Data Stream Mining: an Evolutionary Approach

Angélica Veloza Suan
Ingeniera de Sistemas
ID: 300393

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Bogotá, D.C.
Mayo de 2013

Data Stream Mining: an Evolutionary Approach

Angélica Veloza Suan
Ingeniera de Sistemas
ID: 300393

Thesis Work to Obtain the Degree of
Magister en Ingeniería de Sistemas y Computación

Advisor
Elizabeth León Guzmán, Ph.D.
Doctor in Computer Science

Coadvisor
Jonatan Gómez Perdomo, Ph.D.
Doctor in Computer Science

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Bogotá, D.C.
Mayo de 2013

Title in English

Data Stream Mining: an Evolutionary Approach

Abstract: This work presents a data stream clustering algorithm called *ESCALIER*. This algorithm is an extension of the evolutionary clustering *ECSAGO* - Evolutionary Clustering with Self Adaptive Genetic Operators. *ESCALIER* takes the advantage of the evolutionary process proposed by *ECSAGO* to find the clusters in the data streams. They are defined by sliding window technique. To maintain and forget clusters through the evolution of the data, *ESCALIER* includes a memory mechanism inspired by the artificial immune network theory. To test the performance of the algorithm, experiments using synthetic data, simulating the data stream environment, and a real dataset are carried out.

Keywords: Mining data streams, clustering

Título en español

Minería de Flujo de Datos: un Enfoque Evolutivo

Resumen: Este trabajo presenta un algoritmo para agrupar flujos de datos, llamado *ESCALIER*. Este algoritmo es una extensión del algoritmo de agrupamiento evolutivo *ECSAGO* - Evolutionary Clustering with Self Adaptive Genetic Operators. *ESCALIER* toma el proceso evolutivo propuesto por *ECSAGO* para encontrar grupos en los flujos de datos, los cuales son definidos por la técnica *Sliding Window*. Para el mantenimiento y olvido de los grupos detectados a través de la evolución de los datos, *ESCALIER* incluye un mecanismo de memoria inspirado en la teoría de redes inmunológicas artificiales. Para probar la efectividad del algoritmo, se realizaron experimentos utilizando datos sintéticos simulando un ambiente de flujos de datos, y un conjunto de datos reales.

Palabras clave: Minería de flujos de datos, agrupación

Acceptation Note

Jury
Name 1

Jury
Name 2

Advisor
Elizabeth León Guzmán, Ph.D.

Coadvisor
Jonatan Gómez Perdomo, Ph.D.

Bogotá, D.C., Mayo, 2013

Dedication

This work is dedicated to the memory of my best friend:

Juan Carlos

Acknowledgements

I would like to thank my professors Elizabeth León and Jonatan Gómez for their support in the development of this work. I would also like to express my thanks to my mother and my brother for their continuos encouragement and support. Also, many thanks to Diana, Fernando and specially John and Julián, for all their help, friendship, encouragement and support during this process.

Contents

List of Tables

List of Figures

List of Algorithms

CHAPTER 1

Introduction

In recent years, the interest in analyzing and extracting information from large volume of online data has grown. Such data, known as data streams, are generated at high rates by different environments such as satellite systems and network monitoring applications, among others. These environments are able to handle large amounts of data that are produce at the same time and that must be quickly processed. For example, the National Center for Atmospheric Research (NCAR) has some applications such as a system with data weather in real-time, a system to manage satellite images of earth surface and a system of weather forecasting. The Space Science and Engineering Center (SSEC) at the University of Wisconsin has a system that maintains an image gallery in real time of Antarctica, hurricanes and tropical storms; and an Atmospheric Emitted Radiance Interferometer (AERI). Other examples of data streams are the millions of images captured by the Solar and Heliospheric Observatory (Soho), the ATM (Automated Teller Machine) millionaire transactions, e-commerce applications, and the great number of sensor measurements in areas such as telecommunication and networks [96, 67, 74].

Analyze and understand data streams is not an easy task, due to limitations of computational resources used for this purpose (e.g., not having enough memory to store all data, not having enough time to process and compute data). Moreover, since data evolve in time, the model of its behavior at a specific time, may not be obtained and therefore not exploited at time. For this reason, many conventional algorithms, even when they have been proved to be effective, have been discarded to be applied to data streams.

Classical data mining techniques have been often used for extracting such information regardless online analysis (analysis made to data at the time they are generated). This kind of analysis requires features that are not present in some approaches based in those classical techniques. Some of those features are: a) high-speed in preprocessing and processing data (single scan over the data and constant time to process them), b) an efficient memory management (fixed memory used), c) online incremental learning, and d) the ability to forget past information [43, 89, 153].

To address the problem of mining data streams, some algorithms from data mining have been adapted and some other algorithms have been defined from scratch having in mind the specific set of tasks associated with the mining data streams. A common way to analyze data streams involves using clustering techniques in which a group of data is

represented by an element. Although there are a large amount of proposed works to deal with this problem, there are some issues that can be improve.

Maintaining memory of past data through time is one of the features that has been handled by some algorithms. Some of them use a factor of memorization or forgetting, others have proposed the use of on-line and off-line phases to deal with this issue. Artificial Immune Network Theory intends to explain the immune system behavior including one of its most important features: how immune memory works [93]. The general frame of how the immune system learns continuously and how it is able to forget with or without external stimulus, could be compared with the way in which a cluster data streams model carries out its learning process. The model must be able to learn and forget data which are generated continuously (as an external stimulus) and to maintain the knowledge through time. Among the advantages of using the immune memory mechanism are: it does not require a factor of memorization or forgetting and it does not require an off-line phase to process the data.

This work presents the *ESCALIER* algorithm, an extension of *ECSAGO* (Evolutionary Clustering with Self Adaptive Genetic Operators) algorithm proposed in [109]. *ECSAGO* is based on the Unsupervised Niche Clustering (UNC) [128] and uses the Hybrid Adaptive Evolutionary (HAEA) [69] algorithm. *ECSAGO* has show good results mining large volumes of dynamic data and it may be possible to adapt this algorithm to mining data streams. *ESCALIER* algorithm takes advantage of the evolutionary process proposed by *ECSAGO* to find clusters, it also uses the strategy of summarization (prototypes that represent the founded clusters) presented in [110], and additionally, a memory mechanism based on the artificial immune network theory in order to maintain and forget clusters has been added. Sliding window technique is included into the model, to handle the data streams *ESCALIER* will be tested using synthetic and real datasets. In the next section, the general and specific goals of this work are presented.

1.1 Goal

The purpose of this work is to develop a clustering algorithm for mining large volumes of dynamic data streams, extending the evolutionary clustering algorithm with parameter adaptation *ECSAGO*. To aims this goal, it is proposed specifically:

1. Making literature review: This work presents a study and an analysis of different clustering techniques for mining data streams. To carry out this, a literature review is performed. This review is focused on techniques and algorithms for dynamic data mining streams, and management the evolution of data. A state of the art is developed and presented.

2. Designing and Implementing of the data stream clustering algorithm: This work presents a clustering algorithm for mining data streams. This algorithm is an extended model of *ECSAGO* algorithm. The prototype includes mechanisms for incremental learning, consistency and data preprocessing. Also, a complexity analysis in time and space to determine the efficiency of the proposed prototype is carried out.

3. Designing and Implementing a Memory Mechanism: This work proposes the incorporation of a memory mechanism in the prototype. Techniques for memory management such as artificial immune networks are studied. A memory mechanism is designed and included in the algorithm.

4. Defining the Experimental Design: This work presents the validations of the proposed prototype with real and synthetic datasets, in order to test the quality of the extracted information and the ability to handle the problem. Also, a report with the comparison of the results achieved with those obtained by other algorithms is presented.

1.2 Main Contributions

The main contribution of this work is an algorithm to cluster data streams called *ESCALIER*. It is an extension of *ECSAGO* algorithm, that uses ideas from artificial immune network to handle memory issues and incorporates the sliding window technique as a mechanism to support data streams. *ESCALIER* algorithm has been presented in *CEC - IEEE Congress on Evolutionary Computation 2013*. Additionally, a state of the art on clustering data stream mining techniques is proposed. It includes a structure of some works developed, organized by technique and presented as a tree diagram. A journal with the state of the art will be presented in *IJIPM: International Journal of Information Processing and Management*.

1.3 Thesis Outline

This document is organized as follows: **Chapter 2** presents the basic concepts used, specifically on data streams and artificial immune networks; the proposed state of the art on clustering data stream mining techniques, and the analysis of *ECSAGO* algorithm. **Chapter 3**, presents the *ESCALIER* (Evolutionary Stream Clustering Algorithm with self Adaptive Genetic Operators and Immune Memory) algorithm, and the results and analysis of the performed experiments. Finally, **Chapter 4** presents conclusions and proposes the future work.

CHAPTER 2

Background

In this chapter the definition of data stream is given together with the description of the mining and clustering data stream processes. A state of the art of the clustering data stream mining organized by technique (Data-Based and Task-Based) is presented. *ECSAGO* algorithm is explained and the main concepts about Artificial Immune Theory are presented.

This chapter is divided in four sections. Section 2.1 presents the concepts associated with mining and clustering data streams, and also a state of the art of the clustering data mining techniques. Section 2.4 describes in detail the phases that composed the *ECSAGO* algorithm. Section 2.5 gives the definition of concepts presented by the Artificial Immune Theory. Section 2.6 presents a summary of this chapter.

2.1 Data Streams

A data stream is a continuous sequence of ordered data (each data have a specific arrival time) generated in real time by events that occur day to day. In a formal and general way, a data stream F can be represented as a multidimensional set of records $X_1, X_2, ..., X_k, ...$, generated in a time $T_1, T_2, ..., T_k, ...$, respectively. If each record X_k has d dimensions, then F can be represented as follows:

$$F = X_1, X_2, ..., X_k, ... = (x_{11}, x_{12}, .., x_{1d}), (x_{21}, x_{22}, .., x_{2d}), .., (x_{k1}, x_{k2}, .., x_{kd}), ...$$

F is characterized by quickly changing, having temporal correlations and growing at an unlimited rate, which means that F can have a size very close to infinity.

Data stream representation (model) can vary depending on how the stream describe F. Models in increasing order of generality are as follow [126]:

- Time Series Model: In this model each sequence of data X_k in the data stream F, appears in increasing order of time T_k. This model is used to observed the traffic networks, financial trades, among others applications.

- Cash Register Model: In this model each X_k in F is a positive increment of X_{k-1}. In this way, $X_k = X_{k-1} + Inc_k$, where Inc_k is the stream produced in the time T_k. It is important to notice that stream items do not arrive in a particular order. This model has been used, among other applications, to monitoring the behavior that specific IP addresses have.

- Turnstile Model: This is a general version of cash register model. In turnstile model each X_k in F is a positive or a negative increment of X_{k-1}. In this way, $X_k = X_{k-1} + Inc_k$, where Inc_k is the stream produced in the time T_k. That means, that items could be both inserted or deleted. When $Inc_k > 0$, the model is called strict turnstile, and non-strict turnstile when it also admits negative increments. This model has been used to mining frequent items in the stream.

2.2 Mining Data Streams

Data mining refers to the processing of data in order to discover non-obvious knowledge. This is made by finding and adjusting a model that represents their characteristics, allowing make descriptions or predictions about the data [44]. Mining data streams or streaming mining is the process of applying data mining to data streams [79]. However, conditions for applying mining process to data streams are different from those required for conventional data.

Conventional mining uses datasets residing in hard drive, allowing multiple and complete readings of them. Data streams volume is large and its generation rate is so fast and variable that is not feasible to use those procedures that use the data many times. It is not possible to store and process all data efficiently. For this reason, mining algorithms must be designed or adapted to work with data in just one pass [42]. Additionally, design of algorithms must be aimed to model changes and they must be obtained as a result of mining process, which implies that models must be dynamic due to data tend to evolve over time (implicit temporal component that exists between a datum and another) [2].

Those conditions impose on systems that work with data streams some restrictions. These restrictions are listed below [41, 153, 89]:

- Data streams have, potentially, an unlimited size.
- It is not feasible to store all the data. When a datum is processed must be discarded.
- Data come online, and they are continuously generated.
- There is no way of controlling the order in which data arrives.
- Data in the stream have a temporal correlation which marks a specific order of arrival, for each one.
- Data are highly changeable and presuppose an implicit evolution over time.

The above represents some issues in data streams such as memory management; problems related with the optimization of the memory space consumed and data structure representation; and the time consuming by the pre-processing data, which has to be integrated in the mining techniques.

2.2.1 Clustering Algorithms Taxonomy

In data mining there are some different tasks to process data, such as association, classification and clustering. Clustering is an unsupervised learning technique that organizes data into well separated groups without prior knowledge. Clusters are formed in such way that data assigned to the same group are similar while data of different groups have low similarity. There are many ways of clustering data which can be classified into: hard and fuzzy clustering problems. In the first one, a data belongs only to one cluster. In the second, a data may belong to two or more clusters with certain probability. Hard clustering problems can be divided into partitional, hierarchical, density-based, grid-based and hybrid algorithms [78, 63].

- Partitioning algorithms produce a partition of data into clusters by applying a criterion function defined on the dataset (usually the squared error criterion) [46, 40]. Algorithms such as K-means, K-medians and CLARANS (Clustering Large Applications based on RANdomized Search) [130] are examples of partitioning algorithms.

- Hierarchical algorithms produce a dendrogram that represents data clusters connected to different levels of similarity. Most of these algorithms are variations of "single link", "complete link" and "minimum variance" algorithms [78]. Hierarchical algorithms can be divided into divisive and agglomerative algorithms. In particular, the agglomerative method or "bottom-up" begins with a group for each value and then joint the groups according to their similarity to obtain a single group or until achieve the stop criterion. AGNES (AGglomerative-NESting) [12] and IHC (Incremental Hierarchical Clustering) [166] are agglomerative algorithms.

- Density-based algorithms produce groups associated with dense regions. Clustering is done when the density in the neighborhood is within a certain threshold or when new data are connected to others through their neighbors. DBSCAN (Density Based Spatial Clustering of Applications with Noise) [47], DENCLUE (DENsitybased CLUstEring) [83] and UNC (Unsupervised Niche Clustering) [108], are density-based algorithms. A review and a comparison of this kind of algorithms can be found in [9, 10].

- Grid-based algorithms divide the data space and produced a grid data structure formed by a number of cells. Clustering is dependently of the number of cells grid and independently of the number of data. Among grid-based algorithms are STING (STatistical INformation Grid-base) [164] and OptiGrid (Optimal Grid-Clustering) [84]. A review and a comparison of this kind of algorithms can be found on [11].

- Hybrid algorithms use other techniques, or a combination of them, to find the clusters. Among hybrid algorithms are CLIQUE (CLustering In QUEst) [8], that use cells to divide the data space; COBWEB [52] and AutoClass [27], that create statistical models to establish the best description of the groups in the dataset; Gravitational Clustering [70], that is inspired by the force of gravity as a mechanism for group the data.

2.2.2 Incremental Clustering Techniques for Data Streams

Clustering techniques are widely used, however, they are limited when datasets are too large, in terms of computational cost, especially memory. Incremental clustering techniques attempt to solve this problem by working with portions of data: given a dataset in a

metric space, groups are formed by the way in which data are arriving. When new data are observed they can: a) be assigned to existing groups, b) generate new groups, c) join some groups [53, 91, 143]. Among methods that use incremental clustering technique are:

- Divide and Conquer Method: Algorithms developed with this method take portions of data and apply a clustering algorithm to each partition, one at a time. These algorithms can develop a structure to store information from each partition. This method implies that an iteration of the clustering algorithm must be applied to each partition separately. The iteration must generate representative prototypes of data portion used in it, which will be use as a starting point for the next iteration. These algorithms must learn or extract knowledge from analyzed data without knowing the entire dataset. INCRAIN algorithm [72] uses this technique. This algorithm is the incremental version of RAIN algorithm [71], which is based on the mass concept and its effect on the space. Mass concept is used to delegate a data as a representative of a data group.

- Simple Observation Method: This method allows to analyze data only once. When a data is analyzed, then a) a new group is created or b) the data is assigned to an existing group. There are variations of this mechanism, some algorithms maintain data in the group representation, others develop prototypes that represent the analyzed data (groups). The disadvantage of the first variation is that storage space increases with the number of data. Additionally, these algorithms make not distinction between old and new data, which means that they cannot handle data evolution (when new groups emerge previous groups disappear). STREAMS algorithm [74] uses simple observation method. This algorithm minimized of sum of squared distances without differentiating between new and old data.

A common way to analyze data streams, involves using clustering techniques. Usually in these techniques, a group is represented by an element, commonly called "centroid" or "prototype" when is numeric and "medoid" when is categorical [17]. The challenging of clustering techniques applied to data streams, consists mainly in: a) determining the prototypes of the groups when a new data arrives. These techniques must taking into account that data cannot be used more than once [4]. b) Incorporate a mechanism, that allows to forget portions of data stream and focus on those that are new.

The clustering problem is a difficult task to solve in the domain of data streams due to traditional algorithms become too inefficient in this context. Single-pass algorithms, that have been recently developed for clustering data streams, have particularly two issues: first, groups quality is poor when data tend to change continuously over time; second, algorithms do not include in its functionality a way to explore and discover new groups in different stream portions.

These issues make that clustering algorithms have special requirements when deal with data streams: compactness representation, fast and incremental processing of new data points and, clear and fast identification of the outliers [17, 96].

- Compactness Representation: Due to the arriving data rate, it is not feasible to have a large description of the clusters. Those descriptions grow when new data arrives. For that reason, a compact clusters representation, that does not grows so much, is needed (the linear growth is still high).

- Fast and Incremental Processing: Data streams online nature makes this feature be obvious. Function responsible for deciding the place for the new data should not make comparisons between new elements and all processed data. This function must use the proposed data representation. The assignment of the place for the new data, must have a good performance, that means, good complexity. Additionally, algorithms should to re-learn any recurrently occurring patterns.

- Clear and Fast Identification of Outliers: Outliers are data that do not fill well in any created cluster. Algorithms must be able to detect those data and decide what to do with them. This depends on how many outliers have been seen and on the kind of applications to which data belong. Then, the function must decide if: creates new clusters, redefines the boundaries of existing clusters or ignores the data.

2.3 Clustering Data Streams

2.3.1 Online Preprocessing

Activities used in the mining phase, such as understand, transform and integrate data (pre-processing) are a key part in the process of knowledge discovery (KDD) [67]. Some techniques have been considered to reduce the processing load and improve performance in data streams [3]. As is shown in Figure 2.1 those methods can be classified into data-based and task-based techniques [61, 89, 97].

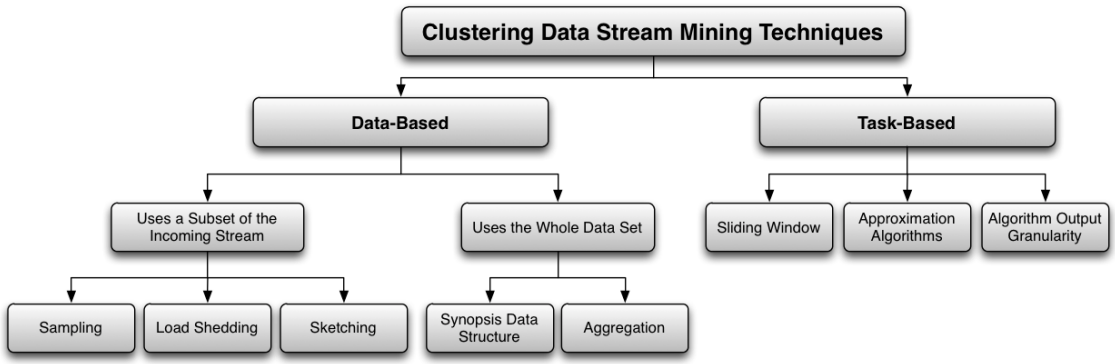


FIGURE 2.1. Clustering Data Stream Mining Techniques

2.1.3.1 Data-Based Techniques

These techniques aim to create an approximate data representation of small size by either of the following: examining only a subset of the dataset or summarizing the dataset. Among these techniques are sampling, load shedding, sketching and aggregation methods. Figure 2.2, shows some works developed in each of these methods. Dotted lines in the figure represents a relation between the works with others data streams algorithms. If the relation is with a classical data clustering algorithm its name is shown in italics.

- **Sampling:** This method selects statistically a subset of the data to be processed. Sampling is the simplest among all other methods of construction of data streams synopsis.

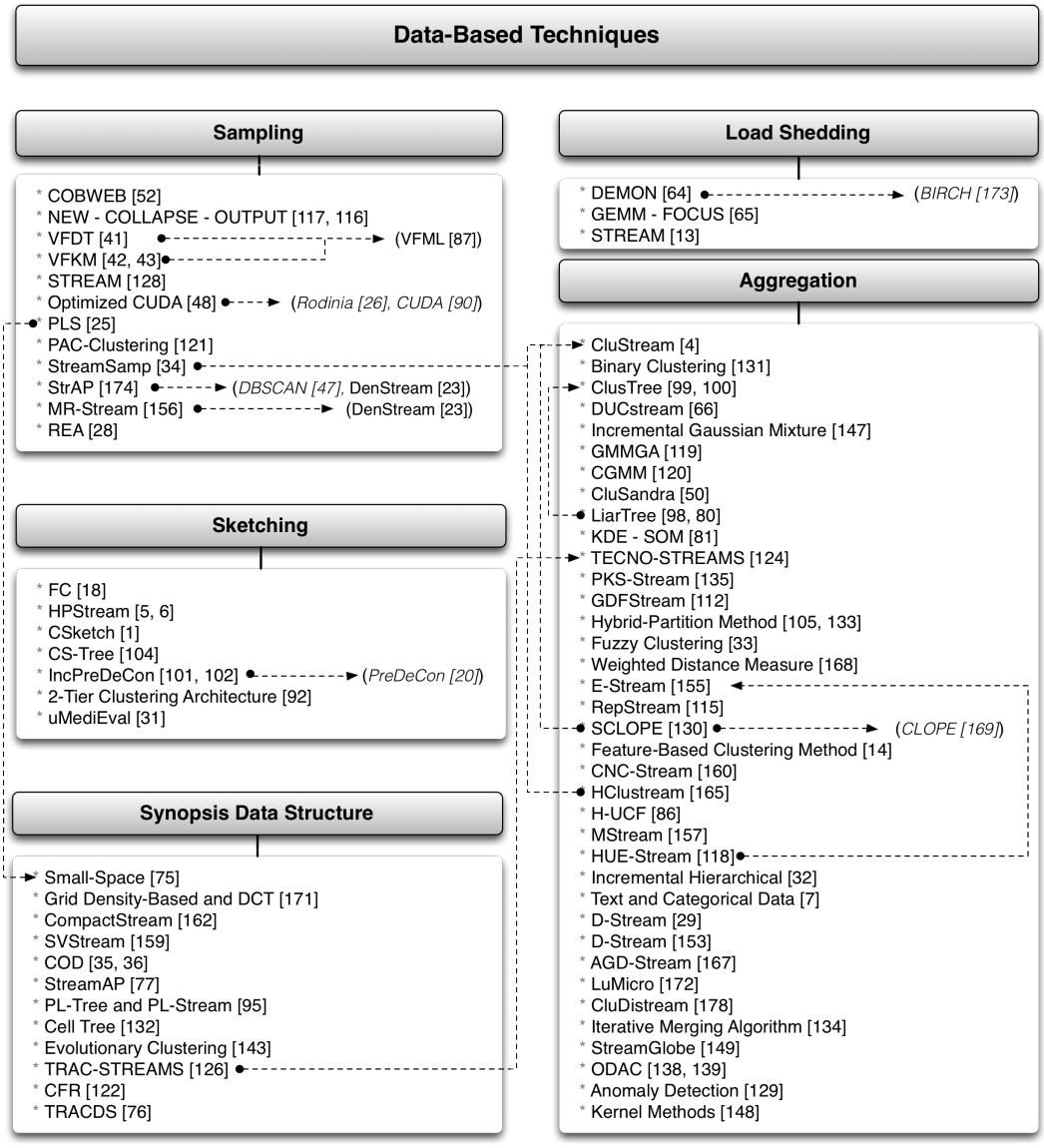


FIGURE 2.2. Data-Based Techniques for Clustering Data Streams. Dashed lines indicate the algorithms in which the presented algorithm are based. Algorithm names in brackets belong to task-based techniques. Algorithm names in brackets and italics belong to traditional data clustering.

The method is easy, efficient and has direct application in data mining and database operations due to it uses the original representation of records. As data volumes are so large, sampling is a very important step in processing. As the dataset size is unknown, this method requires a special analysis to find error bounds (computation error rate). On the other hand, sampling does not address the problem of fluctuating data rate and this is a poor technique for anomaly detection. Some developed works with this technique are shown below.

To deal with the performance of the system Fisher [52], has proposed *COBWEB* a hierarchical clustering algorithm which uses category utility function to form a tree and classify the data. Each node in the tree has a probabilistic description that summarized the kind of objects that compose it. When new data are locate in the tree, node statistics are updated. Manku *et al.* [120, 119] proposed, as a part of a framework, the *NEW, COLLAPSE and OUTPUT* algorithms, which approximate quantiles based on non-uniform random sampling technique, in order to analyze unknown length of the streams, and estimate extreme values in order to calculate memory requirements.

Approaches proposed by Domingos and Hulten, *VFDT* (Very Fast Decision Tree Learning) [41] and *VFKM* (Very Fast K-Means) [42, 43], deal with training time, memory requirements and learning loss. VFDT is based on a decision-tree method called Hoeffding tree. In this algorithm, Hoeffding bounds are used to estimate the size of sub-samples in order to decrease the training time. In the learning phase, VFDT only needs to see each example once, avoiding store them in memory, such that required space is used for the tree and the statistics. VFKM determines the upper limit for learning loss as a function that depends on the number of data analyzed at each step of the algorithm.

VFDT and VFKM algorithms are based on the general method for scaling up machine learning algorithms on K-means, called VFML (Very Fast Machine Learning). The implementation of these algorithms is available in a toolkit which is presented in [87].

Reducing memory requirements is addressed by approaches such as *STREAM*, presented by O'Callaghan *et al.* [131]. This algorithm processes data in chunks. Each chunk is clustered using LOCALSEARCH and calculating the median of each cluster as the sum of the elements in each chunk. In the memory is only maintained the weights of the cluster centers and LOCALSEARCH is again applied to that set, in order to obtained the weighted centers of the stream.

A framework that implements an OpenCL for *SC benchmark* data streams clustering (a suite that provided parallel programs for the study of heterogeneous systems) [26], is presented by Fang *et al.* [48]. The proposed framework has some memory optimizations over CUDA algorithm [90] and uses a model-driven auto-tuning to maximize the clustering performance.

To deal with space reduction problems and the increasing approximation factors, Charikar *et al.* [25] presents the *PLS* (Polylogarithmic Space) algorithm, as an extension of the Small-Space algorithm presented in [75]. PLS aims to maintain an implicit lower bound for the clusters.

A model that improves the running time, is proposed in [124]. This learning model called *PAC-Clustering* (Probably Approximately Correct Clustering) uses the PAC learning model. In the new model, the running time depends of the diameter of the data.

Among the algorithms that handled with dimensionality problems is *StreamSamp* [34], a one pass algorithm, inspired on two steps CluStream algorithm [4]. *StreamSamp* summarized data streams independently of its dimensionality using k-means algorithm.

Approaches that treat with data streams representation, cluster resolutions and imbalanced data streams are, respectively: *StrAP* (Streaming Affinity Propagation) [177], that is based on both DBSCAN [47] and DenStream [23]. StrAP extracts the best representation of the dataset for which is used the message passing method, and stores outliers. *MR-Stream* [159] is an extension of DenStream [23], which uses a grid of cells that are dynamically subdivided by a tree data structure. This algorithm, updates synopsis information in constant time, discovers clusters at multiple resolutions, generates clusters of higher purity, and determines the right threshold function, for density-based clustering, based on the data streams model. *REA* (Recursive Ensemble Approach) [28] attempts to deals with imbalanced data streams using k-nearest neighbor algorithm.

- **Load Shedding:** The purpose of this method is to increase the monitoring system performance. It is expected that the rate at which tuples are processed be at least equal to the rate in which new tuples arrive to the data stream. Unprocessed tuples are deleted to reduce the system load. This technique has been used in querying data streams, although it sacrifies the accuracy of the query answers. Load shedding has the same problems of sampling. Some developed works with this technique are shown below.

  In this method, approaches for data streams monitoring and managing are proposed. Among them are *DEMON* (Data Evolution and MOnitoring) proposed by Ganti *et al.* [64], which is an extension of BIRCH [176]. *DEMON* goal is the maintenance of frequent item sets and clusters algorithms that evolve through systematic addition or deletion of data blocks. In the model is included a new dimension called data span, which allows selections of the temporal subset of the database, defined by the user.

  Ganti proposed in [65] two algorithms that uses insertions and deletion of data blocks: *GEMM*, which works with unrestricted window option, and *FOCUS* that uses techniques such as deviation in the dataset. Another proposed work is *STREAM* [13], a general-purpose prototype data stream management system, that is a system supports query plans and constraints over data streams such as clustering and ordering.

- **Sketching:** "Sketches" are an extension of random projection technique in the domain of time series. An input data of dimensionality d is reduced to an axes system of dimensionality k, choosing k random vectors of dimensionality d and computing the dot product for each vector. This method has been applied in comparing different data streams and it has been shown an improvement in algorithm running times. The accuracy is the major problem of this method and its application in the context of data stream mining is hard. Algorithms based on sketching have been successfully used in distributed computation over multiple streams. Some developed works with this technique are shown below.

  Some approaches to deal with high dimensionality using this method have been proposed. Among them are *FC* (Fractal Clustering) proposed by Barbará and Chen [18], that defines groups as a set of points with high similarity, and assigns new points to the group with minimal fractal impact. The fractal impact is measured in terms of the fractal dimension and it is related to the scale parameter or data contamination rate. *HPStream* (High-dimensional Projected Stream) [5, 6], introduces projected clustering methods which deal with the problem of sparsity in high-dimensional space. The projection allows to update the dimension of each cluster such as its evolve over time. The algorithm uses statistical representations of the elements in the clusters in order to update data streams in a fast way (such clusters are named projected clusters). To integrate historical and actual data, the algorithm uses a fading cluster structure.

Another works to deal with clustering of high-dimensional data streams are: *CSketch* [1], a method for massive-domain clustering (data domains with possible very large number of values). This model reduces the data dimensions in order to maintain intermediate statistics, and assign data to clusters. *CS-Tree* (Cluster-Statistics Tree) [104] proposed a scalable algorithm in which the multi-dimensionality problem is changed to a one-dimensional problem. *IncPreDeCon* [101, 102] is an incremental version of clustering density-based algorithm PreDeCon [20]. In this approach the problem of high dimensionality on dynamic data is treated by the clusters update, when new data arrives.

Among models for address with problems of arbitrary shapes and distribute data streams are *non-linear two-tier architecture* [92]. In the first tier, kernel methods have been used in order to partitioning the stream in the feature space. In the second tier, stream is projected into low-dimensional space (LDS) representation. *uMediEval* [31] is an algorithm to process distributed location streams with redundancy and inconsistency. This is based on min-wise hash in order to obtain uniform samples in distributed systems. The algorithm provides an approximate k-median of the moving objects.

- **Synopsis Data Structures:** The construction of the synopsis, requires a summarization techniques on incoming streams. Histograms, wavelet analysis, quantiles and frequency moments have been used to construct synopsis. The problem of this technique is that the synopsis does not represent all the features of dataset. However, at high rates of incoming streams this technique should be not enough to summarize the data. Some developed works with this technique are shown below.

  Approaches that treat with space and memory issues have been suggested in works such as the proposed by Guha *et al.* [75, 74] in which an adaptation of k-median and k-means algorithms are developed. These algorithms are based on divide and conquer method. They use a constant factor in order to make one pass over the data and to control the large space requirements. The algorithm proposed in [74] makes a single pass over the whole data using *Small-Space*.

  In the same line has been proposed *Grid Density-Based and DCT* algorithm [174]. This is a grid density-based algorithm that is able to detect clusters of arbitrary shapes and that uses DCT (Discrete Cosine Transform) to identify clusters and to compress the data. *CompactStream* [165] deals with storage space issues through an on-line and off-line phases. In the on-line phase, data are summarized in micro-clusters of features in form of hyper-elliptical shape. In the off-line phase, the summary is processed. *SVStream* (Support Vector based Stream Clustering) [162] is based on support vector domain and clustering, in which data are mapped into a kernel. This algorithm uses multiple spheres in which support vectors are used to construct the clusters boundaries. Outliers are detected and removed.

  Among approaches to deal with dynamically multiple data streams is *COD* (Clustering on Demand framework) [35, 36]. This framework has on-line and off-line phases. In the first, it is maintained the summary hierarchies of the data. in the second phase, an adaptive clustering algorithm is used to retrieve the approximations of the data streams. Summarization is based on wavelet and regression analyses.

  To deal with the problem of clustering distributed data streams, in [77] is proposed *StreamAP*, an algorithm that uses propagation techniques in two levels. The first level is the node level on the batch of data. The second level is in which data synopsis is derived. An on-line version of affinity propagation algorithm is used locally on each node.

High-dimensional data streams issues are handle in works such as the proposed in [95] which consists of two algorithms: *PL-Tree* that store data stream summary information and *PLStream* that cluster high dimensional data streams on PL-Tree using damped window and pruning list tree. PLStream considers high dimensionality, space scalability, long running time and high memory consumption issues. *Cell Tree Algorithm* [135] is an approach that maintains distribution statistics from arriving data streams. This algorithm uses a partitioning factor and a partitioning threshold in order to divide the grid-cell in equal and small grid-cells. A structure list is used as an index to manage and locate the grid-cells in a one-dimensional data space. Another algorithm to evolutionary clustering using frequent itemsets (sets of words which occur frequently) is proposed in [146]. Outliers removal and dimensionality reduction are take into account in it.

Evolving and noisy data streams issues are treated in *TRAC-STREAMS* (TRacking Adaptive Clusters in data STREAMS) [129]. This algorithm is a new version of TECNO-STREAMS [127] and it is based on analytical optimization instead of evolutionary computation. This approach uses different criterion function, that optimizes the synopsis of the stream in the presence of noise, focusing on more recent data.

Another kind of issues, such as related with initial values and criterion to combine clustering problems have been faced in [125]. In it *CRF* (Clustering using F-value by Regression analysis) is proposed. This algorithm includes regression analysis and cluster similarities using a criterion based on Maharanobis distance. Clusters are formed taking into account the center of gravity of them. In [76], a framework called *TRACDS* (Temporal Relationships Among Clusters for Data Streams) is presented. This uses the concept of temporal clusters to deal with the problem of temporal relation between clusters. This framework combines data stream clustering with Markov chains. In the algorithm first partitioning of data is made and then clusters are learned dynamically.

- **Aggregation:** In many computing problems is desirable to compute aggregate statistics (hierarchical) on data streams. Usually these statistics have estimates, frequency counts, quantiles and histograms. Working on data statistics brings more benefits than working directly with the data. Aggregation does not operate well on fluctuating data streams. Aggregation has been successfully used in distributed data streams environments. However, this technique does not capture all features of the dataset. Some developed works with this technique are shown below.

  Some approaches to deal with running time and memory efficiency have been proposed. One of them is *CluStream* proposed by Aggarwal *et al.* [4] that uses "Micro-Clustering" with the concept of pyramidal time windows. This algorithm divides the clustering into on-line and off-line processes. On-line process is responsible of maintain micro-clusters, which are structures that store a summary of statistics for each data stream before they have been discarded. Off-line process is responsible to create macro-clusters, which summarize data. Once the algorithm has computed a good number of micro-clusters, they are considered as pseudo-points. This is a modified version of the k-means algorithm. Micro-clusters are grouped to identify groups at higher level.

  To reduce running time, Ordoñez [134] presents a *Clustering Binary Data Streams* algorithm, which has several improvements of k-means algorithm. Among those improvements are include online k-means, scalable k-means and incremental k-means algorithms. The proposed algorithm, updates the cluster centers and weights after the analysis of each batch, instead to be one at time. *ClusTree* [99, 100] is a free parameter algorithm that automatically adapts the data streams speed. The algorithm has a hierarchi-

cal structure that can adapts dynamically the clusters sizes model to the speed of the stream. The algorithm incorporates local aggregates (temporary buffers) and it is based on micro-clusters representation.

To address problems of memory management, there are several works such as *DUC-stream* [66], an incremental single pass algorithm which receive data in chunks. In this a density grid is defined and considered as a dense unit depending on a threshold. Clusters are identified as a connected components of a graph in which vertices are the dense units and edges are common attributes between two vertices. In [150] is proposed *Incremental Gaussian Mixture Model*, a probability-density-based data stream clustering approach by merging components in Gaussian Mixture Model. This algorithm does not store historical data, instead statistical structures are used only in new data. Clustering merging strategy and expectation maximization are used in the algorithm. *GMMGA* (Gaussian Mixture Model with Genetic Algorithm) [122] determines the number of Gaussian clusters and the parameters of them in arbitrary shapes using genetic algorithms and fuzzy clustering estimation method.

To deal with large and quickly streams speed has been proposed *CGMM* (Compound Gaussian Mixture Model) [123], which is a semi-supervised model. This model finds and merges Gaussian components using the Expectation Maximization algorithm to initialize the proposed model.

Some works, that have been proposed to cope with memory requirements, computational cost and processing time, are *CluSandra* framework [50] that includes an algorithm for cluster evolving data streams. This algorithm uses the product of the micro-clusters real-time statistical summary. *LiarTree* [98, 80], an extension of ClusTree [99, 100] algorithm, deals with the overlapping of past entries, incorporates noise handling and allows the transition from noise to novel concepts. *KDE-SOM* [81] a kernel density estimation (KDE) method is based on self-organizing maps (SOM), that are generated on the data streams to obtain summaries of them. The algorithm has a phase for learning in which the SOM is builds to summarize the data, and a phase for analysis in which probability density functions are analyzed on arbitrary periods of time.

Among proposed approaches to deal with the data streams dimensionality is *TECNO-STREAMS* (Tracking Evolving Clusters in NOisy Streams) presented by Nasraoui et. al. [127]. This is a scalable algorithm for clustering multi-dimensional data streams that uses artificial immune systems as a type of evolutionary optimization of a criterion function. An important feature of this algorithm is the incorporation of temporal weights that allow to forget, gradually, portions of data stream, and to focus on those that are new. However, this algorithm does not give the memory bounds.

With the same purpose above, have been proposed *PKS-Stream* [138], that is based on grid density and on the index Pks-tree. The algorithm deals with high dimensional data, and counts with on-line and off-line phases. In the on-line phase, new data are mapped to the related grid cells in the Pks-tree. If there is a grid cell for the data, it is inserted. Otherwise, a new grid cell in the tree is created. In the off-line phase, the grid density is checked and clusters are created or updated. In *GDFStream* (A Grid Fractal Dimension-Based Data Stream Clustering) [114] is incorporated a Grid method and the Fractal Clustering methodology to clustering data streams. The algorithm has on-line and off-line component. In the first, summary of the statistics are stored. In the second, final clusters are generated using the statistics.

Another method that deals with multidimensional data streams is *Hybrid-Partition Method* [105]. This is a grid-based statistical clustering algorithm, which allows three different ways of partitioning a dense cell: based on the average, based on the standard deviation, or choosing the more effective amount, among previous two. Partitions have initially the same size and they are divided based on their distribution statistics. In [136] the earlier work is used, in order to audit data streams. Features are identified and represented by the algorithm, and each cluster represents the frequency range of activities respect to the feature.

To clustering dynamic data streams, a methodology that uses *Fuzzy Clustering* is presented in [33]. The methodology deals with the movement, creation and elimination of classes, updating the previous system with the new data.

Among approaches that aim to monitor data streams is the work proposed in [171]. In thisan algorithm that uses *Weighted Distance Measure* of snapshot deviations as the distance measure between two streams, is presented. This algorithm can reflect the similarity in the data values but ignores the trends in the streams. *E-Stream* [158] is another algorithm to credit fraud detection and network intrusion detection. This supports the evolution of the data streams, and classify them in: appearance, disappearance, self-evolution, merge and split. Each cluster is represented as a Fading Cluster Structure (FCS). The algorithm uses a histogram for each feature in the dataset. *RepStream* [118] is proposed for network packet inspection and financial transaction monitoring. This is an incremental graph-based clustering algorithm that uses the graphs to model the spatio-temporal relationships in the data. Each cluster has exemplar data to capture cluster stable properties, and predictor points to capture evolving properties of the data.

To address issues related with numerical, categorical and mix data, have been proposed *SCLOPE* [133]. This is an algorithm for clustering categorical data streams that uses pyramidal timeframe and micro-clusters. SCLOPE is based on CluStream [4] and CLOPE [172] algorithms. For numerical data streams in [14] is presented a *Feature-Based Clustering Method*. This algorithm uses k-means to select the initial centers of clusters, and two measures to rank features. In this algorithm, unimportant features are removed.

Another proposed algorithms that have been proposed to deal with different data types, are *CNC-Stream* (Clustering data Stream with Numeric and Categorical attributes) [163], that is composed of two parts. An on-line part, for micro-cluster maintenance using Incremental Entropy Cost (IEC) and Merging Entropy Cost (MEC). An off-line part, to generate final clusters. *HClustream* (Heterogeneous CluStream) [168] is an algorithm based on CluStream [4], in which k-means clustering is modified with k-prototypes clustering. *H-UCF* (Heterogeneous Uncertainty Clustering Feature) [86] uses the probability frequency histogram for tracking statistics of categorial data. This algorithm has two phases. In the first phase, cluster candidates are determined. In the second phase, new data are assigned to the created clusters. *MStream* [160] uses micro-clusters to store statistical information, in the on-line phase. In the off-line phase, micro-clusters are merged to form macro-clusters. *HUE-Stream* [121] is an extension of *E-Stream* [158]. The algorithm supports numerical and categorical data, and includes a distance function, cluster representation, and histogram management.

Clustering documents are treat in [32], a framework for text patterns identification. The keywords definition are defined using clustering and an incremental hierarchical clustering algorithm. Nearest neighbor is used to manage the high rate of arriving

documents. In Aggarwal [7] is presented an algorithm that uses statistical summary data to cluster text and categorical data streams.

Arbitrary shapes and handle outliers are addressed in works such as *D-Stream* [29], that uses density-grid approach to clustering data streams. The algorithm does not need to know the number of clusters. This algorithm uses a decay factor to the density of each data point in order to form clusters dynamically. In this scheme, most recent data have more weight. Another algorithm with the same name, *D-Stream* [156] is proposed as a framework that uses density-based approach. In this, each data stream is stored in a grid, in an on-line component. The density of the grid and clusters are computed, in the off-line phase. The grid attraction concept is used to manage the close of the data, in order to improve the quality of the algorithm. *AGD-Stream* (Active Grid Density Streams) [170], finds clusters in arbitrary shapes and deals with the problem of boundary points. Data are mapped on a grid structure using the density concept.

Uncertain data streams are treat in [175], in which two phases algorithm called *LuMicro* (Low uncertainty Micro) is proposed. On-line phase, uses micro-clustering, and off-line phase, uses macro-clusters. The appropriate cluster is selected taking into account the position and the uncertainty, which are calculated based on its probability distribution.

CluDistream framework [181] is proposed to deal with distributed data streams. The algorithm uses Gaussian model based on Expectation Maximization (EM) to clustering distributed data streams with noise or incomplete data. A test-and-cluster strategy is proposed in order to capture the distribution of data stream at remote site and to deal with different data streams lengths. In [137] is proposed an iterative merging algorithm, that produces a partitioning of the data and admits creation, addition and merge clusters, using MDL to decide. The algorithm also uses the Euclidean distance, to find time series, and the nearest neighbor, to find the most similar subsequence in the input time series.

StreamGlobe [152] and *ODAC* (Online Divisive-Agglomerative Clustering) [141, 142], have been proposed to clustering data streams in ad-hoc networks and sensor networks, respectively. In the first work, the system meets challenges of efficient querying data streams in an ad-hoc P2P network scenario. This system uses clustering to identify reusable existing data streams in the network. This process transforms data streams into a structural aspect represented by their respective properties. The second work is a system for on-line prediction of all the variables in large sensor networks. The proposed algorithm is a hierarchical tree-shaped structure of clusters and uses top-down strategy. *ODAC* uses an incremental dissimilarity measure between time series. The algorithm counts with two main operations: expansion and aggregation, that are performed based on the clusters diameters.

Among other applications, it is possible to find a work for anomaly detection, presented in [132], in which clusters are dynamically generated based on a minimum deviation, then the cluster can be splitting or merging with adjacent clusters. For discover geophysical processes, in [151] is proposed an algorithm that uses kernel methods to discover snow, ice, clouds, among other processes. Authors of this work, introduce the Gaussian mixture model with spherical covariance structure as a kernel.

Task-Based Techniques

These techniques adopt or create algorithms in order to achieve efficient solutions in time and space. In general, methods that belong to this technique do not need to store data.

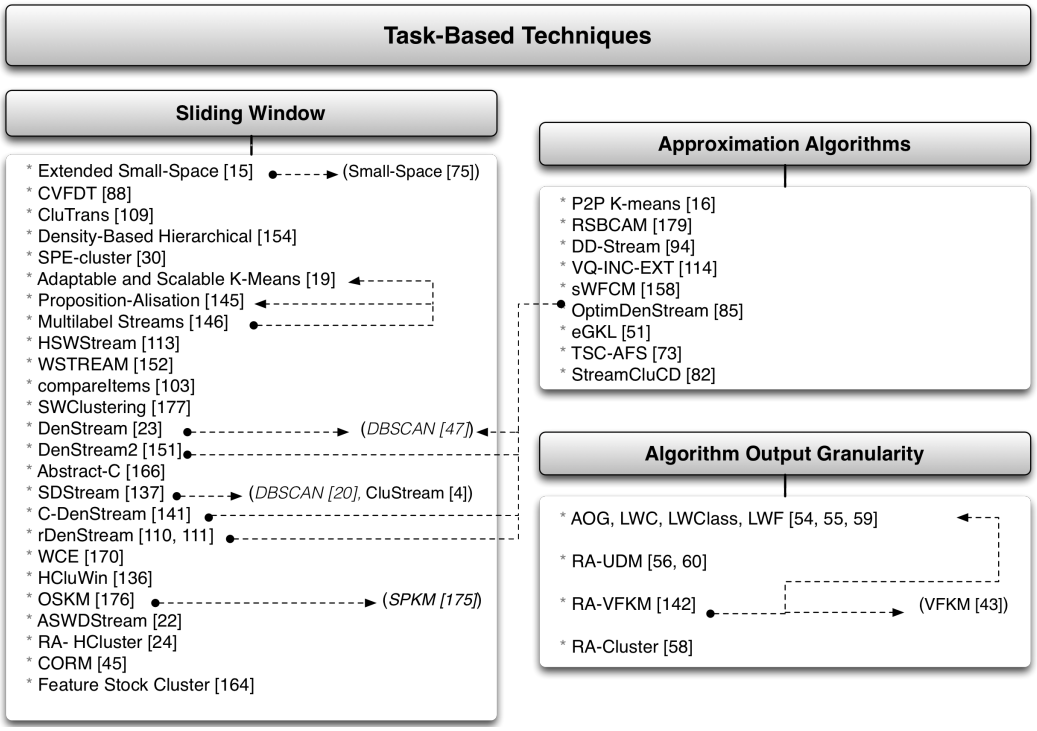


FIGURE 2.3. Task-Based Techniques for Clustering Data Streams. Dashed lines indicate the algorithms in which the presented algorithm are based. Algorithm names in brackets belong to data-based techniques. Algorithm names in brackets and italics belong to traditional data clustering.

Among these methods are approximation algorithms, sliding windows and algorithm output granularity. Figure 2.3, shows some works developed in each of these methods. Dotted lines in the figure represents a relation between the works with others data streams algorithms. If the relation is with a classical data clustering algorithm its name is shown in italics.

- **Approximation Algorithms:** This method considers mining algorithms as hard computational problem, designed with error bounds. These kind of algorithms are efficiently in running time, however they do not solved problems associated with data rates and available resources.

  In this method, approaches for distributed data streams, evolutionary clusters, memory and time responses have been proposed. Among them, in [16] is presented an adaptation of k-means algorithm for distributed data streams peer-to-peer, such as sensor network environment, called *P2P K-means*. The algorithm uses statistical bounds for estimate the centroids error of clusters.

  RSBCAM (Rough Set Based Clustering Algorithm for Multi-stream) [182] has been proposed to mining multiple data streams. This algorithm taking into account the quality and efficiency. *RSBCAM* calculates equivalence relations and similarity between clusters in the initial phase. Then similarity is used to merge the clusters. K-means algorithm is used to adjust dynamically clustering results.

  Among the works to deal with evolutionary clusters are *DD-Stream* [94], a framework that has an algorithm for density-based clustering data streams in grids. The clustering quality is improved with the *DCQ-means* algorithm, which extracts border points of the

grid. The framework has an on-line and off-line phases. In the first phase, new data are read and mapped in the grids. In the second phase, clustering on grids using *DCQ-means* algorithm is performed. Another approach is *VQ-INC-EXT* [116], an incremental clustering method for data streams based on *VQ* (Vector Quantization). To adjust the number of clusters adaptively during incremental clustering is used a sphere of influence in the clusters, as well as some operations, such as deletion of clusters satellites and a split-and-merge strategy.

Some approaches that address memory and time response issues, are*sWFCM* algorithm [161], that is an extension of *FCM* (Fuzzy C-means Algorithm). In the extended algorithm, clusters are treated in a fuzzy way, renewing the centers weight of them. In [85], is proposed *OptimDenStream*, an extension of *DenStream* algorithm [23], that uses the Hoeffding theory to improve the detection time strategy. In the algorithm, a double detection time strategy is used, and the detection time for potential micro-clusters is deleted.

GK-like (eGKL) [51], an extension of *Gustafson- Kessel (GK)* algorithm is proposed to treat whit clusters that have different shapes. This algorithm is based on *k-nearest neighbor* and *LVQ (Linear Vector Quantization)* algorithms.

Text and categorical data issues have been addressed by works such as *TSC-AFS* [73], a text stream clustering based on adaptive feature selection, that uses two thresholds to evaluate the index level. *StreamCluCD* [82], a clustering algorithm for categorical data. This algorithm creates histograms and applies approximation counts technique over data streams to keep the frequencies value. The algorithm uses Lossy Counting Algorithm, and accepts a support threshold and an error parameter.

- **Sliding Window:** This technique is considered as an advanced technique in data streams query. Sliding windows transform a given dataset containing time series, into a new set of simple examples. For this purpose, windows with a certain size are moved through the series, the horizon attribute value is used as a prediction label that is extended to the bottom of the window. Most recent elements are called active and the rest are expired elements. This technique does not provides a model for the whole data stream. Some developed works with this technique are shown below.

  To address issues related with space and memory, Babcock *et al.* [15] proposed an extension of the work presented in [75]. In the extension is included the sliding window model, maintaining an exponential histogram for the active elements. Another extended work of [75], that improves the training time, memory requirements and learning loss is *CVFDT* (Concept-adapting Very Fast Decision Tree Learner) [88]. This is a longer version of *VFDT*, in which a decision tree is maintained, and sliding windows with a fixed size is added, to acquire the ability to learn and respond to dynamic changes in the data. This algorithm has high accuracy and low complexity.

  Another works in this line are *CluTrans* [111], for clustering transactional data streams, which uses the *INCLUS* algorithm in elastic and equal-width time window. In [157] is proposed a *Density-Based Hierarchical* method, which uses variance within cluster, and density and distance intra clusters in the merging process. Three parameters, number of cluster of the final step, the minimum and maximum number of clusters are needed.

  In order to clustering multiple and parallel data streams, have been proposed *SPE-cluster* (SPectral Cluster algorithm) [30], in which multiple streams with similar behavior based on auto-regressive model, are clustered. For this purpose is used the estimated frequency spectrum (amplitude, frequency, damping rate) of the streams. In [19] is proposed

an adaptable and scalable on-line version of *K-means* algorithm using discrete Fourier transforms and a fuzzy version of on-line k-means to cluster parallel data streams.

An approach based on the work developed in [19] and on the incremental proposition-alisation algorithm [148] is proposed in [149]. This is an on-line version of k-means algorithm to deal with growing objects (objects that grow and change their definition over time), such as customers transactions. The algorithm transforms a multi-label streams in single streams. Another work for clustering multi-label streams is proposed in [148]. This model transforms streams in single-table streams, and maintains cache and sliding windows from them.

High-dimensional issues are treated in works such as *HSWStream* [115], in which projected clustering technique, and exponential histograms EHCF (Exponential Histogram of Cluster Feature) are used to deal with the in-cluster evolution, and eliminate the influence of old points. *WSTREAM* [155], an extension of kernel density clustering to mining data streams, uses d-dimensional hyper-rectangles to discover clusters which are incrementally adjusted to capture dynamics changes in the streams. *compareItems* [103] is a work that involves evolution of complex data, using a visual technique to see the evolution of the streams. The algorithm uses similarity function based on keywords comparison, in order to clustering articles intro threads. Additionally, introduces an elevance-based technique, which depends on aging, number of articles and duration. *SWClustering* algorithm [180], uses the Exponential Histogram of Cluster Features (EHCF) to cluster evolution. This algorithm eliminates the influence of old data improving the quality of the clusters.

To deal with clusters in arbitrary shapes and handle with outliers, have been proposed *DenStream* [23], which is based on DBSCAN [47]. The extended algorithm uses core-micro-cluster to summarized the clusters in an arbitrary shape. To determined the micro-clusters is uses the density. A modification of *DenStream* algorithm called *DenStream2* is presented in [154]. In this extension is included a time interval, a time windows size, controls for the dimension of data streams and a maximum number of micro-clusters. Another proposed work is *Abstract-C* framework [169], which includes a clustering algorithm density-based outlier in order to maintain compact summaries.

Another approaches proposed to treat with arbitrary shapes are *SDStream* [140] and *C-DenStream* [144]. *SDStream* performs density-based data stream clustering over sliding windows. The method adopts *CluStream* [4] algorithm and introduces the core-micro-cluster and outlier micro-cluster for on-line phase. Additionally, the method uses Exponential Histogram of Cluster Feature (EHCF) to store the clusters and outliers. *DBSCAN* [47] is used in the off-line phase to maintain the core-micro-clusters. *C-DenStream* proposes a semi-supervised method and uses the background knowledge of the data streams. The modifications are basically the adaptation of the off-line step of *DenStream* [23] and the inclusion of constraints in creation, maintained and deletion of micro-cluster.

Another extension of *DenStream* [23] is *rDenStream* (DenStream with retrospect) [112, 113], this is a three-step clustering algorithm with retrospect that learn about outliers. First and second phases are similar to *DenStream*, and in the third phase, the algorithm learns from discarded data. To deal with different temporal data representations has been proposed *WCE* [173] an algorithm that uses weighted clustering ensemble on different representations. This algorithm is composed of three modules: representation extraction, initial clustering analysis and weighted clustering ensemble.

To address with heterogeneous data streams has been proposed *HCluWin* [139], that contains both continuos and categorical attributes over sliding window. The *Heterogeneous Temporal Cluster Feature (HTCF)* and *Exponencital Histogram of Heterogeneous Cluster Feature (EHHCF)* are used to monitoring the distribution statistics.

OSKM (Online Spherical K-Means) algorithm [179] is proposed to mining text data streams. The algorithm is an extension of *SPKM (Spherical K-Means)* [178], and includes a modification of the scalable clustering techniques algorithm [49] based on the winner-take-all technique. Additionally, a forgetting factor is introduce in order to apply exponential decay to the history data.

Among some applications are *ASWDStream* [22], a clustering data stream algorithm applied to an intrusion detection. This algorithm uses micro-clusters, and is based on sliding and damped window techniques. *RA-HCluster* (Resource-Aware High Quality Clustering) [24] is used for clustering data streams on mobile devices. The algorithm is based on resources-aware technique and has on-line and off-line phases. In the first phase, the sliding window is used to create micro-clusters. In the second phase, hierarchical summarized statistics are used for clustering. For anomaly detection has been proposed *CORM* (Cluster based OutlieR Miner) [45], that uses *k-mean* to cluster the stream, which has been previously divided in chunks. The algorithm takes into account the outliers detected, and gives them the chance of survival in the next incoming chunk of data. In [167], is proposed a method to clustering data streams for stock data analysis, based on morphological characteristics.

- **Algorithm Output Granularity (AOG):** This technique is able to work with very high data rate fluctuations, depending of the available resources. AOG has three stages: mining, adaptation of resources and data streams rates. Then, generated structures are merge. This is a general approach that can be use in any mining technique such as clustering, classification and frequency counting. Some developed works with this technique are shown below.

  Issues related with limitations of computer resources are addressed by works such as *LWC* (one-look clustering) [54, 55, 59], in which an one-pass approach is used to clustering, classification and counting frequent items. The algorithm is adaptable to memory, time constraints and data streams rate. The authors proposed the use of data rate adaptation from the output side, using the *Algorithm Output Granularity (AOG)* [59].

  Based on previous works, in [56, 60] is incorporated an ubiquitous data mining (process of performing analysis of data on mobile), and a resource aware system (*RA-UDM*), which includes management for local resource information, context-aware middleware, resource measurements and solutions optimizer.

  To manage resource with fluctuating data streams (memory, time), Gaber *et al.* [57] have been developed the *LWC* (Light-Weight Clustering), *LWClass* (Lightweight Classification) and *LWF* (Lightweight Frequent items) algorithms, which are based on the *AOG*. *AOG* is able to manage resource with fluctuating data streams (memory, time). In *LWC*, *AOG* defined an adaptive threshold for the minimum distance among the center cluster and the data.

  In the same line, are proposed *RA-VFKM* (Resource Aware Very Fast K-Means) [145] and *RA-Cluster* (Resource-Aware Clustering) algorithm, proposed by Gaber and Yu [58]. In *RA-VFKM* is proposed to ubiquitous environments, using *AOG*, and an extension of *VFKM (Very Fast K-Means)* [43]. *RA-VFKM* is able to adapt variations in memory availability in mobile devices. Additionally, this algorithm increases the values

of allowable error and probability, which compromises the accuracy of the results, but enables the convergence, avoiding the total execution failure. *RA-Cluster* is a threshold-based micro-clustering algorithm, that is able to adapt the available resources using the *AIG (Algorithm Input Granularity)*, *AOG (Algorithm Output Granularity)* and *APG (Algorithm Processing Granularity)* algorithms, to adapt the availability of resources periodically, as an algorithm settings. Each previous algorithms are in charge of battery and bandwidth, and memory and processing power, respectively. *RA-Cluster* deals with unpredictable data streams rates, and resources constraints.

2.4 Evolutionary Clustering with Self Adaptive Genetic Operators Algorithm - ECSAGO

Evolutionary clustering algorithms deal with the problem of incorporating new data into previously formed clusters. Those clusters should be similar to those existing and should accurately reflect the new data. Among those kind of algorithms there is the Evolutionary Clustering with Self Adaptive Genetic Operators (*ECSAGO*), proposed by Leon et. al. [109]. This is a self adaptive genetic clustering algorithm which uses as basis the Unsupervised Niche Clustering (UNC) algorithm [128]. *ECSAGO* follows the same philosophy than UNC and improves some aspects. One of the most important is the inclusion of the Hybrid Adaptive Evolutionary Algorithm (HAEA), proposed by Gomez [68], which is a parameter adaptation technique for evolutionary algorithms. HAEA allows to *ECSAGO* adapting the genetic operator rates automatically while evolves cluster prototypes. Thus, crossover and mutation parameters are eliminated, and the best genetic operators to solve the clustering problem are selected.

ECSAGO has been extended to real representations and tested using synthetic and real data from anomaly detection applications and document clustering [106], [107]. Reported results show that the algorithm is able to detect the majority of clusters except when clusters have small size. *ECSAGO* model is shown in Figure 2.4

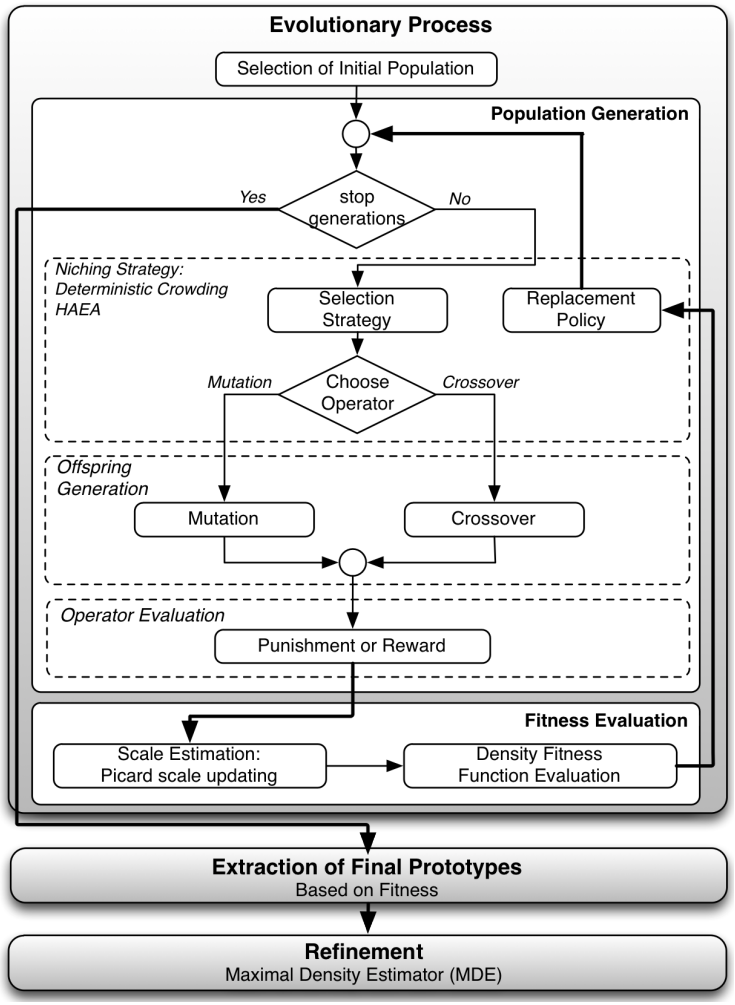


FIGURE 2.4. ECSAGO Model

As mentioned before, *ECSAGO* is based on UNC, an evolutionary clustering technique based on genetic niching optimization that is able to determine automatically the number of clusters and that presents robustness to noise. *ECSAGO* is divided in the same three stages that are proposed in the UNC model: *evolutionary process*, *extraction of the final prototypes* and *refinement of extracted prototypes*. These phases are described below.

Evolutionary Process

In the first step of this component, the population P for each generation G is initialized. P is composed by individuals p_i and each of them represents a cluster candidate (hypothetical cluster). An individual is determined by a center and a scale parameter σ_i^2. Then, niches are formed and maintained using Deterministic Crowding (DC) technique. DC is presented in the algorithm 1.

Algorithm 1 Deterministic Crowding

1: **Repeat for** G **generations**
2: **Repeat** $\frac{PopulationSize}{2}$ **times**
3: Select two parents p_1 and p_2 randomly without replacement
4: Cross them to produce children c_1 and c_2
5: Optionally apply mutation to produce children c_1' and c_2'
6: ***If*** $[d(p_1, c_1') + d(p_2, c_2')] \leq [d(p_1, c_2') + d(p_2, c_1')]$ ***Then***
7: ***If*** $f(c_1') > f(p_1)$ ***Then*** Replace p_1 with c_1'
8: ***If*** $f(c_2') > f(p_2)$ ***Then*** Replace p_2 with c_2'
9: ***Else***
10: ***If*** $f(c_2') > f(p_1)$ ***Then*** Replace p_1 with c_2'
11: ***If*** $f(c_1') > f(p_2)$ ***Then*** Replace p_2 with c_1'

DC technique selects two parents which are replaced by the respective child, which has been generated by applying crossover and mutation with some probability, if its fitness is greater than its parent fitness. Even though the improvement of each niche arises when members are near to the real peaks, DC allows to apply crossover between different niches.

UNC implements a mating restriction to avoid those interactions between different niches, however, *ECSAGO* does not use this restriction and implements another mechanism for selection and replacement by joining HAEA and DC. This is presented in the algorithm 2.

HAEA evolves each individual independently of the others. For each genetic operator is maintained a rate that represents the dynamically learned of them. In this way, in each generation only one operator is applied to the individual, which is selected based on its rates. If one parent is needed, it is selected without replacement. If two parents are needed, the second one is selected with replacement. If the offspring fitness is greater than its parents, then they are replaced and the rate of the applied operator is increases as a reward. In other case, replacement is not made and operator rate is decreases as a punishment. Operators have a real representation in *ECSAGO*, unlike UNC where the representation is binary.

The fitness evaluation is based on a density measure of a hypothetical cluster. This measure f_i follows a Gaussian Distribution and it is calculated respect to the candidate cluster c_i and its dispersion measure σ_i^2. Fitness is calculated in the following way:

$$f_i = \frac{\sum_{j=1}^{N} w_{ij}}{\sigma_i^2} \tag{2.1}$$

where:

- w_{ij} is a robust weight that measures how typical data point x_j is in the i^{th} cluster. In this way, the weight is high for points that are within the boundaries of the cluster and low for those points that are outside of the boundaries.

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right) \tag{2.2}$$

- d_{ij}^2 is the Euclidean distance from data point x_j to cluster center c_i, define as:

Algorithm 2 ECSAGO Evolutionary Algorithm: HAEA and DC Mixing

Evolution (δ, terminationCondition)

$t_0 = 0$
P_0 = initPopulation(δ)
while (terminationCondition(t, P_t) is false) **do**
  $P_{t+1} = \{\}$
  for each $ind \in P_t$ **do**
    rates = extract_rates (ind)
    δ = random(0, 1) ▷ learning rate
    oper = OP_SELECT (operators, rates)
    parents = Extra_Sel(arity(oper)−1, P_t, ind) $\cup \{ind\}$
    offspring = apply (oper, parents) $\cup \{ind\}$
    N = size(offspring) ▷ Initation of Replacement Policy
    x = ind
    min = $1e + 8$
    for $i = 1$ **to** N **do**
      if d(ind; offspring$_i$) > 0 and d(ind; offspring$_i$) < min **then**
      x = offspring$_i$
      min = d(ind; offspring$_i$)
      if f(ind) > f(x) **then** x = ind
        child = x ▷ Best child
    if (fitness(child) > fitness(ind)) **then**
      rates[oper] = $(1.0 + \delta) *$ rates[oper] ▷ reward
    else
      rates[oper] = $(1.0 - \delta) *$ rates[oper] ▷ punish
    normalize_rates(ratest)
    set_rates(child, rates)
    $P_{t+1} = P_{t+1} \cup \{child\}$
  $t = t + 1$

$$d_{ij}^2 = \|x_j - c_i\| = (x_j - c_i)^t (x_j - c_i) \tag{2.3}$$

- N is the number of data points.
- σ_i^2 is a robust measure of dispersion of each cluster c_i, which helps to determined the boundaries of the cluster, whose radius can be written as $K\sigma_i^2$. K is close to $\chi^2_{n,0.995}$ for spherical Gaussian clusters. The dispersion measure is updated through an iterative *hill-climbing* procedure using the past values of σ_i^2 in each iteration in the following way:

$$\sigma_i^2 = \frac{\sum_{j=1}^{N} w_{ij} d_{ij}^2}{\sum_{j=1}^{N} w_{ij}} \tag{2.4}$$

Extraction of Final Prototypes

In this component, better candidates (better individuals from each niche) are selected. For this purpose, clusters are sorted in descending order of their fitness values. First,

candidates under a minimum fitness value are discarded, this is the *fitness extraction*. Then, the *niche extraction* is performed. This consists in select only the best cluster of each niche, removing the others that have less fitness and that belong to the same niche. Niche extraction is determined using the following:

$$\begin{array}{c} \textit{If } \left(dist\left(P_i, P_k\right) < Kmin\left(\sigma_i^2, \sigma_k^2\right)\right) \textit{ Then} \\ P_i \text{ and } P_k \textit{ are from the same niche} \end{array} \tag{2.5}$$

Refinement of the Extracted Prototypes

This component is an optional phase into the process. Its goal is to improve the center and the size of final clusters. This phase consists in make a partition of the dataset into c clusters in which each feature vector is assigned to the closest to the prototype. A local search is performed with the partitions using a robust estimator called Maximal Density Estimator (MDE), to estimate the center and scale parameters.

2.5 Artificial Immune Theory

Artificial Immune Systems (AIS) is a bio-inspired computing area that is based on immune principles. This theory attempts to explain the memory and learning capabilities of the immune system. The main hypothesis states that interactions between immune cells are responsible for the behavior of the immune system and in consequence, responsible for the immune memory [38, 37]. In a general way, an AIS is composed by:

- antigens (Ag): molecules that immune cells are able to recognize and that act like training elements.
- antibodies or B-cells (Ab): elements generated by the model which recognized Ag. The B-cells interactions (excitatory or inhibitory) are responsible for maintaining the immune memory.

Immune Response

The process followed by the AIS begins when an Ab recognizes an Ag. This recognition is made by binding a portion of the Ag called *epitope*, with a portion of the Ab region called *paratope*. To measure how well an antibody recognizes an antigen a criteria called *affinity* is used, which produce the activation of the immune system when this value is greater than a threshold (*affinity threshold*). Once the immune system is activated an *immune response* is triggered. The basic features of the *immune response* are:

- *Clonal selection principle*: this principle states that those Ab responsible of the immune response, must proliferate and begin to be part of the *memory cells*. *Clonal selection* does not use cross over due to it is an asexual reproduction.
- *Affinity maturation of the immune response*: this process leads to the evolution of Ab populations to better adapt in order to recognize specific Ag. This is made by replicating and mutating the *memory cells* in a process called *clonal expansion*, which is composed by:

  - *clonal proliferation*: this is the cloning process of the *memory cells*. Cloning is made with a rate that is directly proportional to the affinity with the *Ag*.
  - *hypermutation*: once *memory cells* are cloned, they are mutated at high rates. Mutation rates are inversely proportional to their affinity.

Immune Memory

Learning and memory properties include raising the population size and affinity of those *Ab*'s that are consider valuable when recognized some *Ag*'s. These properties emerge from *clonal selection principle*. The *maturation of the immune response* is composed by two phases:

- *Primary immune response*: this is the process of generate new *Ab*'s when an *Ag* is presented for the first time.
- *Secondary immune response*: this process is generated when an *Ag* has been presented before to the *Ab*'s. In this case, *Ab*'s that recognized *Ag*'s are produced quickly and in a larger amounts. The effectiveness of this response is explained by the existence of the *immune memory*.

  Those *Ab*'s presented in the *secondary immune response* (memory response) have, on average, higher affinities than those presented in the *primary immune response*. In the process can happen that:

  - *clonal selection* process allows that occasionally an increases of the fitness of the *Ab*'s occurs. *Ab*'s with high affinity are selected to be part of the pool of *memory cells*,

    or,
  - *memory cells* may be gradually disabled. This is, those *Ab*'s with low affinity may be suffer mutation again and may be eliminated if there is not improvement on its affinity. Thus, something learned by the network can be forgotten unless it is reinforced by other part of it. The programmed cell death of *Ab*'s that are non-stimulated is called *apoptosis* [21, 39]. It is important to note that only those cells which are sufficiently closely related to active parts of the network will survive.

Artificial Immune Network

Artificial Immune Networks (AIN) are based on the Artificial Immune Network Theory, proposed initially by Jerne [93]. This theory suggests that the immune system is composed of a regulated network of cells *Ab*'s, that recognizes one another even in the absence of *Ag*'s and that the system achieves *immunological memory* through the formation of a network of B-cells in a process that could be stimulatory or suppressive: (In figure 2.5 is shown the idiotypic network hypothesis (Figure taken from [117]).

- *Stimulation*: when the *paratope* of an *Ab* recognize *epitopes* or *idiotopes* (a set of *epitopes*), the *Ab* is stimulated and its concentration increases (*proliferation*).

- *Suppression*: when the *idiotope* of an *Ab* is recognized by the *paratopes* of other *Ab*'s, the first is suppressed and its concentration is reduced.

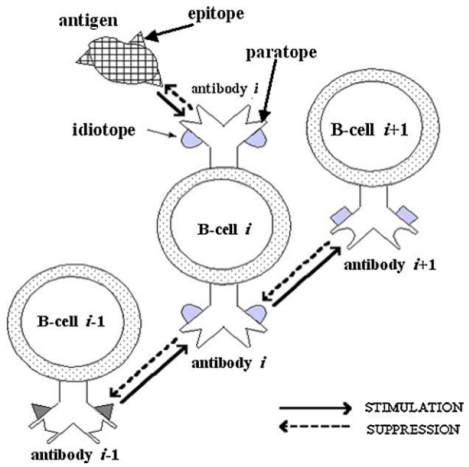


FIGURE 2.5. Idiotypic Network Hypothesis

In general, an AIN has three characteristics: *structure*, *dynamics* and *metadynamics*.

- *Network structure*: describes the possible interactions among its components and how they are link together.
- *Dynamics*: decribes the interactions among immune components; and represents the changes of the concentration, the affinity variation of the *Ab*'s, and their adaptation to the environment.
- *Metadynamics* addresses the process of *Ab*'s production, the death of unstimulated *Ab*'s and the changes of the network connections. The *metadynamic* represents the immune learning mechanism [39].

In [62], a comparison and classification of the AIN models is presented. Authors proposed three branches in which models are presented. Another brief review of AIN models and applications can be found in [147].

2.6 Summary

Clustering data stream mining techniques can be classified into *data-based* and *task-based*. In the first one, *Sampling*, *Load Shedding*, *Sketching*, *Synopsis Data Structure* and *Aggregation* methods can be found. The second technique is composed by *Sliding Window*, *Approximation Algorithms* and *Algorithm Output Granularity* methods. Each technique is explained and shown in a diagram. Each method is described and some works developed in each of them are presented. Dependence among works is shown.

The three phases that composes the *ECSAGO* algorithm: *evolutionary process*, *extraction of the final prototypes* and *refinement of extracted prototypes* are described. Concepts and main processes about *Artificial Immune Theory* are introduced and explained together with the way in which an *Artificial Immune Network* is formed and the immune memory maintained.

CHAPTER 3

ESCALIER Algorithm

In this chapter is presented a new model to mining data streams. The new algorithm called *ESCALIER: Evolutionary Stream Clustering Algorithm with seLf adaptive genetic operators and Immune mEmoRy* is based on the *ECSAGO*. *ESCALIER* takes advantage of the evolutionary process proposed by *ECSAGO* and adds some new features. Such as the use of the sliding window technique and a memory mechanism based on the artificial immune network theory. The first is used to deal with high amount of data. The second aims to maintain and forget clusters that have been discovered.

ESCALIER is divided in two stages: *evolutionary process* and *extraction of the prototypes*, which are inherited from *ECSAGO*. However, some processes within these phases have also been modified to adjust the model to deal with data streams.

This chapter is divided in four sections. Section 3.1 presents the proposed evolutionary stream clustering algorithm with self adaptive genetic operators and immune memory, *ESCALIER*. Section 3.2 reports some experimental results on synthetic dataset. Section 3.3 reports some results on a real dataset. Section 3.4 presents a summary of this chapter.

3.1 Proposed ESCALIER Model and Algorithm

In order to extend the *ECSAGO* for mining large volumes of dynamic data streams, some modifications over the original model have been developed in terms of Artificial Immune Network Theory. The proposed model *ESCALIER* is divided in two stages: *evolutionary process* and *extraction of the prototypes*, which are inherited from *ECSAGO*. However, some processes within these phases have also been modified to adjust the model to deal with data streams.

In the first stage is shows how the model performs the selection of the initial population. Population P_i to be evolve is composed by three subsets: wp that contains the Ab's selected randomly from the window w_i; the prototype $prot$ that contains the discovered prototypes in the window $w_i - 1$ and the clones of the prototypes $ClonProt$ that are generated by the *Clonal Selection Principle*. These two last subsets are part of the *Pool of Memory Cells*. Each individual $ind \in P_i$ evolves independently as proposed by *ECSAGO*. Fitness evaluation is performed making a distinction between Ab's that belong to w_i and that

represent a single data; and *Ab*'s that belong to *prot* or *ClonProt* which have information of the past data.

In the second stage, *suppression* is performed. This is made in two steps of elimination: first, *Ab*'s under an affinity threshold and second *Ab*'s that belong to the same niche (*Ab*'s that recognize another *Ab*'s). *ESCALIER* model is presented in the Figure 3.1 and its detail explanation is given in the following subsections.

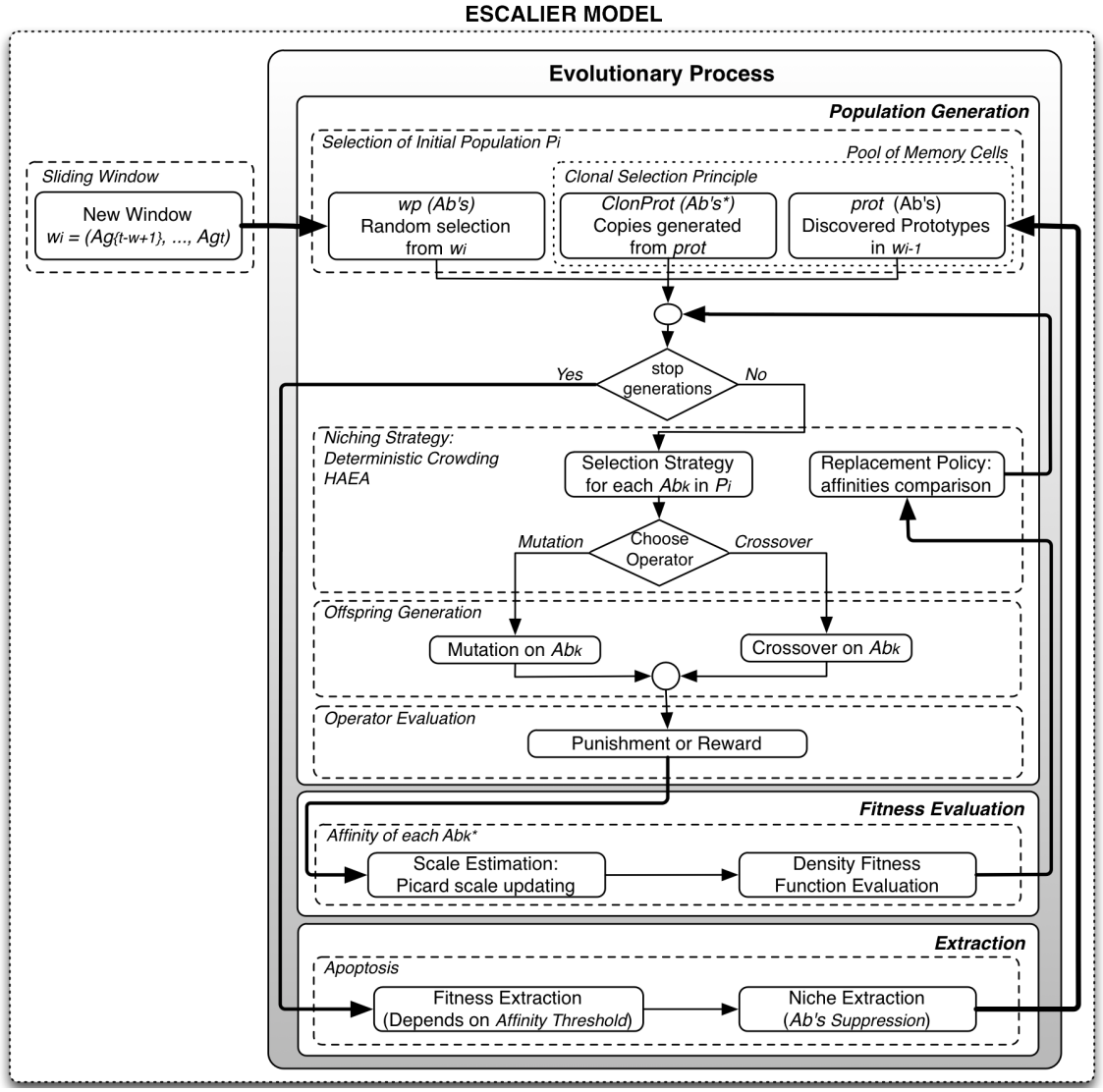


FIGURE 3.1. ESCALIER Model

3.1.1 Sliding Window

The high amount of data produced continuously at high rates requires the inclusion of a technique to handle them. Sliding window technique is the mechanism to divide the data in order to be processed. In this technique, a window w_i (line 1 in the algorithm 3) is a chunk of data that is continuous and keeps the arriving order of the data. A window w in a time i is defined as: $w_i = (X_{t-w+1}, ..., X_t)$, where X_i is the *i-es*-record (Ag) in the

window and w is the number of total records in the window i, that means that X_1 is the oldest record in the window. These windows are count-based which means that all of the windows that arrive to the algorithm, have the same size and the size is determined by the kind of the application (satellite systems, sensor systems, network monitoring applications, etc).

3.1.2 Population Generation using Clonal Selection

The initial population of each window P_i (line 2 in the algorithm 3) is performed in the following way: for each window w_i, population generated is composed by the union of three sets:

- *wp* is a proportion of the *Ag*'s that is taken randomly from the w_i.
- *prot* is composed by the discovered prototypes *Ab*'s in the window $w_i - 1$
- *ClonPop* is formed by non-identical copies generated from the *prot* set. These copies are product of the *clonal selection principle* but unlike this process, *ESCALIER* generates copies using one of the genetic operators: Gaussian Mutation or Linear CrossOver. The selection of the operator is made according to the process explained in 3.1.4.

Notice that the *prot* set is smaller in proportion to the *wp* set; and the number of generated copies of each cluster is proportionally to its density. This means the higher the amount of points the cluster has recognized through its existence the higher number of copies of it.

Explicitly from AIN point of view, *Ab*'s are clusters that have been recognized and *Ag*'s are the data arriving in each window. In each new window, clusters coming from previous windows are represented by those *Ab*'s that have been activated. They must proliferate proportionally with their affinity value. Despite that proliferation of clones is given by performing hypermutation, this model proposes to perform it by using the same operators applied in the population evolution. Thus, proliferation is made by selecting Gaussian mutation or Linear Crossover, depending on the rates learned from them. Additionally, as the generated copies correspond to a proportion of the population, the algorithm ensures that the amount of population is controlled, and avoids the explosion population. In the figure 3.2 the process of population generation is shown.

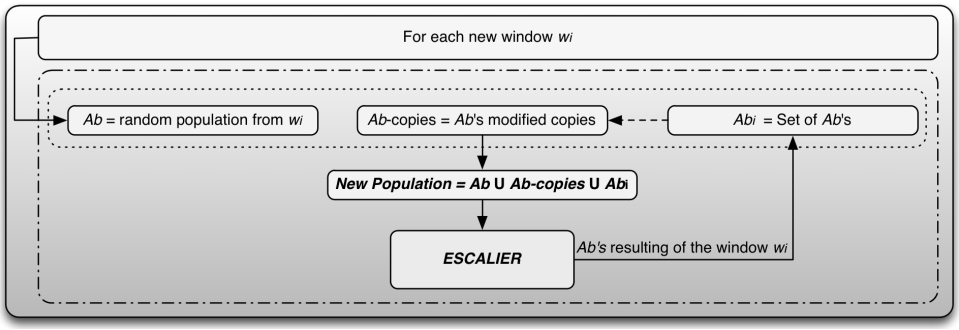


FIGURE 3.2. ESCALIER - Population Generation

Population for a window w_i is generated in the following way:

$$P_i = wp \cup prot \cup ClonPop \tag{3.1}$$

where:

- wp is the population generated randomly from w_i.
- $prot$ is the set composed by prototypes that have been discovered in past windows.
- $ClonPop$ is the population generated from the $prot$ set, which is defined as:

$$ClonPop = clonPop_1 \cup ... \cup clonPop_i \tag{3.2}$$

where each $clonPop_n$ with $1 > n >= i$ has the *copies* generated of the respective $prot_k \in prot$. The number of *copies* of each prototype is calculated as:

Be $\{NPointsProt_1, ..., NPointsProt_i\}$ the number of recognized points by each $prot_k$ when the window w_i arrives, and be NP the total population to be created. Then the number of copies by prototype $NClonesProt$ is given by:

$$\begin{aligned} NClonesProt_1 &= \frac{NPointsProt_1 * NP}{NPointsProt_1 + ... + NPointsProt_i} \\ &\vdots \\ NClonesProt_i &= \frac{NProt_n * NP}{NPointsProt_1 + ... + NPointsProti} \end{aligned} \tag{3.3}$$

Notice that for the first window w_1, $ClonPop = \{\emptyset\}$

3.1.3 Population Summarization and Fitness Evaluation - Affinity

Each individual in the population P_i is composed by a weight W_j and a radius σ_j. Such values have the information about the points that they represent. Individuals that belong to wp are initialized with the values 0 and 1 respectively; while prototypes from $prot$ have been learning those values through their evolution. Used fitness function (lines 3 and 18 in the algorithm 3) is shown in the equation 3.4, and explained in detail in subsection 2.4.

$$f_i = \frac{\sum_{j=1}^{N} w_{ij}}{\sigma_i^2} \tag{3.4}$$

The above implies that interactions with prototypes are different from the interactions with individuals that not represent a subset of points, they are individuals that do not have summarized data. For this reason, the weight and the radius calculations for prototypes should be different. These values are based on the *Scalable ECSAGO* algorithm [110]. According to this, the weight W_{ij} takes into account the proportion that an individual i intersects to another j and it is calculated as:

$$W_{ij} = \begin{cases} W_j * \dfrac{\int \exp\left(-\frac{dx^2}{2\sigma_j^2}\right) dx}{\int \exp\left(-\frac{x^2}{2\sigma_j^2}\right) dx} & \text{if } j \in prot \text{ or } j \in ClonProt \\ 1 & \text{if } j \in wp \text{ and } W_{ij} > T_w \\ 0 & otherwise \end{cases} \tag{3.5}$$

where:

- i represents the $i-th$ individual.
- j represents the $j-th$ prototype.
- W_j is the cumulative weight of the $j-th$ prototype.
- T_w is a weight threshold used for binarization.

The radius σ_{new}^2 of the individual i is calculated as:

$$\sigma_{new}^2 = \frac{\sum_{j=1}^{Nnew} W_{ij} d_{ij}^2}{\sum_{j=1}^{Nnew} W_{ij}} \tag{3.6}$$

where:

- $Nnew$ is the size of the window w_i + number of summarized points.
- $$d_{ij}^2 = \begin{cases} \frac{2*r_i - |d_{ij} - r_i| + r_j}{2} & \text{if } j \in prot \\ \sigma_j^2 & \text{if } j \in wp \text{ and } d_{ij} \neq 0 \\ d_{ij}^2 & otherwise \end{cases} \tag{3.7}$$

Generated copies inherit a portion of the parent fitness. First, a percentage of area that child and parent has in common is calculated. Then, that percentage of fitness and recognized points from the parent are added to the child (line 4 in the algorithm 3). The *affinity* is given by the fitness measure. Thus, when a point is cover by an specific cluster means that an Ag has been presented and recognized by an Ab, and this reinforces the memory of that Ab.

3.1.4 Selection and Replacement Mechanism

Replacement mechanism used in *ECSAGO* is followed by *ESCALIER*. For this, DC and HAEA are used to select the best individual ind_j in each generation (lines 6 to 29 in the algorithm 3). The operators used by each individual are Gaussian mutation (GM) and linear crossover (LC) (line 9 in the algorithm 3). GM is performed by randomly selecting a gene from the parent and adding to it a random number which follows the Gaussian distribution. Notice that *hypermutation* is not performed due to HAEA has a mechanism

to learn the operator rates. LC is applied over two parents. In the process, a value between 0 and 1 is selected. Children are generated as follows: the first one is the result of multiplying the first parent by the *selected value*, the second parent is multiplying by 1− *selected value*, and then the two parents are added. The second child is generated in the same way, but changing the multiplication value. This is, the first parent is multiply by 1− *selected value* and the second one by the *selected value*.

3.1.5 Prototypes Extraction - Suppression

At the end of each generation, two things are made: first, the fitness of the prototypes that have survived for more than a window is updated by taking into account the new data on the window. This update is made by calculating the fitness of the prototype with the new data and by adding it to the fitness that they had (line 31 in the algorithm 3). Second, fitness and niche extraction (equation 2.5) are performed (lines 32 to 36 in the algorithm 3). Existing prototypes after extraction process are include into the population, *prot*, to be evolve in the next window $w_i + 1$.

This process means that activated *Ab*'s are included as part of the *pool of memory cells*. *Ab*'s that were not activated are eliminated, as part of the *apoptosis* process. Notice that *ESCALIER* ensures that at the end of each generation only the best prototypes per niche are maintained, and for this reason, suppression is enough to avoid the population explosion, which makes not necessary the application of *apoptosis* (programmed cell death).

3.1.6 ESCALIER Algorithm

Since *ESCALIER* is based on *ECSAGO* algorithm, and modifications over this algorithm are not in its evolutionary process, *ESCALIER* inherits its complexity. *ECSAGO*[109] is linear with respect of size of the dataset, population size and number of generations. Due to *ESCALIER* does not need all the dataset, the algorithm complexity only depends on the windows size. And since the size of the population and number of generations compared with the size of the windows are very small, complexity in time remains linear.

Something similar occurs with the space complexity. Every window that arrives to *ESCALIER* is evolved, and data stored in memory are replaced by the arriving data. So, linear complexity in space in maintained through all the process. *ESCALIER* algorithm is presented in algorithm 3.

Algorithm 3 ESCALIER Algorithm

Repeat for each window $(w_i, prot)$
  Generate Population $P_i = wp \cup prot \cup ClonPop$
  Calculate Fitness f_j for each $ind_j \in P_i$
  Calculate Inheritance Fitness hf_j for each $ClonPop_j \in P_i$
  while *Generations*
    for $j = 1$ to $j = \text{size}(p_i)$ **do**
      $\text{rates}_j = \text{extract\_rates}\ (ind_j)$
      $\delta = \text{random}(0, 1)$ ▷ Learning rate
      oper = op_select ((GM, LC), rates_j)
      parents = extra_sel(arity(oper)−1, $p_i, ind_j) \cup \{ind\}$
      offspring = apply (oper, parents) $\cup \{ind_j\}$
      $\text{x} = \text{ind}_j$
      $\text{min} = 1e + 8$
      for $k = 1$ **to** size(offspring) **do**
        if $\text{d}(\text{ind}_j; \text{offspring}_k) > 0$ and $\text{d}(\text{ind}_j; \text{offspring}_k) < \text{min}$ **then**
          $\text{x} = \text{offspring}_k$
          $\text{min} = \text{d}(\text{ind}_j; \text{offspring}_k)$
        Calculate $f(\text{x})$
        if $f(ind_j) > f(\text{x})$ **then**
          $\text{x} = \text{ind}_j$
          $\text{rates}_j[\text{oper}] = (1.0 - \delta) * \text{rates}_j[\text{oper}]$ ▷ Punish
        else
          $\text{rates}_j[\text{oper}] = (1.0 + \delta) * \text{rates}_j[\text{oper}]$ ▷ Reward
        child = x ▷ Best child
        normalize_rates(rates_j)
        set_rates(child, rates_j)
        $P_i = P_i \cup \{child\}$
      end for
    end for
  end while
  Update *prot* Fitness $f_{i-1} + f_i$
  Sort each $ind \in P_i$ by its fitness value
  for $k = 1$ **to** size(p_i) **do**
    if $f(ind_k) > f_{ext})$ and niche of ind_k is $\neq$ from niches $\in$ *prot* **then**
      prot $\leftarrow$ prot $\cup$ ind_k
  end for

3.2 Experiments Using Synthetic Data

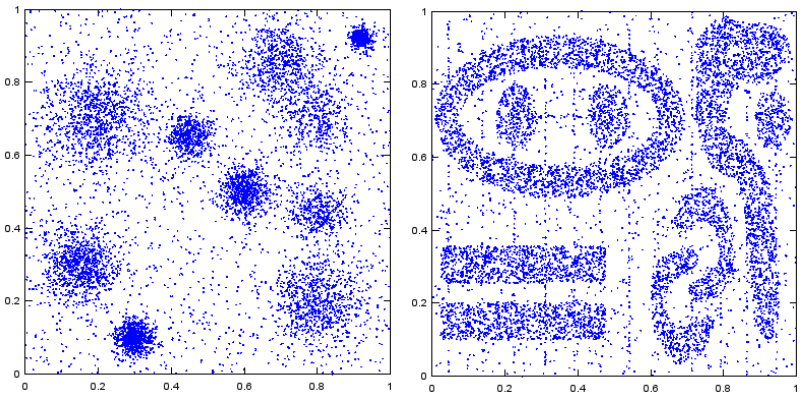
In order to determine the performance of ESCALIER, the algorithm has been tested using three synthetic datasets: *1)* a 2-dimensional Gaussian dataset; *2)* the $t7.10k$ Chameleon dataset; and *3)* a synthetical dataset generated by MOA (Massive Online Analysis)[1]. In the two first datasets, two stages are proposed: *a)* clusters in datasets are in order, and *b)* dataset has been shuffled randomly. Experimental setup, results and analysis for these

[1]Available: http://moa.cms.waikato.ac.nz/. April, 2013

datasets are shown in sections 3.2.1 and 3.2.2. The third dataset is compared with other stream clustering algorithms. Results and analysis are shown in section 3.2.3.

3.2.1 Experimental Setup

The first dataset 2-dimensional Gaussian consists of 8000 points and 10 clusters. Chameleon dataset is composed by 10000 points. The figure 3.3 shows the two datasets. For each of the two stages of the two datasets, three experiments have been carried out, each of them with three different window sizes. For all the experiments the percentage of population generated from a new window is 60%, and 40% is the percentage of copies generated from the clusters. Parameters used in the experimentation are shown in table 3.1.



a) 2-dimensional Gaussian b) t7.10k Chameleon

FIGURE 3.3. 2-dimensional Gaussian and t7.10k Chameleon Datasets

	2-dimensional Gaussian set		*t7.10k Chameleon*	
	in order	*disorganized*	*in order*	*disorganized*
Population Size	80	80	20	20
Number of Generations	30	30	10	10
Window Size Experiment 1	200	200	400	400
Window Size Experiment 2	400	400	600	600
Window Size Experiment 3	600	600	800	800

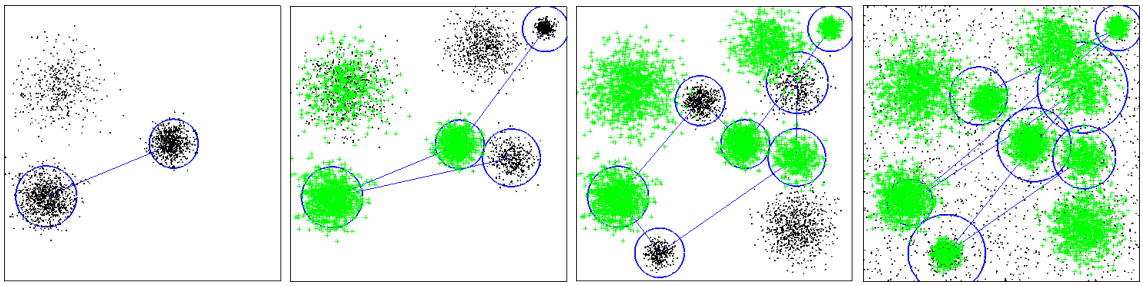
TABLE 3.1. Parameters Setup for Synthetic Datasets

3.2.2 Results and Analysis

Figures presented below belong to the four experiments performed over the two datasets. Each figure shows the experiments carried out with different windows size, and how the environment is in four specific windows. For all the figures, the black points are data that belong to the current window and the green points are data belonging to past windows.

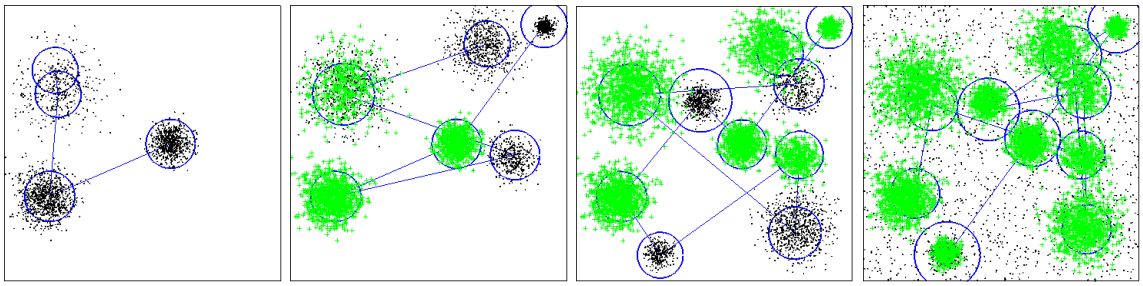
3.2.2.1 Stage 1: 2-Dimensional Gaussian Set (In Order)

Figures 3.4, 3.5 and 3.6 show the results obtained using a window size of 200, 400 and 600, respectively. Each subfigure in the figures show the obtained results in the windows $10, 20, 30$ and 40.



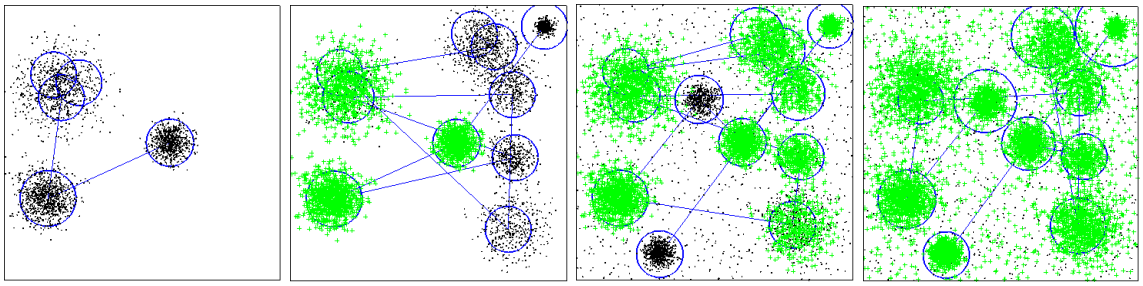
a) window 10 b) window 20 c) window 30 d) window 40

FIGURE 3.4. Detected clusters on different windows for the 10-Gaussian Cluster Dataset. Window Size = 200



a) window 5 b) window 10 c) window 15 d) window 20

FIGURE 3.5. Detected clusters on different windows for the 10-Gaussian Cluster Dataset. Window Size = 400



a) window 4 b) window 8 c) window 12 d) window 14

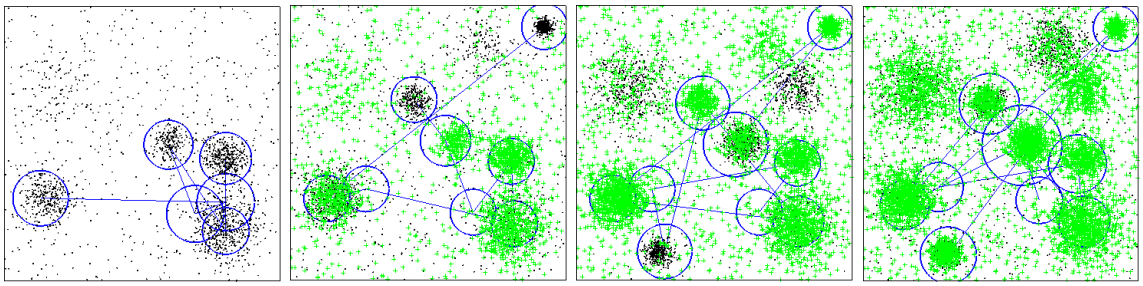
FIGURE 3.6. Detected clusters on different windows for the 10-Gaussian Cluster Dataset. Window Size = 600

Figures presented above show how *ESCALIER* recognizes clusters in those zones with high density of points. For windows size equal to 400 and 600, all 10 clusters in the dataset are recognized. For the window size equal to 200 only 7 out 10 clusters have been recognized. This is because the points presented in small windows are not enough dense

to be considered as a cluster. The algorithm considers those points as outliers. Subfigures in each figure show how through time recognized clusters are maintain in the following windows and how new clusters that represent the points in the new window are detected.

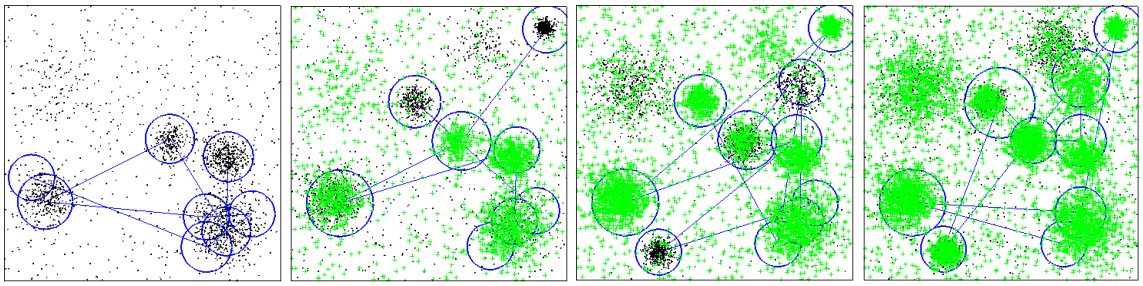
3.2.2.2 Stage 2: 2-Dimensional Gaussian Set (Disorganized)

Figures 3.7, 3.8 and 3.9 show the results obtained using a window size of 200, 400 and 600, respectively. Each subfigure in the figures show the obtained results in the windows $10, 20, 30$ and 40.



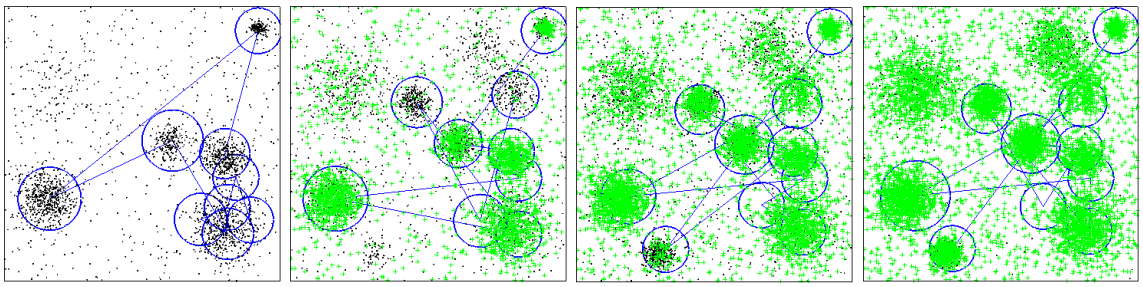
a) window 10 b) window 20 c) window 30 d) window 40

FIGURE 3.7. Detected clusters on different windows for the 10-Gaussian Cluster Dataset Disorganized. Window Size = 200



a) window 5 b) window 10 c) window 15 d) window 20

FIGURE 3.8. Detected clusters on different windows for the 10-Gaussian Cluster Dataset Disorganized. Window Size = 400



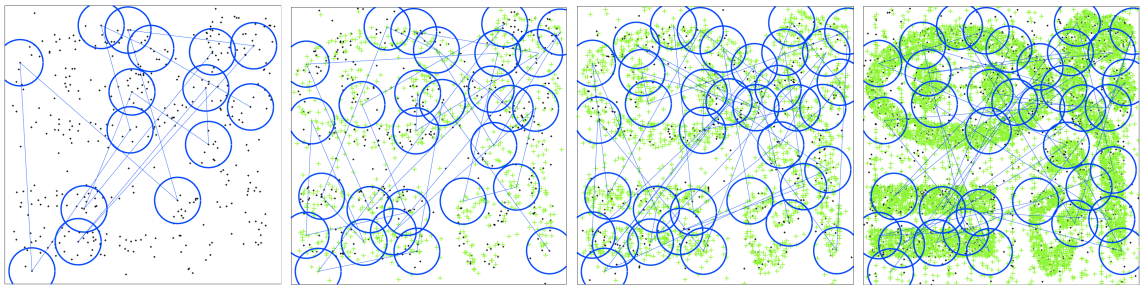
a) window 4 b) window 8 c) window 12 d) window 14

FIGURE 3.9. Detected clusters on different windows for the 10-Gaussian Cluster Dataset Disorganized. Window Size = 600

When clusters are presented in a disorganized way, for windows size equal to 200 and 400 is recognized 9 out 10 clusters. For the window size equal to 600, 10 clusters have been recognized. However, 2 of them are not part of the real clusters in the dataset. Figures show that there are two zones with less density of points, that are recognized as outliers.

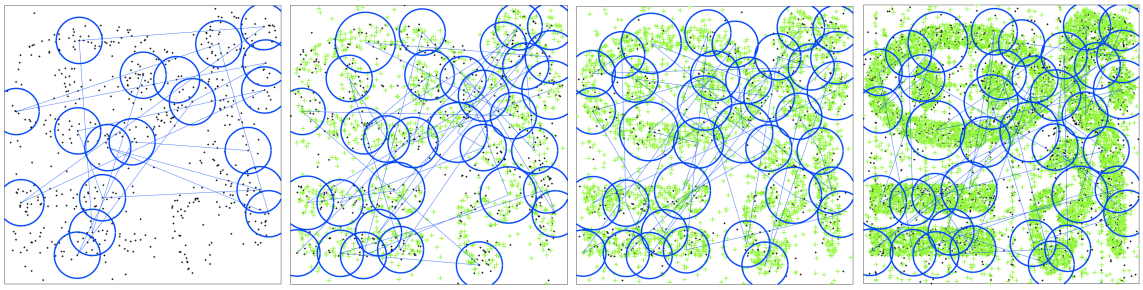
3.2.2.3 Stage 3: t7.10k (In Order)

Figures 3.10 , 3.11 and 3.12 show the results obtained using a window size of 400, 600 and 800, respectively. Each subfigure in the figures show the obtained results in the windows $1, 3, 6$ and the last window in each window size.



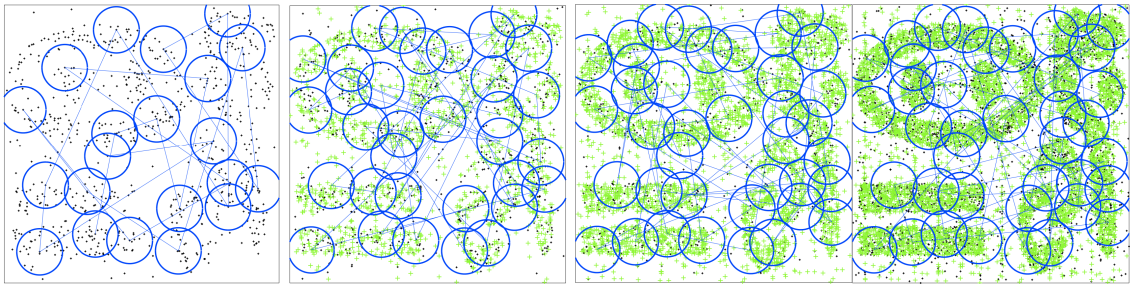
a) window 5 b) window 15 c) window 20 d) window 25

FIGURE 3.10. Detected clusters on different windows for the t7.10k Dataset. Window Size = 400



a) window 4 b) window 8 c) window 12 d) window 17

FIGURE 3.11. Detected clusters on different windows for the t7.10k Dataset. Window Size = 600



a) window 1 b) window 3 c) window 6 d) window 13

FIGURE 3.12. Detected clusters on different windows for the t7.10k Dataset. Window Size = 800

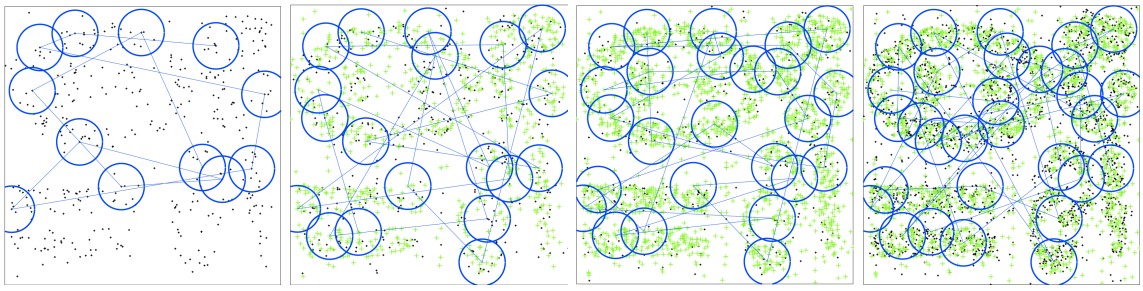
	Windows Size		
	400	600	800
Window 1	15	19	35
Window 3	28	26	35
Window 6	36	38	35
Window (25 - 17 - 13)	37	38	38

TABLE 3.2. t7.10k (In Order) - Recognized Clusters

Figures above show that the algorithm is able to recognized all the dense areas and maintain clusters through time. Notice that clusters represent the dataset and only 10 generations for each window has been used. The amount of recognized clusters per windows are shown in table 3.2

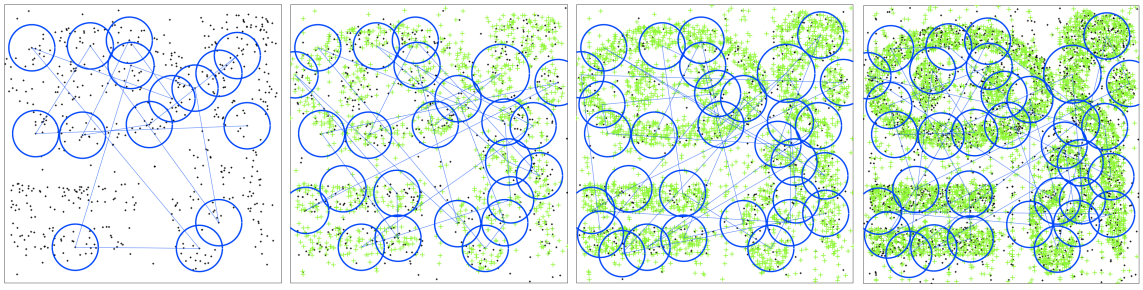
3.2.2.4 Stage 4: t7.10k (Disorganized)

Figure 3.13 shows results obtained with a size window = 400, figure 3.14 with a size window = 600 and figure 3.15 with a size window = 800. In each of them, subfigures show the results in windows intervals of $1, 3, 6$ and the last window in each window size.



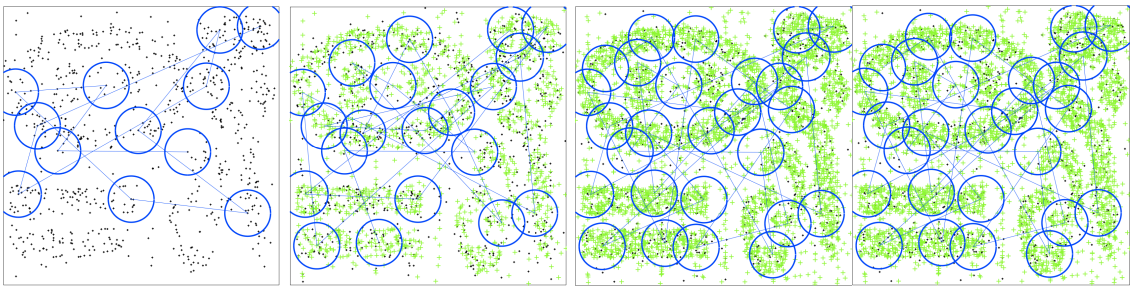
a) window 5 b) window 15 c) window 20 d) window 25

FIGURE 3.13. Detected clusters on different windows for the t7.10k Dataset Disorganized. Window Size = 400



a) window 4 b) window 8 c) window 12 d) window 17

FIGURE 3.14. Detected clusters on different windows for the t7.10k Dataset Disorganized. Window Size = 600



a) window 1 b) window 3 c) window 6 d) window 13

FIGURE 3.15. Detected clusters on different windows for the t7.10k Dataset Disorganized. Window Size = 800

Figures above show that the algorithm is able to recognized all the dense areas and maintain clusters through the time even when clusters in the dataset are disorganized. Although the amount of recognized clusters when dataset is disorganized is less than the recognized amount when dataset is organized, recognized clusters in the disorganized case, have less intersections between them and in general, represents the dataset. The amount of recognized clusters per windows are shown in table 3.3

	Windows Size		
	400	600	800
Window 1	12	15	12
Window 3	19	24	20
Window 6	25	26	26
Window (25 - 17 - 13)	26	26	26

TABLE 3.3. t7.10k (Disorganized) - Recognized Clusters

Discussion

Experiments show that *ESCALIER* is able to recognize clusters in dense areas and to maintain those clusters through time. Notice that there are clusters that are not forgotten and they are carried from window to window. This is because once clusters are recognized, the following windows contain points they recognize. Those points make that clusters are kept active. Likewise, there are clusters that are forgotten after some windows. These are the clusters that have not been activated for a while.

In the experiments in which the smaller windows size has been used, not all the clusters have been recognized. This is because the points presented in small windows are not enough dense to be considered as a cluster. The algorithm considers those points as outliers. In the case of shuffled data is possible to observe that not all the clusters are recognized and some of the recognized clusters are not part of the real clusters. These situations are presented because: *i)* the points are distributed in small portions among the windows, in this case points are taken as outliers. *ii)* nearby points which belongs to different classes are contained in a window and the algorithm recognized them as a cluster.

3.2.3 Synthetic MOA Dataset

In order to compare the *ESCALIER* performance with another algorithms, a synthetic dataset has been generated using the RandomRBG Generator function[2] provided by MOA. MOA is a real time analytics tool for data streams. Algorithms to perform clustering data streams included in the tool are: *CluStream*, *DenStream*, *ClusterGenerator*, *CobWeb*, *WekaClustering*, *ClusTree* and *StreamKM*. The generated dataset has 5 attributes, 100000 instanes, and 3 classes. For all the algorithms a windows size equal to 1000 has been used.

The performance of *ESCALIER* is compared with those algorithms implemented in MOA that reported results. The clusters quality are shown in the table 3.4. This table present the purity of the clusters in the windows 1, 25, 50, 75 and 100, for all the algorithms. Purity is calculated using the equation 3.8 [154].

$$purity\ w_i = \frac{\sum_{i=1}^{K} \frac{|\ C_j^d\ |}{|\ C_j\ |}}{K} \tag{3.8}$$

where:

- K is the number total of clusters in the window w_i.
- C_j^d is the number of data points of the dominant class in the cluster C_j.
- C_j is the total number of data points in the cluster C_j.

	Algorithm			
	Cluster Generator	*CluStream*	*ClusTree*	*ESCALIER*
Window 1	0.455	0.881	0.923	0.797
Window 25	0.441	0.879	0.857	0.734
Window 50	0.408	0.768	0.772	0.678
Window 75	0.432	0.621	0.813	0.756
Window 100	0.423	0.729	0.762	0.737

TABLE 3.4. Purity - MOA Synthetical Dataset

Results show that the algorithms, with the exception of *Cluster Generator*, have a similar behavior. However, the number of generated clusters in each window for all the algorithms is 3, except *ESCALIER* that in its first window generated 4 clusters. The average purity of the 5 reported windows for each algorithm is: *Cluster Generator* 0.43; *CluStream* 0.77; *ClusTree* 0.82 and *ESCALIER* 0.74. Notice that *ESCALIER* does not need prior knowledge about the dataset, such as the class to which each data belongs and the total number of clusters.

[2]Available: http://www.cs.waikato.ac.nz/ abifet/MOA/Manual.pdf. Page 25. April 2013

3.3 Experiments Using Real Data

ESCALIER has been tested using the Sensor Stream dataset[3]. This dataset contains information (temperature, humidity, light, and sensor voltage) collected from 54 sensors deployed in Intel Berkeley Research Lab. The stream contains consecutive information recorded over a 2 months period (1 reading per $1-3$ minutes). The dataset is composed by 5 attributes, 2.219.803 instances and 54 classes. The proportion of samples per class is shown in table 3.5.

Class	*Samples*	%
1	43047	1.93
2	46915	2.11
3	46633	2.10
4	43793	1.97
6	35666	1.60
7	55354	2.49
8	15809	0.71
9	45204	2.03
10	47155	2.12
11	41833	1.88
12	19016	0.85
13	27013	1.21
14	26667	1.20
15	2038	0.09
16	32998	1.48
17	33779	1.52
18	33433	1.50
19	39455	1.77
20	28832	1.29

Class	*Samples*	%
21	58521	2.63
22	60164	2.71
23	62409	2.81
24	57352	2.58
25	53162	2.39
26	61513	2.77
27	37627	1.69
29	64384	2.90
30	38343	1.72
31	65689	2.95
32	43092	1.94
33	35749	1.61
34	48764	2.19
35	51338	2.31
36	56357	2.53
37	47915	2.15
38	49155	2.21
39	32736	1.47
40	46621	2.10

Class	*Samples*	%
41	40836	1.83
42	44860	2.02
43	38656	1.74
44	47681	2.14
45	53245	2.39
46	52988	2.38
47	56858	2.56
48	58215	2.62
49	34811	1.56
50	15737	0.70
51	42259	1.90
52	34067	1.53
53	25622	1.15
54	28718	1.29
55	2850	0.12
56	2372	0.10
58	4497	0.20

TABLE 3.5. Class Distribution for Sensor Stream Dataset

3.3.1 Experimental Setup

For this dataset, two experiments have been carried out: First, for a window size of 10000; second, for a window size of 50000. Population size and number of generation were the same for all the experiments. The dataset has been normalized maintaining its original order, this is, as the data generated by the sensors. Parameters used in the experimentation are shown in the table 3.6.

3.3.2 Results and Analysis

Every window in the experiments is composed of sample of all the classes. For the analysis of the results two metrics have been taken into account: the amount of clusters recognized in each window and the purity of those clusters. Since results obtained in both experiments

[3]http://www.cse.fau.edu/ xqzhu/stream.html

	Sensor Data Stream
Population Size	300
Number of Generations	30
Window Size Experiment 1	10000
Window Size Experiment 2	50000

TABLE 3.6. Parameters Setup for Real Dataset

are similar, reported results are from the second experiment, using a window size equal to 50000.

For this experiment, a total of 45 windows has been obtained. Figure 3.16 shows the amount of clusters recognized in each window. The amount of clusters in each window are between 17 and 29, except the last window in which clusters are 10. It is important to notice that the last window has only 19803 data.

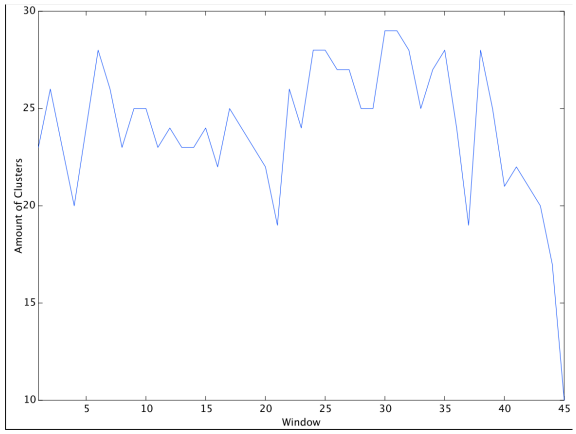


FIGURE 3.16. Real Dataset - Generated Clusters per Window

In more detail, figure 3.17 shows the clusters distribution for each window. In the figure is clear that some classes are never detected as clusters, such as class numbers 12, 13, 14, 15, 50, 52 and 53. Other classes have been detected only in few windows, such as class numbers 6, 8, 33, 39, 54, 55 and 56. As we can see in the table 3.6, those classes have a low percentage of presence in the whole dataset. This could be an explanation of why those classes are not totally recognized.

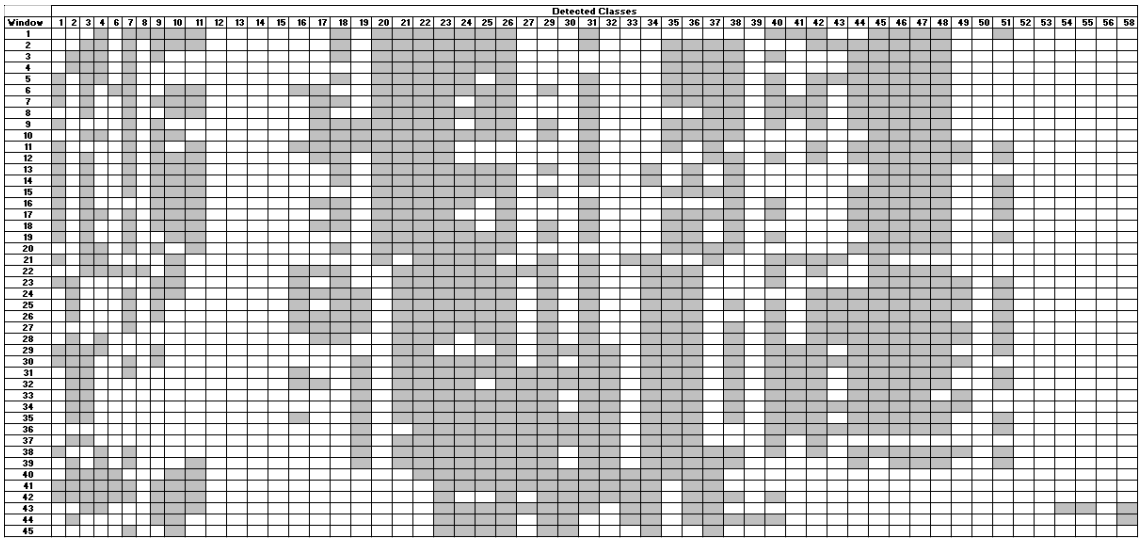
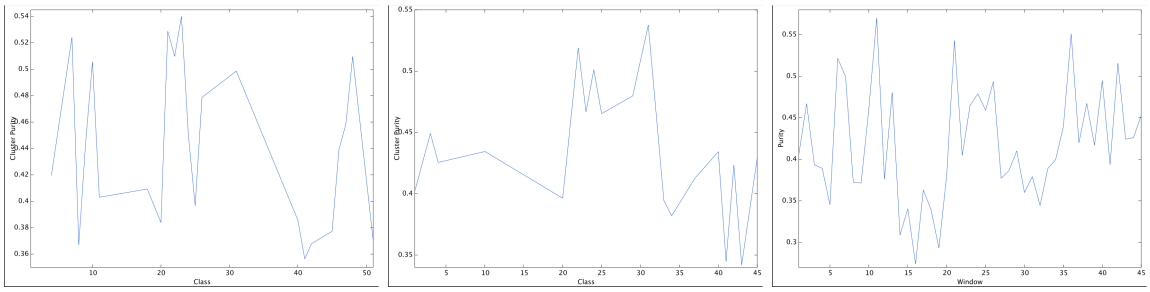


FIGURE 3.17. Real Dataset - Distribution of Detected Clusters per Window

In each window, clusters quality has been evaluated as the average purity of each cluster using the equation 3.8. In figure 3.18, graphics *a)* and *b)*, show the purity of the recognized clusters for the windows 1 and 21. Graphic *c)*, shows the clusters purity of whole 45 windows. In average, clusters purity reach values into the range 0.25 and 0.57.



a) Window 1 b) Window 21 c) All Windows

FIGURE 3.18. Real Dataset - Clusters Purity

3.3.3 Comparison with other Streaming Algorithms

The real dataset was tested using other algorithms from MOA (Massive Online Analysis)[4]. MOA is a real time analytics tool for data streams. Algorithms included in MOA are: *CluStream*, *DenStream*, *ClusterGenerator*, *CobWeb*, *WekaClustering*, *ClusTree* and *StreamKM*.

All algorithms in MOA were tested with the real dataset, but only three of them reported results. Additionally, processing all the data was not possible. At the moment to extract the results, the tool reported a memory issue. For this reason, only 150000 data were processed. Windows size in MOA are adapted automatically. For Sensor dataset, the size of the window was adapted to 1000. In the experiments, default parameters were used.

[4]http://moa.cms.waikato.ac.nz/

Tables 3.7 and 3.8 show the purity and number of clusters recognized using *CluStream*, *DenStream*, *ClusterGenerator* and *ESCALIER*. Results correspond to windows 1 and 3 from ESCALIER, and windows 1 and 150 for the other algorithms (size of the windows in MOA is equal to 1000).

	Algorithm			
	CluStream	*DenStream*	*ClusterGenerator*	*ESCALIER*
Purity	0.82	0.44	0.64	0.41
Number of Clusters	43	42	52	23

TABLE 3.7. Algorithms Comparison - Window 1

	Algorithm			
	CluStream	*DenStream*	*ClusterGenerator*	*ESCALIER*
Purity	0.63	0.28	0.38	0.39
Number of Clusters	42	47	52	23

TABLE 3.8. Algorithms Comparison - Window 3

We can see that despite that purity in clusters recognized by *ESCALIER* is relative low, the algorithm maintains a constant behavior throughout the whole dataset. Algorithms from MOA, suffer sharp falls in their purity. Although *CluStream* has better values for purity, the tool does not provided a way to check how the clusters are formed. Algorithms mentioned above presented good performance with other datasets, but with this in particular, results have not presented good quality because this is a complex dataset.

3.4 Summary

A new model and algorithm for clustering data streams called *ESCALIER - Evolutionary Stream Clustering Algorithm with seLf adaptive genetic operators and Immune mEmoRy* are proposed. This model is an extension of *ECSAGO* algorithm. *ESCALIER* adopts the two first stages proposed by *ECSAGO*: *evolutionary process* and *extraction of the prototypes*. These stages have been modified to adjust the model to deal with data streams. Additionally, *ESCALIER* uses the sliding window technique and introduces a memory mechanism based on the artificial immune network theory.

ESCALIER algorithm has been tested by simulating a data stream environment using three synthetical datasets. Also, a real dataset that contains data taken by a group of sensors has been used for testing. Experiments have been carried out with different size of windows. Results show that *ESCALIER* is able to recognized and maintained clusters through time and forgetting those clusters that have not been activated for a while. The behavior of *ESCALIER* is similar to the other tested algorithms, with the advantage that not need previous knowledge of the dataset.

CHAPTER 4

Conclusions and Future Work

In this work, a new evolutionary stream clustering algorithm with self adaptive genetic operators and immune memory called *ESCALIER* was presented. This algorithm is an extension of the *ECSAGO* algorithm from which two of its three stages are taking as basis: *evolutionary process* and *extraction of the prototypes*. *ESCALIER* modifies the first stage, incorporating new processes for the population generation and the population summarization, taking as a frame the general process proposed by the artificial immune network theory, in order to simulate a memory mechanism. Additionally, sliding window technique is used to handle the data streams.

We highlight the fact that populations size and number of generations needed to evolve each window is low. Additionally, *ESCALIER* does not need a memorization factor to remember recognized clusters, as some other algorithms. Memory is given as a result of the process of the Artificial Immune Network. Also, *ESCALIER* does not need to make off-line phase to performed any process, such as summarization process.

Synthetical and real datasets were used to evaluate the performance of *ESCALIER* algorithm. In the first case, 2-dimensional datasets were used and performance was measured by the amount of recognized clusters. Results suggest that the algorithm is able to detect clusters through time and cover the dense areas. In the second case, a real dataset was used and the amount of clusters per window, and its purity were measured. Results suggest that the algorithm presents a constant behavior through windows, this means, that the algorithm maintained its rate of detecting clusters. Although clusters purity are not high, comparisons made with another algorithms supported by MOA tool, confirmed that the dataset is complex, and that *ESCALIER* recognizes classes with low density as noise.

Obtained results suggest that the algorithm is able to maintain a representation of the past data. This representation is defined as memory of the summarized data, through each window. Data need to be presented only once to the process. We also notice that clusters that are not stimulated with new data are forgotten. They can disappear if arriving windows do not have data of the corresponding class.

The use of the Artificial Immune Network makes that clusters recognized can maintaining their knowledge of the past data. And one of the major issues, the population explosion in the network, it is totally controlled by the phase of extraction, which makes that the network has good performance. The cost of the whole process of recognize and

evolve clusters is low, which makes possible to have good representations of data regardless the size and dimensionality of them. Taking into account the above, together with use of the sliding window technique, makes *ESCALIER* scalable. Since arriving data are processed and discarded, keeping their knowledge through time, and maintaining time and space complexity linear.

As a summary, the contributions of this work are:

- a state of the art on clustering data stream mining techniques which include a structure of some works developed, organized by technique and presented as a tree diagram. A journal with the state of the art will be presented in *IJIPM: International Journal of Information Processing and Management*.
- a new algorithm for the problem of clustering data streams with linear complexity. This algorithm uses ideas from artificial immune network to handle memory issues and extends the *ECSAGO* approach. The algorithm was presented in *CEC - IEEE Congress on Evolutionary Computation* 2013.
- elimination of the memory factor used by *Scalable ECSAGO*.
- the incorporation of the sliding window technique as a mechanism to deal with data streams.

As a future work we propose:

- testing *ESCALIER* with some other real datasets, modifying population size and generation number parameters.
- incorporating a mechanism to merge and divide clusters. This could make that classes with low presence be recognized and maintained through time.
- incorporating a new mechanism to performed summarization, in which fitness is not based on the intersection of the individuals.
- incorporating a mechanism to adapt automatically the size of the windows instead of using a fixed one.

Bibliography

[1] Charu Aggarwal. A framework for clustering massive-domain data streams. In *IEEE International Conference on Data Engineering*, 2009.

[2] Charu C. Aggarwal. An intuitive framework for understanding changes in evolving data streams. In *18th International Conference on Data Engineering*, 2002.

[3] Charu C. Aggarwal. *Data Streams: Models and Algorithms*. Springer, 2007.

[4] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *29th International Conference on Very Large Data Bases*, 2003.

[5] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for projected clustering of high dimensional data streams. In *Thirtieth International Conference on Very Large Data Bases*, 2004.

[6] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. On high dimensional projected clustering of data streams. *Data Mining and Knowledge Discovery*, 10:251 – 273, 2005.

[7] Charu C Aggarwal and Philip S Yu. A framework for clustering massive text and categorical data streams. In *Sixth SIAM International Conference on Data Mining*, 2006.

[8] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD International Conference on Management of Data*, 1998.

[9] Amineh Amini and Teh Ying Wah. Density micro-clustering algorithms on data streams: A review. In *International Conference on Data Mining and Applications (ICDMA)*, 2011.

[10] Amineh Amini and Teh Ying Wah. A comparative study of density-based clustering algorithms on data streams: Micro-clustering approaches. *Intelligent Control and Innovative Computing*, 110:275 – 287, 2012.

[11] Amineh Amini, Teh Ying Wah, Mahmoud Reza Saybani, and Saeed Reza Aghabozorgi Sahaf Yazdi. A study of density-grid based clustering algorithms on data streams. In *Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2011.

[12] Giulio Antoniol, Massimiliano Di Penta, and Markus Neteler. Moving to smaller libraries via clustering and genetic algorithms. In *Proceedings of the Seventh European Conference on Software Maintenance and Reengineering*, 2003.

[13] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. Stream: The stanford data stream management system. Technical report, Stanford InfoLab, 2004.

[14] Mohsen Jafari Asbagh and Hassan Abolhassani. Feature-based data stream clustering. In *International Conference on Computer and Information Science*, 2009.

[15] Brain Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2003.

[16] Sanghamitra Bandyopadhyay, Chris Giannella, Ujjwal Maulik, Hillol Kargupta, Kun Liu, and Souptik Datta. Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, 176:1952 – 1985, 2006.

[17] Daniel Barbara. Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, 3:23 – 27, 2002.

[18] Daniel Barbara and Ping Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the ACM-SIGKDD International Conference on Knowledge and Data Mining*, 2000.

[19] JÃ¼rgen Beringer and Eyke HÃ¼llermeier. Online clustering of parallel data streams. *Data & Knowledge Engineering archive*, 58:180 – 204, 2006.

[20] Christian Bohm, Karin Kailing, Hans-Peter Kriegel, and Peer Kroger. Density connected clustering with local subspace preferences. In *Fourth IEEE International Conference on Data Mining*, 2004.

[21] Barbara Borowik, Bohdan Borowik, Jan Kucwaj, and Sophie Laird. Associative memory in artificial immune systems. *Annales UMCS Informatica*, 10:111 – 122, 2010.

[22] Zhu Can-Shi, Dun Xiao, and Zhu Lin. A study on the application of data stream clustering mining through a sliding and damped window to intrusion detection. In *Fourth International Conference on Information and Computing (ICIC)*, 2011.

[23] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SIAM Conference on Data Mining*, 2006.

[24] Ching-Ming Chao and Guan-Lin Chao. Resource-aware high quality clustering in ubiquitous data streams. In *ICEIS*, 2011.

[25] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, 2003.

[26] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang ha Lee, and Kevin Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *IEEE International Symposium on Workload Characterization*, 2009.

[27] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A bayesian classification system. *Fifth International Conference on Machine Learning*, 27:54 – 64, 1988.

[28] Sheng Chen and Haibo He. Towards incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach. *Evolving Systems*, 2:35 – 50, 2011.

[29] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *13th International Conference on Knowledge Discovery and Data Mining*, 2007.

[30] Ling Chena, Ling-Jun Zoua, and Li Tuc. A clustering algorithm for multiple data streams based on spectral component similarity. *Information Sciences*, 183:35 – 47, 2012.

[31] Zhihong Chong, Weiwei Ni, Lizhen Xu, Zhuoming Xu, Hu Shu, and Jinwang Zheng. Approximate k-median of location streams with redundancy and inconsistency. In *Int. J. Software and Informatics*, 2010.

[32] Seokkyung Chung and Dennis Mcleod. Dynamic pattern mining: An incremental data clustering approach. *Journal on Data Semantics*, 2:85 – 112, 2005.

[33] Fernando Crespo and Richard Weber. A methodology for dynamic data mining based on fuzzy clusterings. *Fuzzy Sets and Systems*, 150:267 – 284, 2005.

[34] Baptiste Csernel, Fabrice Clerot, and Georges Hebrail. Streamsamp datastream clustering over tilted windows through sampling. In *Fourth International Workshop on Knowlegde Discovery from Data Streams*, volume 11, pages 251 – 252, 2007.

[35] Bi-Ru Dai, Jen-Wei Huang, Mi-Yen Yeh, and Ming-Syan Chen. Clustering on demand for multiple data streams. In *Fourth IEEE International Conference on Data Mining*, 2004.

[36] Bi-Ru Dai, Jen-Wei Huang, Mi-Yen Yeh, and Ming-Syan Chen. Adaptive clustering for multiple evolving streams. *IEEE Transactions on Knowledge and Data Engineering*, 18:1166 – 1180, 2006.

[37] Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, 2002.

[38] Leandro Nunes de Castro and Fernando Jose Von Zuben. Artificial immune systems: Part i-basic theory and applications. Technical report, TR - DCA, 1999.

[39] Leandro Nunes de Castro and Fernando Jose Von Zuben. *In Data Mining: A Heuristic Approach*, chapter aiNet: An Artificial Immune Network for Data Analysis, pages 231 – 259. Idea Group Publishing, 2002.

[40] Inderjit Dhillon, Jacob Kogan, and Charles Nicholas. *A Comprehensive Survey of Text Mining*, chapter Feature Selection and Document Clustering, pages 73 – 100. Springer-Verlag, 2003.

[41] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[42] Pedro Domingos and Geoff Hulten. Catching up with the data: Research issues in mining data streams. In *Workshop on Research Issues in Data Mining and Knowledge Discovery DMKD*, 2001.

[43] Pedro Domingos and Geoff Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In *Eighteenth International Conference on Machine Learning*, 2001.

[44] Margaret H. Dunham. *Data Mining Introductory and Advanced Topics*. Prentice Hall, 2003.

[45] Manzoor Elahi, Kun Li, Wasif Nisar, Xinjie Lv, and Hongan Wang. Efficient clustering-based outlier detection algorithm for dynamic data stream. *Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, 5:298 – 304, 2008.

[46] Martin Ester, Hans-Peter Kriegel, Jorg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of 24rd International Conference on very large Data Bases*, 1998.

[47] Martin Ester, Hans peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery*, 1996.

[48] Jianbin Fang, A. L. Varbanescu, and H. Sips. An auto-tuning solution to data streams clustering in opencl. In *IEEE 14th International Conference on Computational Science and Engineering (CSE)*, 2011.

[49] Fredrik Farnstrom, James Lewis, and Charles Elkan. Scalability for clustering algorithms revisited. In *The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[50] Jose R. Fernandez and Eman M. El-Sheikh. Clusandra: A framework and algorithm for data stream cluster analysis. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 2:87 – 99, 2011.

[51] Dimitar Filev and Olga Georgieva. *Evolving Intelligent Systems: Methodology and Applications*, chapter An Extended Version of the Gustafson-Kessel Algorithm for Evolving Data Stream Clusterings, pages 273 – 299. Institute of Electrical and Electronics Engineers, 2010.

[52] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139 – 172, 1987.

[53] Dimitris Fotakis. Incremental algorithms for facility location and k-median. *Theoretical Computer Science*, 361:275 – 313, 2005.

[54] Mohamed Medhat Gaber, Shonali Krishnaswamy, and Arkady Zaslavsky. Adaptive mining techniques for data streams using algorithm output granularity. In *Congress on Evolutionary Computation*, 2003.

[55] Mohamed Medhat Gaber, Shonali Krishnaswamy, and Arkady Zaslavsky. Cost-efficient mining techniques for data streams. In *Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, volume 32, pages 109 – 114, 2004.

[56] Mohamed Medhat Gaber, Shonali Krishnaswamy, and Arkady Zaslavsky. Ubiquitous data stream mining. In *8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2004.

[57] Mohamed Medhat Gaber, Shonali Krishnaswamy, and Arkady Zaslavsky. *Advanced Methods for Knowledge Discovery form Complex Data*, chapter On-board Mining of Data Streams in Sensor Networks, pages 307 – 335. Springer, 2005.

[58] Mohamed Medhat Gaber and Philip S. Yu. A holistic approach for resource-aware adaptive data stream mining. *New Generation Computing*, 25:95 – 115, 2006.

[59] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Resource-aware knowledge discovery in data streams. In *First International Workshop on Knowledge Discovery in Data Streams*, 2004.

[60] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Towards an adaptive approach for mining data streams in resource constrained environments. *Lecture Notes in Computer Science*, 3181:189 – 198, 2004.

[61] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM SIGMOD Record*, 34:18 – 26, 2005.

[62] Juan Carlos Galeano, Angelica Veloza-Suan, and Fabio A. Gonzalez. A comparative analysis of artificial immune network models. In *Conference on Genetic and Evolutionary Computation*, 2005.

[63] Guojun Gan, Chaoqun Ma, and jianhong Wu. *Data Clustering. Theory, Algorithms, and Applications*. ASA-SIAM, 2007.

[64] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. Demon: Mining and monitoring evolving data. In *IEEE Transactions on Knowledge and Data Engineering*, 2001.

[65] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. Mining data streams under block evolution. *ACM SIGKDD Explorations Newsletter*, 3:1 – 10, 2002.

[66] Jing Gao, Jianzhong Li, Zhaogong Zhang, and Pang-Ning Tan. An incremental data stream clustering algorithm based on dense units detection. *Lecture Notes in Computer Science. Advances in Knowledge Discovery and Data Mining*, 3518:420 – 425, 2005.

[67] Lukasz Golab and M. Tamer Ozsu. Issues in data stream management. *ACM SIGMOD Record*, 32:5 – 14, 2003.

[68] Jonatan Gomez. Self adaptation of operator rates for multimodal optimization. *Congress on Evolutionary Computation*, 2:1720 – 1726, 2004.

[69] Jonatan Gomez. Self adaptation of operator rates in evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2004.

[70] Jonatan Gomez, Dipankar Dasguptaw, and Olfa Nasraoui. A new gravitational clustering algorithm. In *In Proceedings of the Third SIAM International Conference on Data Mining*, 2003.

[71] Jonatan Gomez, Elizabeth Leon, and Olfa Nasraoui. Rain: Data clustering using randomized interactions of data points. In *In Proceedings of the Third International Conference on Machine Learning and Applications*, 2004.

[72] Jonatan Gomez, Juan Pena-Kaltekis, Nestor Romero-Leon, and Elizabeth Leon. Incrain: An incremental approach for the gravitational clustering. In *6th Mexican International Conference on Advances in Artificial Intelligence*, 2007.

[73] Linghui Gong, Jianping Zeng, and Shiyong Zhang. Text stream clustering algorithm based on adaptive feature selection. *Expert Systems with Applications*, 38:1393 – 1399, 2011.

[74] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15:515 – 528, 2003.

[75] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *IEEE Annual Symposium on Foundations of Computer Science*, 2000.

[76] Michael Hahsler and Margaret Dunham. Temporal structure learning for clustering massive data streams in real-time. In *SIAM International Conference on Data Mining*, 2011.

[77] Maria Halkidi and Iordanis Koutsopoulos. Online clustering of distributed streaming data using belief propagation techniques. In *12th IEEE International Conference on Mobile Data Management (MDM)*, 2011.

[78] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

[79] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[80] Marwan Hassani, Philipp Kranen, and Thomas Seidl. Precise anytime clustering of noisy sensor data with logarithmic complexity. In *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, 2011.

[81] Haibo He and Yuan Cao. Kernel density estimation with stream data based on self-organizing map. In *IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*, 2011.

[82] Zengyou He, Xiaofei Xu, Shengchun Deng, and Joshua Zhexue Huang. Clustering categorical data streams. *Journal of Computational Methods in Science and Engineering*, 11:185 – 192, 2011.

[83] Alexander Hinneburg and Daniel A. Keim. An eficient approach to clustering in large multimedia databases with noise. In *Fourth International Conference on Knowledge Discovery and Data Mining*, 1998.

[84] Alexander Hinneburg and Daniel A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clusterings. In *25th International Conference on Very Large Data Bases*, 1999.

[85] Ma Hong, Kang Jing, and Liu Li-xiong. Research on clustering algorithms of data streams. In *The 2nd IEEE International Conference on Information Management and Engineering (ICIME)*, 2010.

[86] Guo-Yan Huang, Da-Peng Liang, Chang-Zhen Hu, and Jia-Dong Ren. An algorithm for clustering heterogeneous data streams with uncertainty. *International Conference on Machine Learning and Cybernetics*, 4:2059 – 2064, 2010.

[87] Geoff Hulten and Pedro Domingos. Vfml – a toolkit for mining high-speed time-changing data streams, 2003.

[88] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[89] Elena Ikonomovska, Suzana Loskovska, and Dejan Gjorgjevik. A survey of stream data mining. Technical report, University Ss. Cyril & Methodius, Faculty of Electrical Engineering and Information Technologies, 2007.

[90] NVIDIA Inc., 2011.

[91] Anil K. Jain, M.Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264 – 323, 1999.

[92] Ankur Jain, Zhihua Zhang, and Edward Y. Chang. Adaptive non-linear clustering in data streams. In *15th ACM International Conference on Information and Knowledge Management*, 2006.

[93] N. Jerne. Towards a network theory of the immune system. *Ann. Immunology (Inst. Pasteur)*, 125:373 – 389, 1974.

[94] Chen Jia, ChengYu Tan, and Ai Yong. A grid and density-based clustering algorithm for processing data stream. In *Second International Conference on Genetic and Evolutionary Computing*, 2008.

[95] Hong Jiang, Qingsong Yu, and Dongxiu Wang. A high-dimensional data stream clustering algorithm based on damped window and pruning list tree. In *Biomedical Engineering and Informatics (BMEI)*, 2011.

[96] Madjid Khalilian and Norwati Mustapha. Data stream clustering: Challenges and issues. In *International Conference on Data Mining and Applications (IAENG)*, 2010.

[97] Mahnoosh Kholghi, Hamed Hassanzadeh, and MohammadReza Keyvanpour. Classification and evaluation of data mining techniques for data stream requirements. In *International Symposium on Computer Communication Control and Automation*, 2010.

[98] Philipp Kranen, Felix Reidl abnd Fernando Sanchez Villaamil, and Thomas Seidl. Hierarchical clustering for real-time stream data with noise. *Lecture Notes in Computer Science*, 6809:405 – 413, 2011.

[99] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. Self-adaptive anytime stream clustering. In *Ninth IEEE International Conference on Data Mining*, 2009.

[100] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. The clustree: Indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 29:249 – 272, 2011.

[101] Hans-Peter Kriegel, Peer Kroger, Irene Ntoutsi, and Arthur Zimek. Towards subspace clustering on dynamic data: An incremental version of predecon. In *Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques*, 2010.

[102] Hans-Peter Kriegel, Peer Kroger, Irene Ntoutsi, and Arthur Zimek. Density based subspace clustering over dynamic data. In *23rd International Conference on Scientific and Statistical Database Management*, 2011.

[103] Milos Krstajic, Enrico Bertini, Florian Mansmann, and Daniel A. Keim. Visual analysis of news streams with article threads. In *Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques*, 2010.

[104] Jae Woo Lee, Nam Hun Park, and Won Suk Lee. Efficiently tracing clusters over high-dimensional on-line data streams. *Data & Knowledge Engineering*, 68:362 – 379, 2009.

[105] Nam Hun Parkand Won Suk Lee. Statistical grid-based clustering over data streams. *ACM SIGMOD Record*, 33:32 – 37, 2004.

[106] Elizabeth Leon. *Scalable and Adaptive Evolutionary Clustering for Noisy and Dynamic Data*. PhD thesis, Universiy of Louisville, 2005.

[107] Elizabeth Leon, Jonatan Gomez, and Olfa Nasraoui. A genetic niching algorithm with self-adaptating operator rates for document clustering. In *LA-WEB*, 2012.

[108] Elizabeth Leon, Olfa Nasraoui, and Jonatan Gomez. Anomaly detection based on unsupervised niche clustering with application to network intrusion detection. In *Congress on Evolutionary Computation*, 2004.

[109] Elizabeth Leon, Olfa Nasraoui, and Jonatan Gomez. Ecsago: Evolutionary clustering with self adaptive genetic operators. In *IEEE Congress on Evolutionary Computation*, 2006.

[110] Elizabeth Leon, Olfa Nasraoui, and Jonatan Gomez. Scalable evolutionary clustering algorithm with self adaptive genetic operators. In *Congress on Evolutionary Computation*, 2010.

[111] Yanrong Li and Raj P. Gopalan. Clustering transactional data streams. In *19th Australian joint conference on Artificial Intelligence: advances in Artificial Intelligence*, 2006.

[112] Liu Li-xiong, Huang Hai, Guo Yun-fei, and Chen Fu-cai. rdenstream, a clustering algorithm over an evolving data stream. In *International Conference on Information Engineering and Computer Science*, 2009.

[113] Liu Li-xiong, Kang Jing, Guo Yun-fei, and Huang Hai. A three-step clustering algorithm over an evolving data streams. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 1, pages 160 – 164, 2009.

[114] Guopin Lin and Leisong Chen. A grid and fractal dimension-based data stream clustering algorithm. *International Symposium on Information Science and Engineering*, 1:66 – 70, 2008.

[115] Weiguo Liu and Jia OuYang. Clustering algorithm for high dimensional data stream over sliding windows. In *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2011.

[116] Edwin Lughofer. Extensions of vector quantization for incremental clustering. *Pattern Recognition*, 41:995 – 1011, 2008.

[117] Guan-Chun Luh and Chun-Yi Lin. Pca based immune networks for human face recognition. *Applied Soft Computing*, 11:1743 – 1752, 2011.

[118] Sebastian Luhr and Mihai Lazarescu. Incremental clustering of dynamic data streams using connectivity based representative points. *Data & Knowledge Engineering*, 68:1 – 27, 2009.

[119] Gurmeet Singh Manku, Sridhar Rajagopalan, , and Bruce G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *ACM SIGMOD International Conference on Management of Data*, 1999.

[120] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *ACM SIGMOD International Conference on Management of Data*, 1998.

[121] Wicha Meesuksabai, Thanapat Kangkachit, and Kitsana Waiyamai. Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty. In *Advance Data Mining and Applications. Lecture Notes in Computer Science*, 2011.

[122] Ming ming Gao and Chang Tai hua Xiang-xiang Gao. Research in data stream clustering based on gaussian mixture model genetic algorithm. In *2nd International Conference on Information Science and Engineering (ICISE)*, 2010.

[123] Ming ming Gao, Ji zhen Liu, and Xiang xiang Gao. Application of compound gaussian mixture model clustering in the data stream. *International Conference on Computer Application and System Modeling (ICCASM)*, 7:172 – 177, 2010.

[124] Nina Mishra, Dan Oblinger, and Leonard Pitt. Sublinear time approximate clusterings. In *Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[125] Masahiro Motoyoshi, Takao Miura, and Isamu Shioya. Clustering stream data by regression analysis. In *Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, 2004.

[126] S. Muthu Muthukrishnan. Data streams: Algorithms and applications. In *Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[127] Olfa Nasraoui, Cesar Cardona, Carlos Rojas, , and Fabio Gonzalez. Tecnostreams: Tracking evolving clusters in noisy data streams with a scalable immune system learning model. In *Third IEEE International Conference on Data Mining (ICDM)*, 2003.

[128] Olfa Nasraoui and Raghu Krishnapuram. A novel approach to unsupervised robust clustering using genetic niching. *Proceedings of the Ninth IEEE International Conference on Fuzzy Systems*, 1:170 – 175, 2000.

[129] Olfa Nasraoui and Carlos Rojas. Robust clustering for tracking noisy evolving data streams. In *Computer Engineering*, 2006.

[130] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering,*, 14:1003 – 1016, 2002.

[131] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clusterings. In *18th IEEE International Conference on Data Engineering*, 2002.

[132] Sang-Hyun Oh, Jin-Suk Kang, Yung-Cheol Byun, Gyung-Leen Park, and Sang-Yong Byun. Intrusion detection based on clustering a data stream. In *Third ACIS International Conference on Software Engineering Research, Management and Applications*, 2005.

[133] Kok-Leong Ong, Wenyuan Li, Wee-Keong Ng, and Ee-Peng Lim. Sclope: An algorithm for clustering data streams of categorical attributes. *Lecture Notes in Computer Science*, 3181:209 – 218, 2004.

[134] Carlos Ordonez. Clustering binary data streams with k-means. In *8th ACM SIGMOD Workshop on Research issues in Data Mining and Knowledge Discovery*, 2003.

[135] Nam Hun Park and Won Suk Lee. Cell trees: An adaptive synopsis structure for clustering multi-dimensional on-line data streams. *Data & Knowledge Engineering*, 63:528 – 549, 2007.

[136] Nam Hun Park, Sang Hyun Oh, and Won Suk Lee. Anomaly intrusion detection by clustering transactional audit streams in a host computer. *Information Sciences*, 180:2375 – 2389, 2010.

[137] Thanawin Rakthanmanon, Eamonn J. Keogh, Stefano Lonardi, and Scott Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *IEEE 11th International Conference on Data Mining (ICDM)*, 2011.

[138] Jiadong Ren, Binlei Cai, and Changzhen Hu. Clustering over data streams based on grid density and index tree. *Journal of Convergence Information Technology*, 6:83 – 93, 2011.

[139] Jiadong Ren, Changzhen Hu, and Ruiqing Ma. Hcluwin: An algorithm for clustering heterogeneous data streams over sliding windows. *International Journal of Innovative Computing, Information and Control*, 6:2171 – 2179, 2010.

[140] Jiadong Ren and Ruiqing Ma. Density-based data streams clustering over sliding windows. In *Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, 2009.

[141] Pedro Pereira Rodrigues and JoÃ£o Gama. Online prediction of clustered streams. In *Fourth International Workshop on Knowlegde Discovery from Data Streams*, volume 11, pages 251 – 252, 2007.

[142] Pedro Pereira Rodrigues, JoÃ£o Gama, and Joao Pedroso. Hierarchical clustering of time-series data streams. *IEEE Transactions on Knowledge and Data Engineering*, 20:615 – 627, 2008.

[143] Josep Roure and Luis Talavera. Robust incremental clustering with bad instance orderings: A new strategy. In *Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence*, 1998.

[144] Carlos Ruiz, Ernestina Menasalvas, and Myra Spiliopoulou. C-denstream: Using domain knowledge on a data stream. In *12th International Conference on Discovery Science*, 2009.

[145] Rahul Shah, Shonali Krishnaswamy, and Mohamed Medhat Gaber. Resource-aware very fast k-means for ubiquitous data stream mining. In *2nd International Workshop on Knowledge Discovery in Data Streams*, 2005.

[146] Ravi Shankar, Kiran G. V. R., and Vikram Pudi. Evolutionary clustering using frequent itemsets. In *Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques*, 2010.

[147] Xian Shen, X. Z. Gao, Rongfang Bie, and Xin Jin. Artificial immune networks: Models and applications. *International Conference on Computational Intelligence and Security*, 1:394 – 397, 2006.

[148] Zaigham Faraz Siddiqui and Myra Spiliopoulou. Combining multiple interrelated streams for incremental clustering. In *21st International Conference on Scientific and Statistical Database Management*, 2009.

[149] Zaigham Faraz Siddiqui and Myra Spiliopoulou. Stream clustering of growing objects. *Lecture Notes in Computer Science*, 5808:433 – 440, 2009.

[150] Mingzhou Song and Hongbin Wang. Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. *Intelligent Computing: Theory and Applications III*, 5803:174 – 183, 2005.

[151] Ashok N. Srivastava and Julienne Stroeve. Onboard detection of snow, ice, clouds and other geophysical processes using kernel methods. In *Workshop on Machine Learning Technologies for Autonomous Space Applications*, 2003.

[152] Bernhard Stegmaier, Richard Kuntschke, and Alfons Kemper. Streamglobe: Adaptive query processing and optimization in streaming p2p environments. In *Proceeedings of the 1st International Workshop on Data Management for Sensor Networks*, 2004.

[153] Michael Stonebraker, Ugur Cetintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 34:42 – 47, 2005.

[154] Jiaowei Tang. An algorithm for streaming clustering. Master's thesis, Uppsala University, 2011.

[155] Dimitris K. Tasoulis, Niall M. Adams, and David J. Hand. Unsupervised clustering in streaming data. In *Sixth IEEE International Conference on Data Mining Workshops*, 2006.

[156] Li Tu and Yixin Chen. Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3:1 – 27, 2009.

[157] Q. Tu, J.F. Lu, B. Yuan, J.B. Tang, and J.Y. Yang. Density-based hierarchical clustering for streaming data. *Pattern Recognition Letters*, 33:641 – 645, 2012.

[158] Komkrit Udommanetanakit, Thanawin Rakthanmanon, and Kitsana Waiyamai. E-stream: Evolution-based technique for stream clustering. *Lecture Notes in Computer Science*, 4632:605 – 615, 2007.

[159] Li Wan, Wee Keong, Xuan Hong Dang, Philip S. Yu, and Kuan Zhang. Density-based clustering of data streams at multiple resolutions. In *ACM Transactions on Knowledge Discovery from Data*, 2009.

[160] Renxia Wan and Lixin Wang. Clustering over evolving data stream with mixed attributes. *Journal of Computational Information Systems*, 6:1555 – 1562, 2010.

[161] Renxia Wan, Xiaoya Yan, and Xiaoke Su. A weighted fuzzy clustering algorithm for data stream. *International Colloquium on Computing, Communication, Control, and Management*, 1:360 – 364, 2008.

[162] C. Wang, J. Lai, D. Huang, and W. Zheng. Svstream: A support vector based algorithm for clustering data streams. In *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[163] Shuyun Wang, Yingjie Fan, Chenghong Zhang, HeXiang Xu, Xiulan Hao, and Yunfa Hu. Entropy based clustering of data streams with mixed numeric and categorical values. In *Seventh IEEE/ACIS International Conference on Computer and Information Science*, 2008.

[164] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, 1997.

[165] Niwan Wattanakitrungroj and Chidchanok Lursinsap. Memory-less unsupervised clustering for data streaming by versatile ellipsoidal function. In *20th ACM International Conference on Information and Knowledge Management*, 2011.

[166] Dwi H. Widyantoro, Thomas R. Ioerger, and John Yen. An incremental approach to building a cluster hierarchy. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, 2002.

[167] Xueyan Wu and Daoping Huang. Data stream clustering for stock data analysis. In *2nd International Conference on Industrial and Information Systems (IIS)*, 2010.

[168] Chunyu Yang and Jie Zhou. Hclustream: A novel approach for clustering evolving heterogeneous data stream. In *Sixth IEEE International Conference on Data Mining Workshops*, 2006.

[169] Di Yang. *Mining and Managing Neighbor-Based Patterns in Data Streams.* PhD thesis, Worcester Polytechnic Institute, 2012.

[170] Jing Yang, Wenxin Zhu, Jianpei Zhang, and Yue Yang. Data stream clustering algorithm based on active grid density. In *Fifth International Conference on Internet Computing for Science and Engineering (ICICSE)*, 2010.

[171] Jiong Yang. Dynamic clustering of evolving streams with a single pass. In *19th International Conference on Data Engineering*, 2003.

[172] Yiling Yang, Xudong Guan, and Jinyuan You. Clope: A fast and effective clustering algorithm for transactional data. In *Knowledge Discovery and Data Mining*, 2002.

[173] Yun Yang and Ke Chen. Temporal data clustering via weighted clustering ensemble with different representations. *IEEE Transactions on Knowledge and Data Engineering*, 23:307 – 320, 2011.

[174] Feng Yu, Damalie Oyana, Wen-Chi Hou, and Michael Wainer. Approximate clustering on data streams using discrete cosine transform. *Journal of Information Processing Systems*, 6:67 – 78, 2010.

[175] Chen Zhang, Ming Gao, and Aoying Zhou. Tracking high quality clusters over uncertain data streams. In *IEEE 25th International Conference on Data Engineering*, 2009.

[176] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1996.

[177] Xiangliang Zhang, Cyril Furtlehner, and MichÃ¨le Sebag. Data streaming with affinity propagation. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2008.

[178] Shi Zhong. Efficient online spherical k-means clustering. In *IEEE International Joint Conference on Neural Networks*, 2005.

[179] Shi Zhong. Efficient streaming text clustering. *Neural Networks*, 18:790 – 798, 2005.

[180] Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, 15:181 – 214, 2008.

[181] Aoying Zhou, Feng Cao, Ying Yan, Chaofeng Sha, and Xiaofeng He. Distributed data stream clustering: A fast em-based approach. In *23rd International Conference on Data Engineering*, 2007.

[182] Haiyan Zhoua, Xiaolin Baib, and Jinsong Shana. A rough-set-basedclustering algorithm for multi-stream. *Procedia Engineering*, 15:1854 – 1858, 2011.