuitive and also look for other possible functionalities that will make the tool more useful. We also intend to create an API inside the visualization tool to connect with real data flows, allowing the tool to gather, cluster and visualize the results with the smallest latency possible.

References

[1] D. Parikh e P. Tirkha, "Data mining & data stream mining – open source tools ", *Nature*, vol. 2, pp. 5234–5239, 2013, ISSN: 2319-8753.

[2] S. Guha, A. Meyerson, N. Mishra, R. Motwani e L. O'Callaghan, "Clustering data streams: Theory and practice.", em *IEEE Trans. Knowl. Data Eng.*, 2003, pp. 515–528.

[3] IBM e S. B. School, "Analytics: the real-world use of big data - how innovative enterprises extract value from uncertain data", em *Executive Report*, 2012, pp. 1–19.

[4] K. Faceli, J. Gama, A. C. P. L. d. Carvalho e A. C. Lorena, *Inteligência Artificial, Uma Abordagem de Aprendizado de Máquina*. LTC, 2011, vol. 1, pp. 1–378, ISBN: 9788521618805.

[5] P. N. TAN, M. STEINBACH e V. KUMAR, *Introdução ao Data Mining, Mineração de Dados*, 1ª ed., C. Moderna, ed. 2009, pp. 1–978, ISBN: 9788573937619.

[6] R. Linden, "Técnicas de agrupamento", *Revista de Sistemas de Informação da FSMA*, nº 4, pp. 18–36, 2009. endereço: http : / / www . fsma . edu . br / si / edicao4/FSMA_SI_2009_2_Tutorial.pdf.

[7] R. M. M. Vallim, J. A. A. Filho, R. F. de Mello, A. C. P. L. F. de Carvalho e J. Gama, "Unsupervised density-based behavior change detection in data streams", *Intell. Data Anal.*, vol. 18, nº 2, pp. 181–201, mar. de 2014, ISSN: 1088-467X.

[8] E. G. E. Carreon, "Técnicas de visualização de informação para a análise de dados de sensores e biossensores", diss. de mestrado, ICMC/USP, São Carlos/SP, 2013.

[9] G. De Francisci Morales, "Samoa: A platform for mining big data streams", em *Proceedings of the 22Nd International Conference on World Wide Web Companion*, sér. WWW '13 Companion, Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, 2013, pp. 777–778, ISBN: 978-1-4503-2038-2.

[10] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker e I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing", em *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, sér. NSDI'12, San Jose, CA: USENIX Association, 2012, pp. 2–2.

[11] M. Ester, H.-P. Kriegel, J. Sander e X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", em *Proc. of 2nd International Conference on Knowledge Discovery and*, 1996, pp. 226–231.

[12] R. Sibson, "Slink: An optimally efficient algorithm for the single-link cluster method.", *Comput. J.*, vol. 16, nº 1, pp. 30–34, 1973. endereço: http://dblp.uni-trier.de/db/journals/cj/cj16.html#Sibson73.

[13] B. Babcock, S. Babu, M. Datar, R. Motwani e J. Widom, *Models and Issues in Data Stream Systems*, sér. PODS '02. Madison, Wisconsin: ACM, 2002, pp. 1–16, ISBN: 1-58113-507-6.

[14] C. C. Aggarwal, J. Han, J. Wang e P. S. Yu, "A framework for clustering evolving data streams", VLDB '03, pp. 81–92, 2003.

[15] J. Beringer e E. Hüllermeier, "Online clustering of parallel data streams.", Data Knowl. Eng., 2006, pp. 180–204.

[16] C. A. R. Pinheiro, *Inteligência Analítica*, 1ª ed., C. Moderna, ed. 2008, pp. 1–414, ISBN: 9788573937077.

[17] J. Leskovec, A. Rajaraman e J. D. Ullman, *Mining of Massive Datasets*, 1ª ed., Cambridge, ed. EUA, 2011, pp. 1–336, ISBN: 1107015359.

[18] D. T. Larose, *Discovering Knowledge in Data: an introduction to data mining*, 1ª ed., Wiley-Interscience, ed. 2005, pp. 1–240, ISBN: 9780471687542.

[19] M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka e A. C. P. L. F. Carvalho, "Efficiency issues of evolutionary k-means", *Appl. Soft Comput.*, vol. 11, nº 2, pp. 1938–1952, mar. de 2011, ISSN: 1568-4946.

[20] A. Murdopo, A. Severien, G. d. F. Morales e A. Bifet, *Samoa - Developer's Guide*, 0.0.1, Y. L. Barcelona, ed. 2012, pp. 1–48.

[21] J. P. Reiter, "Satisfying disclosure restrictions with synthetic data sets", *JOURNAL OF OFFICIAL STATISTICS-STOCKHOLM-*, vol. 18, nº 4, pp. 531–544, 2002.

[22] G. Grinstein, M. Trutshl e U. Cvek, "High-dimensional visualizations", San Francisco/CA: 7th Data Mining Conference KDD Workshop, 2001, pp. 1–14.

[23] E. F. Iepsen, "Estudo e desenvolvimento de uma ferramenta de visualização de informações temporais para banco de dados estruturados no formato mestre/detalhe", diss. de mestrado, ESIN/UCPEL, Pelotas/RS, 2008.

[24] A. L. S. Moraes, "Visualização de informações fuzzy para auxílio em diagnósticos médicos", diss. de mestrado, ESIN/UCPEL, Pelotas/RS, 2009.

[25] P. C. Wong e R. D. Bergeron, "30 years of multidimensional multivariate visualization.", em *Scientific Visualization*, 1994, pp. 3–33.

[26] I. H. Witten, E. Frank e M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011, ISBN: 0123748569, 9780123748560.

[27] J. Abello e J. L. Korn, "Mgv: A system for visualizing massive multidigraphs.", *IEEE Trans. Vis. Comput. Graph.*, vol. 8, nº 1, pp. 21–38, 2002. endereço: http://dblp.uni-trier.de/db/journals/tvcg/tvcg8.html#AbelloK02.

[28] C. Stolte e P. Hanrahan, "Polaris: A system for query, analysis and visualization of multidimensional relational databases", em *Proceedings of the IEEE Symposium on Information Vizualization 2000*, sér. INFOVIS '00, Washington, DC, USA: IEEE Computer Society, 2000, pp. 5–, ISBN: 0-7695-0804-9.

[29] P. C. Wong, H. Foote, D. Adams, W. Cowley e J. Thomas, "Dynamic visualization of transient data streams", em *Proceedings of the Ninth Annual IEEE Conference on Information Visualization*, sér. INFOVIS'03, Seattle, Washington: IEEE Computer Society, 2003, pp. 97–104, ISBN: 0-7803-8154-8. endereço: http://dl.acm.org/citation.cfm?id=1947368.1947389.