VARMOG: A Co-Evolutionary Algorithm to Identify Manifolds on Large Data

Héctor D. Menéndez
Computer Science Department
University College London
London, United Kingdom
h.menendez@ucl.ac.uk

***Abstract*—Detecting clusters defining a specific shape or manifold is an open problem and has, indeed, inspired different machine learning algorithms. These methodologies normally lack scalability, as they depend on the performance of very sophisticated processes, such as extracting the Laplacian of a similarity graph in spectral clustering. When the algorithms need not only to identify manifolds on large amounts of data or streams, but also select the number of clusters, they failed either because of the robustness of their processes or by computational limitations. This paper introduces a general methodology that works in two levels: the initial step summarizes the data into a set of relevant features using the Euclidean properties of manifolds, and the second applies a robust methodology based on a co-evolutionary multi-objective clustering algorithm that identifies both, the number of manifolds and their associated manifold. The results show that this method outperforms different state of the art clustering processes for both, benchmark and real-world datasets.**

***Index Terms*—Clustering, Map Reduce, Evolutionary Computation, Big Data, Stream**

I. Introduction

On this new era of Big Data algorithms, analysts need to deal not only with the problem of extracting patterns from data, but also with the efficiency required by the markets demands. These challenges are known as the five V's [1]: velocity, variety, volume, veracity and value extracted.

As Big Data is an area that covers several different fields [2], [3], in this work, I focus on the problem of clustering large datasets, i.e. a dataset with a large number of samples. Besides, the main goal of this work is not only the discrimination of clusters, but also identifying specific structures hidden into the clusters. These structures, called manifolds, have the property of continuity. Consequently, every clustering algorithm targeting these kinds of structures needs not only to deal with the stress associated with high volumes of data, but also with the notion of continuity immerse inside the manifolds.

There are several methodologies that deal with the manifold extraction problem [4]. The most relevant are continuity-based clustering techniques. These methodologies try to separate the data according to the form they define in the space [5]. The most representative technique is Spectral Clustering [6]. This technique defines a similarity graph among the data in order to find the best way to cut it. The resulting components after the cutting process are the manifolds.

Spectral Clustering has several problems according to its robustness [5] and memory consumption [7]. The latter is specially important for large data analysis, due to the algorithm consumes a lot of memory during the search process. There are some methodologies, such as the Nyström extension [8], which tries to reduce the memory usage of Spectral Clustering, however, this technique makes the algorithm more sensitive to noisy data.

Other techniques which are gaining importance in large data analysis are online clustering algorithms [9]. These analytical techniques are focused on data streams. Data streams generate data continually. The idea behind online clustering algorithms is to analyse these data using real-time analysis [9]. These techniques usually need to deal with large data quantities that produce several limitations on the algorithm representation. One of the main tools used for online clustering analysis is the Massive On-line Analysis (MOA) tool [10].

This work is focused on combining a co-evolutionary Multi-Objective Genetic Algorithms (MOGA) [11] and Map-Reduce [12] to improve the manifold identification process, allowing the algorithm not only to identify manifolds but also to choose the number of clusters automatically. This new algorithm, called VARMOG, divides the clustering process into two levels: the low level summarizes the information of the original search space using the Voronoi regions, while the high-level uses a co-evolutionary multi-objective approach to join the regions, discriminating the manifold structure. This approach extends the MOGCLA algorithm [13] that needs to know the number of cluster in advance from the analyst. This extension includes a co-evolutionary process where different populations collaborate with the aim of identifying the most stable solutions for different numbers of clusters automatically.

The experimental section compares VARMOG with the MOGCLA algorithm, the classical K-means version of Map-Reduce [12], the combination of Spectral Clustering and the Nyström method [8] and modern online clustering algorithms (CluStream [14], Online K-means [9] and ClusTree [15]), in order to evaluate the performance of the new method on benchmarks and real-world datasets. The results show that the algorithm is competitive with respect to the other methods, and it has the ability to automatically detect the proper number of clusters in almost every case.

This work is structured as follows: Section II introduces

the motivation of this work specially focused on the Voronoi Tessellation, Section III presents the VARMOG algorithm which is evaluated in Section IV using benchmark and real-world datasets, and comparing its accuracy value against other algorithms extracted from the literature. Finally, the paper presents the conclusions and future work.

II. Background

The main hypothesis of this work comes from the definition of manifold. The topological definition of a manifold describes it as a *space where each point is locally in an Euclidean space* [16]. This is a strong advantage for large quantities of data, as although there are several points in the space, lots of them will fall in the neighbourhood of an Euclidean space. This goes to the main research question: Is it possible to summarize the space into a set of points which are representative points of the whole dataset?

First of all, it is important to consider the type of data to be analysed. In this work I focus on stream data, i.e. continues data coming from different sources. There are two main ways to deal with these data: *offline*, where the analyst sets a time frame and directly analyses the data using a significant amount of it; or *online* where the data are analysed as they arrive to the system. This work focuses on offline data, although my aim is to extend this idea also for online environments.

Based on the former hypothesis for space summarization, and considering that the data flow is offline, my method will divide the analysis into two parts: first, it will reduce the information of the space and, later, it will identify the manifold directly on this reduced information. Fig. 1 shows an example of this reduction and how the main points (blue dots) would be selected to reconstruct the original shapes. As I need to create this data summary in a timely way, I leverage the Voronoi Tesselation.

A. Clustering and The Voronoi Tessellation

The Voronoi Tessellation is a geometric construction that allows to reconstruct an Euclidean plane. In the case of clustering, the Tessellation represents the partition produced by the clustering algorithm, when this partition is based on an Euclidean notion. Formally, given a dataset $X = \{x_1, \ldots, x_n\}$ with n elements, the clustering process divides the data blindly into k clusters ($C = \{c_1, \ldots, c_k\}$). The partition C is also known as *Voronoi Tessellation* or *Dirichlet Tessellation* [17]. Considering that each cluster has a centroid, the set of centroids is $V = \{v_1, \ldots, v_k\}$. This process generally minimizes a cost function which, in the case of an Euclidean space, is:

$$J = \sum_{c_j \in C} \sum_{x_i \in c_j} ||x_i - v_j||^2 \tag{1}$$

where the distance denoted by $||\cdot||$ is the Euclidean distance. The minimization aims to identify, fixed the number of clusters k, the best position for the centroids. This translates to:

$$\arg \min_{c_1,\ldots,c_k} J = \arg \min_{v_1,\ldots,v_k} \sum_{c_j \in C} \sum_{x_i \in c_j} ||x_i - v_j||^2 \tag{2}$$

Giving this notion of centrality, it is straightforward to define a Voronoi Tesselation considering it in terms of distance to the centroid. The region R_j defined by the centroid v_j is the closest set of points to this centroid, i.e., considering a d dimensional space $\mathbb{R}^d$:

$$R_i = \{z \in \mathbb{R}^d \mid i = \arg \min_j ||z - v_j||^2\}. \tag{3}$$

This partition provides areas where the data density is higher and determines regions that will work as Euclidean pieces of the manifold's puzzle. Once the whole space is divided into regions, they only need to be joined, defining a continuity criterion. The region center is a strong piece of information as it provides the means of all the data instances inside the region. It leads to consider the centroids as the starting point to find manifolds from a high level perspective. This reduces the effort for the manifold-based algorithms, as they do not need to deal with noisy data, as far as the centroids are stable, their results must be as good as the original data.

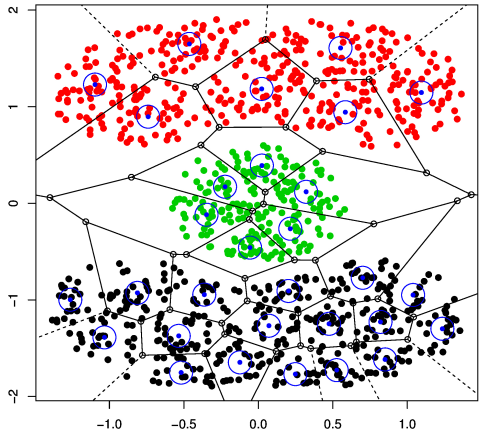

Fig. 1. Voronoi tessellation of Cassini dataset applying K-means.

Fig. 1 shows an example of how the Voronoi Tesselation can summarize a dataset of 10,000 instances into 30 centroids. This dataset, called Cassini (Section IV-A), contains 3 clusters that can hardly be identified, for example, by a 3-means, as two of them are surrendering the third. In this specific example, the centroids are spared around the three clusters, defining the regions where the main pieces of the dataset are. Here, there are no data instances of two different clusters in a region, this strongly depends on the number of regions defined at the beginning. For grouping the regions, the centroids just need to be joined by continuity. This will combine two kinds of clustering at two different levels: the lowest level will focus on defining the regions using the Euclidean properties of manifolds and the highest level will reconstruct the manifold by continuity. This two-level clustering process allows the identification of manifolds.

III. VARMOG: Manifold Clustering via Co-Evolution

This section describes the Co-evolucionary Multi-Objective Genetic Graph-based Clustering Algorithm (VARMOG). This algorithm combines the Map-Reduce version of K-means and a cooperative co-evolutionary approach between populations. This co-evolution uses the Multi-Objective Genetic Graph-based Clustering algorithm [7] (MOGGC) in each individual population, in order to reduce the computationally effort and improve the scalability for large datasets. Fig. 2 shows the schema of the whole process.

A. Low-level: Space Summarization

The lowest level of the algorithm defines the Voronoi regions in the form of centroids. They are extracted using a Map-Reduce version of K-means. This information is sent to the next level in order to improve scalability.

The Map-Reduce version of K-means is divided in three main steps (see Fig. 2):

1) **Ini:** This step splits the data among nodes creating different blocks. It also chooses an initial set of centroids uniformly at random. Every node keeps a copy of these centroids which will be updated in every step (see Fig. 2 *Ini*).
2) **Map:** Each node maps the data instance with the closest centroid by using the Euclidean distance among them. This groups the data block by centroid. After, the information of each centroid will be the same to the same reducer, e.g. the instances from different mappers associated to centroid 1 will be sent to reducer 1, those of centroid 2 to reducer 2 and so on (see Fig. 2 *Map*).
3) **Reduce:** As every reducer has all the data corresponding to a centroid they can independently recalculate the centroid position. This is calculated as the means of the data, i.e.:

$$v_i = \frac{1}{|c_i|} \sum_{x_j \in c_i} x_j.$$

Then, the centroids list is updated and sent to all the nodes again (see Fig. 2 *Reduce*). When the algorithm reaches a convergence criterion or a maximum number of iterations, it stops. Otherwise, it goes to the map step.

Once this process finishes, the algorithm produces the Voronoi regions, defined as centroids, and sends the centroids to the high level process.

B. High-level: Manifold Discrimination

The high-level clustering process uses the centroids as a search space to apply clustering. But it has no knowledge of the number of clusters. To guess the number of clusters it will create a population per potential number of clusters inside a range: $[k_{min}, k_{max}]$. Each population will calculate potential solutions in every generation based on a multi-objective fitness. They will also collaborate by exchanging individuals from the Pareto Fronts at the end of each generation.

The algorithm starts defining a similarity graph between the centroids. The centroids are nodes of the graph, and the weight of the edges are their similarity, measured by the Radial Basis Function (RBF) between each pair of centroids [6]. The algorithm only keeps the K-closest nodes connected to reduce the amount of information, defining a K-neighbour similarity matrix [6]. This topology will define the final manifold/clusters using a graph-cut methodology. To cut the graph, I use a genetic algorithm: the MOGGC algorithm [7].

This algorithm leverages SPEA2 algorithm [18] for the genetic search process of the set of solutions, i.e., selections from centroids of the low level to manifolds in the high level. To deal with the co-evolutionary process, I extend SPEA2 to contain an archive per population of the co-evolution (each archive defines independent Pareto Fronts per number of clusters). It is also extended to exchange solutions from the archives at the end of each generation, so it can optimize the search process.

This algorithm has the advantage that several parts can run in parallel. Each population runs in a different core. Also each chromosome fitness is calculated in different machines, as the chromosomes do not need information from each other during the fitness evaluation. The selection divides the chromosomes in order to reproduce them among different machines.

1) Encoding and Genetic operators: Each chromosome is encoded as an N-dimensional vector, where each allele represents a number between 1 and the number of clusters assigned to the specific population. This is called a label-based representation [19]. These labels are considered clustering solutions.

The evolution might create invalid individuals. These solutions are chromosomes with empty clusters. As the algorithm works with partitional clustering, these solutions are removed from the population, assigning them the minimum fitness.

Each population applies the following operators to their chromosomes:

- **Selection**: A tournament selection chooses the best individual for reproduction.
- **Crossover**: The algorithm applies a two point crossover, exchanging strings of numbers of the same length.
- **Mutation**: The algorithm follows an adaptive mutation that improves the convergence. It is calculated in three steps:
  1) For each individual, it uniformly estimates if the mutation is applied. This probability is initially fixed.
  2) After, for each chromosome, it decides which alleles are mutated taking into account the current element probability to belong to the assigned cluster. The higher this probability, the lower the mutation probability and vice versa. This probability is calculated using the fitness function on one allele.
  3) It mutates the element by replacing it using a uniformly at random number between 1 and the number of clusters of that population.

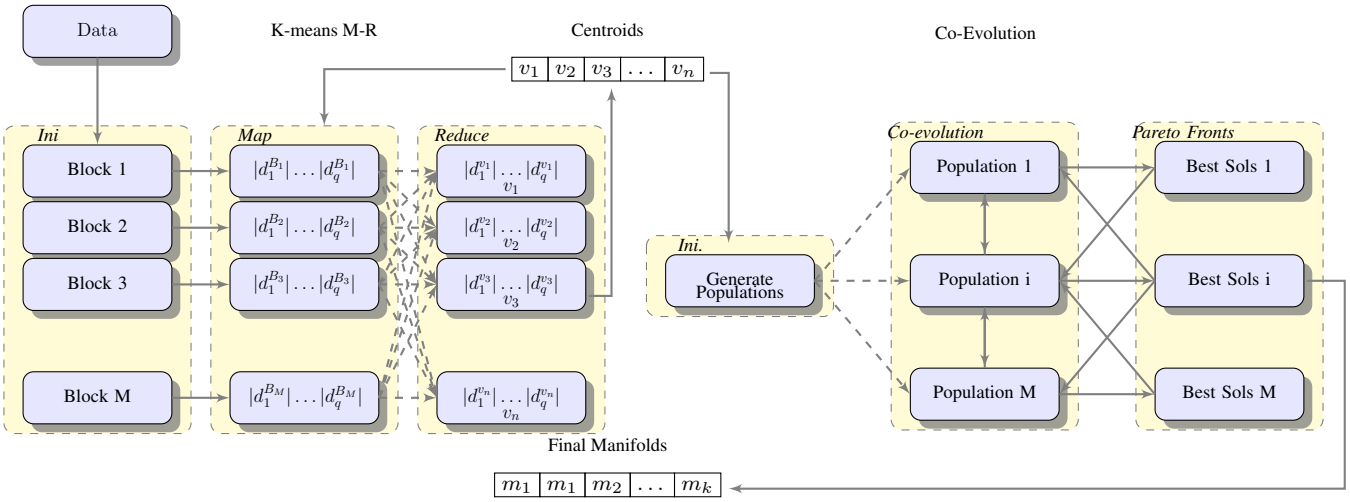

Fig. 2. Architecture of the algorithm. The first part corresponds with the Map-Reduce implementation of the low level search, where the centroids position is optimized. The second part corresponds with the co-evolution. The algorithm generates a population per number of clusters and, then, it creates different Pareto Fronts whose archives exchange the best solutions.

2) The Fitness Function: The fitness has two objectives. First, it aims to improve the data continuity degree to create clusters whose intra-data are well connected. The second objective aims to increment as much as possible the cluster separation. They are computed as follows:

- **Data Continuity Degree**: This objective function calculates the sum of all the edges, creating a minimal spanning tree for each connected component of the similarity graph.
- **Clusters Separation**: This metric measures the cost of separating a node from a cluster based on the weights of other cluster nodes. It is computed as:

$$\frac{\sum_{v_i \in C} \frac{\sum_{v_j \in G}\{w_{ij} |\ v_j \notin C\}}{|G|-|C|}}{|C|} \tag{4}$$

where C is a cluster, G is the similarity graph, v_i is the centroid i, w_{ij} is the edge weight value from i to j. It calculates the arithmetic average value of the edge weights among different clusters.

These two objectives are opposite. They improve the intra and inter cluster distances, respectively. This is the reason for choosing a multi-objective approach. For the first objective, a single cluster would guarantee a maximum value while, for the second, a cluster per instance would guarantee the maximum value.

Due the necessity of choosing one of the solutions from the Pareto Front, I chose the solution with the highest value of the Cluster Separation metric, because, empirically, this solution always obtains better accuracy values.

C. Time and Memory Complexity

This section focuses on the complexity and scalability of VARMOG. It is relevant for the analysis to take into account that the first step significantly reduces the amount of data for the second, improving the scalability on the second one. Besides, for the first step I implemented the Map-Reduce version of K-means which is already scalable and whose complexity only depends on the number of clusters and data points (although, in practice, it is also important to have into account the messages exchanged between mappers and reducers).

For the low-clustering process, the complexity is completely equivalent to the complexity of the Map-Reduce version of K-means. According to [12], this complexity is $O(k \cdot N/p)$, where k is number of clusters, N is number of data points and p is number of parallel nodes.

The high-clustering process has some extra dependencies. The co-evolutionary algorithm depends on: the initial populations that are collaborating between them, their chromosomes, operations and fitness calls. The number of populations does not affect to the complexity, as each population can be executed in parallel. The fitness function has a direct dependency on the number of regions. In the worst case scenario the fitness consumes $O(r^2)$, where r is the number of regions, as it is a square computation on the regions, i.e. it considers the regions by pairs (see equation 4). SPEA2 complexity depends on the population and archive sizes (P and $\bar{P}$, respectively). It is calculated as $O((P+\bar{P})^2 \cdot \log(P+\bar{P}))$. I also need to take into account that the algorithm will process G sequential generations.

Considering that the two levels of the algorithm are sequential, as the number of centroids is the input for the co-evolution, the complexity of the algorithm is:

$$O(k \cdot N/p + G \cdot (r^2 + (P+\bar{P})^2 \cdot \log(P+\bar{P}))) \tag{5}$$

Considering that the algorithm will run in parallel, the estimation for the memory consumption of the Map-Reduce version of K-means is $N + p \cdot r$ [12], where N is the size of the

dataset, p represents the nodes used during the Map-Reduce execution, and r is the number of centroids. For the memory consumption of the co-evolution, I consider the population size P and the number of populations C, apart of the number of neighbours chosen in the similarity matrix K. This leads to a consumption of $P \cdot C \cdot r \cdot K$. However, it is important to consider the size reduction where $r << N$. The total memory consumption is:

$$N + p \cdot r + P \cdot C \cdot r \cdot K \tag{6}$$

IV. Experimental Results

VARMOG evaluation is divided in two parts: the first one compares the performance of the algorithm on different synthetic benchmarks, extracted from the literature, which are composed by different continuity based-clusters. The second measures the performance of the algorithm on real-world datasets.

A. Dataset Description

The benchmarks belong to the *mlbench*[1] package of the R-project. Using this package, I have generated 8 datasets containing 50,000 instances each. The description of these datasets is the following (Fig. 3):

- **Cassini**: Three continuity based clusters, two larger at the top and bottom and one smaller in the middle.
- **Cuboids**: Three cuboids defining a frame in 3 dimensions, and one smaller cube in the middle.
- **Hypercube**: A three dimensional dataset composed by 8 spheres in two levels, divided by four spheres per level.
- **Shapes**: A dataset containing 4 different shapes in two dimensions.
- **Simplex**: Similar to hypercube, but this dataset has four spheres, 3 on the bottom level and one on the top.
- **Smiley**: A two dimensional dataset defining a smiley face.
- **Spirals1**: A noiseless dataset defining two spirals around each other.
- **Spirals2**: The previous dataset but affected by noise.

For the experiments on real world data, I have selected four datasets from different repositories. These datasets are:

- Forest Covertype Dataset (**CovType**): extracted from the UCI Machine Learning repository. This dataset uses cartographic variables to predict forest cover type. It is defined by 54 features, targeting 7 classes. The total number of instances is 581012.
- **Electricity**[2]: This dataset contains electricity consumption records for different prices. It contains 45312 instances described by 5 features and targeting 2 classes.
- **Statlib**[3]: The statlib dataset describes the stock of Dow Jones 30. It has 6 features, 138166 samples and 30 classes.

[1]http://cran.r-project.org/web/packages/mlbench/mlbench.pdf
[2]http://www.inescporto.pt/~jgama/ales/ales_5.html
[3]http://tunedit.org/repo/StatLib/numeric/dj30-1985-2003.arff

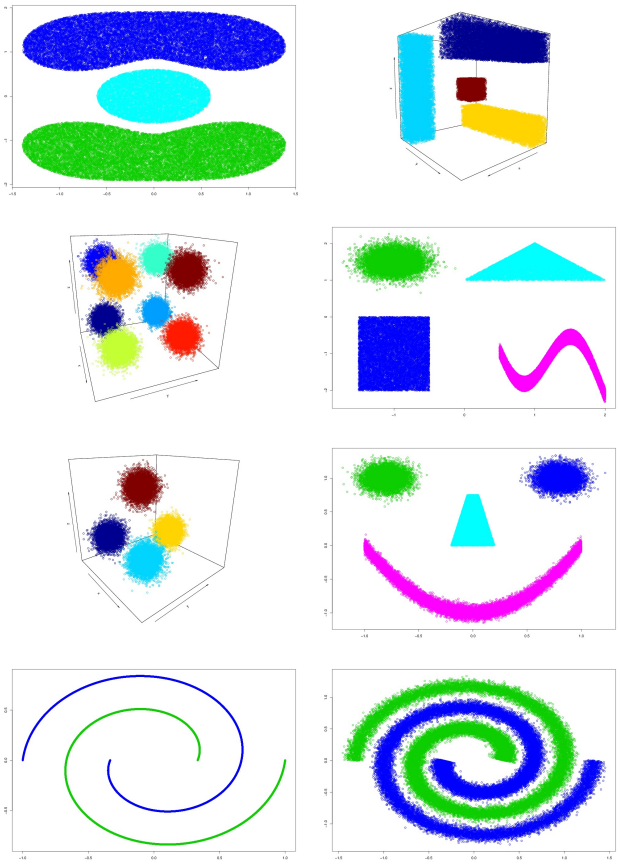

Fig. 3. The synthetic datasets. From top to bottom and from right to left: Cassini, Cuboids, Hypercube, Shapes, Simplex, Smiley, Spirals1, Spirals2.

- Records **Block** (UCI Machine Learning Repository): This dataset contains the underlying records from the epidemiological cancer registry in North Germany. It has 2 classes, 4 attributes and 574913 instances.

B. Experimental Setup

The evaluation of the clustering algorithm is a sensitive process, specially when the algorithm deals with large datasets. In this work, I have focused the evaluation on comparing the algorithm with other algorithms through the accuracy of the clustering solutions.

During the experiments, I compared the new implementation with different algorithms that can handle big amounts of data. These algorithms are: the MOGCLA algorithm [13], the Nyström extension of Spectral Clustering [8] (SC+N) - that samples the complete dataset to reduce its size, K-means modification using Map-Reduce [12] (Big K-means) and three online clustering algorithms: CluStream [14], Online K-means [9] and ClusTree [15]. As VARMOG, the online algorithms require to work in two levels. For the top level, which defines the solution, they apply the KNN algorithm [20] to define the final clusters.

All algorithms apply the Euclidean distance as their similarity metric. The genetic algorithms uses the following

parameters. The number of populations generated for the co-evolution is 10 (each population describes a different number of clusters). Each population contains 300 individuals which pass to the next generation by using a $(\mu + \lambda)$ selection. The elitism is set to 5, and the exchange parameter between populations is set to 2 per generation. The algorithm runs for 100 generations, applying, at the end of each generation, the mutation and the crossover with probabilities 0.1 and 0.5, respectively. These values were chosen from a grid of parameters, and they are a good trade off between accuracy and speed.

For the high level of VARMOG and MOGCLA, I use the Radial Basis Function (RBF) [21] to define the similarity matrix that the algorithm uses. For the Nyström reduction, I sample the set up to 0.3% of the total dataset using uniform sampling. In order to evaluate the statistical significance between the results of VARMOG and the rest of the algorithms, I apply the Wilcoxon test [22]. The test considers that two sets of results are significantly different when the p-value is smaller than 0.05.

The distributions for the results are obtained by running each algorithms 100 times, as all of them are non-deterministic.

C. Experiments with Benchmark Datasets

The benchmark datasets aim to highlight different abilities of the algorithms depending on how they are able to identify the clusters under different circumstances. I compare the performance of VARMOG with similar algorithms in the area (Section IV-B). Table I (top) shows the results for all the algorithms. In the case of VARMOG, as it works with a variable number of clusters, the table also shows the final number of clusters chosen from the Pareto Front.

Initially, the algorithm combines the different Pareto Front of the populations created during the co-evolutionary search (Section III-B). The selection process prioritizes the clustering separation metric over the continuity degree. This provides a clear selection criteria among the multiple populations. Fig. 4 shows three examples of Pareto Fronts from three different benchmarks. The Pareto Front of the chosen population is highlighted with a red start. In the case of Cassini and Cuboids, the five populations correspond to number of clusters from 2 to 6, where the red one represents 3 clusters for Cassini and 4 for Cuboids. For Hypercube, they go from 6 to 10, where the red one corresponds to 8. As Fig. 3 shows, the selection in the Pareto Front corresponds to the optimum number of clusters on the three benchmarks, while the other number of clusters normally reduces either the continuity or the cluster separation, creating dominated solutions with respect to the optimum one. This shows that the fitness function provides the ability to find the best individuals according to different numbers of clusters.

As the Pareto Front evaluation shows that the number of clusters can be reasonably identified by the co-evolutionary search, I can compare with the rest of the algorithm fixing their number of clusters to the optimum one. This comparison, shown in Table I, shows that VARMOG obtains similar results to MOGCLA in several cases, as it is expected, but in some of them, these results are slightly worse. To understand the weaknesses of the algorithm, it is relevant to analyse each dataset, individually.

The application of VARMOG to **Cassini** shows good accuracy results (the median accuracy is 100%) and a good selection on the number of clusters (as is also shown in Fig. 4). This results are similar to the results of MOGCLA, which is based on the same principles but require an initial number of clusters. For the rest of the tools, Big K-means and the online clustering algorithms obtain the worse results (around 66%) while SC+N obtains good results, very similar to the results of VARMOG (98.6%). There is no statistically significant difference between VARMOG, MOGCLA and SC+N, while VARMOG is significantly better than the others.

In the case of **Cuboids**, the results are slightly worse than in the previous case. The results of CluStream (100%) and SC+N (99.7%) are the best in this case, but both VARMOG and MOGCLA obtain better results than Big K-means and the online clustering techniques (between 72% and 89%). This is also confirmed by the Wilcoxon test.

The **Hypercube** dataset shows maximum results for VARMOG, MOGCLA and CluStream. Besides, for this dataset both VARMOG and MOGCLA achieves maximum stability (0 standard deviation). The rest of the algorithms have accuracy results around 80%. This might be a consequence of the noisy behaviour of this specific dataset. Again, the Wilcoxon test confirms the improvement with statistical significance.

For **Shapes** every algorithm obtains the maximum accuracy results (100%) but SC+N algorithm (68.7%). It is interesting to remark that VARMOG, MOGCLA and ClusStream are the most stable (0.044 and 0 standard deviation). The Wilcoxon test confirms that SC+N is statistically worse.

The **Simplex** dataset has a clear discrimination that every algorithm is able to identify with maximum accuracy. In this specific case, the most stable results are for VARMOG, MOGCLA, SC+N and CluStream. The Wilcoxon test shows no significant different among the algorithms.

In the case of **Smiley**, it is noticed that there are differences between the results of MOGCLA and the results of VARMOG. Although VARMOG successfully identifies the number of clusters, its accuracy is significantly lower (92.3%) according to the Wilcoxon test. Nevertheless, VARMOG is still better than the rest of the algorithms. CluStream (91.6% of accuracy) has similar results to VARMOG (there is no statistical significant difference between them). The rest are significantly worse (around 60 and 75% of accuracy).

The case of **Spiral1** is similar to the former, VARMOG obtains worse results than MOGCLA (89.1% for VARMOG and 100% for MOGCLA), and also than SC+N. However, it correctly identifies the number of clusters (2) and its results are significantly better than the rest of the algorithms.

The last benchmark, **Spiral2**, introduces noise to the previous one. In this case, the results of every algorithm are more confuse. The best algorithm is MOGCLA (76.9%),

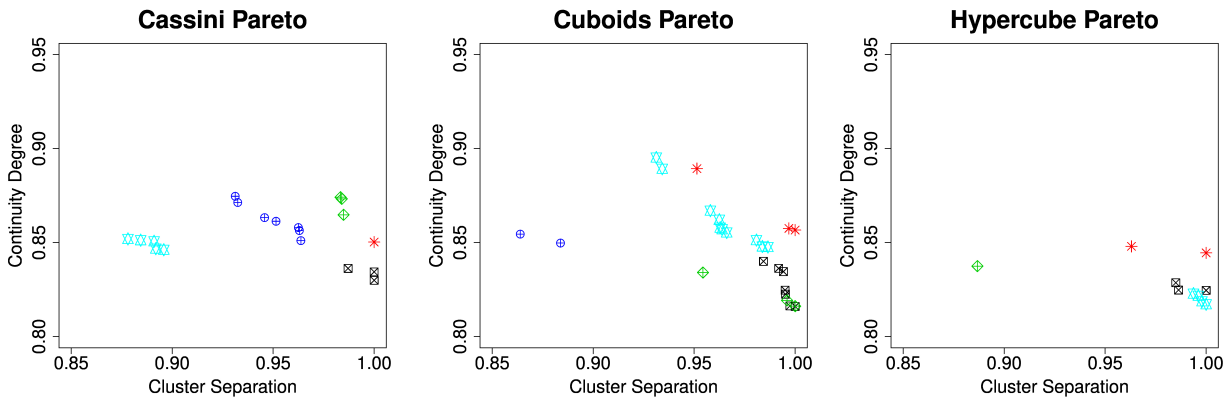

Fig. 4. Representation of the Pareto Front associated with different populations during the co-evolution.

TABLE I

MEDIAN ACCURACY AND STANDARD DEVIATION FOR THE COMPARATIVE RESULTS USING BOTH, BENCHMARK DATASETS AND REAL-WORLD DATASETS. BOLD HIGHLIGHTS THE BEST RESULTS WHILE ITALICS THE SECOND. ▲SYMBOL REPRESENTS THOSE CASES WHERE THE WILCOXON TEST DETERMINES THAT THERE ARE SIGNIFICANTLY IMPROVEMENTS OVER VARMOG, AND ▼REPRESENTS THE OPPOSITE.

Dataset	VARMOG	MOGCLA	Big K-means	SC+Nyström	O-Kmeans	ClusTree	CluStream
Synthetic							
Cassini	**100%** ± 0.04 (3)	**100%** ± 0.04	▼65.5% ± 0.05	*98.6%* ± 0.17	▼65.5% ± 0.00	▼67.0% ± 0.13	▼67.4% ± 0.05
Cuboids	91.4% ± 0.08 (4)	91.4% ± 0.08	▼87.2% ± 0.09	▲*99.7%* ± 0.17	▼88.8% ± 0.12	▼72.7% ± 0.11	▲**100%** ± 0.12
Hypercube	**100%** ± 0.00 (8)	**100%** ± 0.00	▼81.6% ± 0.12	▼76.7% ± 0.16	▼81.4% ± 0.11	▼82.3% ± 0.11	**100%** ± 0.05
Shapes	**100%** ± 0.04 (4)	**100%** ± 0.04	**100%** ± 0.17	▼68.7% ± 0.44	**100%** ± 0.14	**100%** ± 0.18	**100%** ± 0.00
Simplex	**100%** ± 0.00 (4)	**100%** ± 0.00	**100%** ± 0.15	**100%** ± 0.00	**100%** ± 0.15	**100%** ± 0.17	**100%** ± 0.00
Smiley	*92.3%* ± 0.03 (4)	▲**100%** ± 0.05	▼63.5% ± 0.18	▼75.0% ± 0.17	▼62.5% ± 0.17	▼64.4% ± 0.16	91.6% ± 0.15
Spirals1	89.1% ± 0.11 (2)	▲**100%** ± 0.10	▼50.0% ± 0.00	▲**100%** ± 0.10	▼50.0% ± 0.00	▼50.0% ± 0.00	▼50.0% ± 0.00
Spirals2	*63.3%* ± 0.02 (3)	▲**76.9%** ± 0.18	▼59.2% ± 0.00	▼*59.6%* ± 0.04	▼59.4% ± 0.00	▼58.8% ± 0.03	▼59.6% ± 0.04
Real							
Covtype	27.1% ± 0.09 (5)	▲**30.7%** ± 0.08	▼23.3% ± 0.01	▲*29.3%* ± 0.07	▼23.3% ± 0.01	▼19.2% ± 0.03	▼19.0% ± 0.04
Electricity	**61.3%** ± 0.04 (2)	**61.3%** ± 0.04	▼52.2% ± 0.01	*60.0%* ± 0.12	▼52.2% ± 0.01	▼52.2% ± 0.01	▼52.2% ± 0.00
Statlib	5.01% ± 0.08 (12)	▲8.24% ± 0.01	▲**9.99%** ± 0.00	▲9.95% ± 0.00	▲*9.98%* ± 0.00	▲9.76% ± 0.00	▲8.28% ± 0.01
Block	**86.9%** ± 0.04 (2)	**86.9%** ± 0.04	▼51.2% ± 0.00	▼*76.9%* ± 0.01	▼51.4% ± 0.01	▼55.2% ± 0.03	▼55.9% ± 0.03

followed by VARMOG (63.3%). This is the only case of the benchmarks where VARMOG can not identify the number of clusters rightly (it estimates 3 in the majority of cases). Nevertheless, VARMOG is significantly better than the rest of the algorithms.

Although VARMOG was designed to estimate the number of clusters, it shows competitive results against similar algorithms that need to know the number of clusters in advance. Nevertheless, it is still affected by the noise, which can be a consequence of the low-level clustering.

D. Experiments with Real-World data

In the case of the real world datasets, it is harder to have an intuition whether the classes describe manifolds or to evaluate the results via visualization. Nevertheless, it is important to evaluate whether the algorithm can identify patterns in these datasets. Table I (low) shows the results of the algorithms on the four datasets described in Section IV-A.

For **Covtype** every algorithm has problems discriminating the patterns of the data. The results of MOGCLA and SC+N (30.7% and 29.3%, respectively) are the best, followed by VARMOG (27.1%), which is not able to identify the right number of clusters (the algorithm identifies 5 and the dataset has 7 classes). Nevertheless, the results are significantly better than the rest of the algorithms (they obtain results between 19% and 24%). These problems might be produced by the sparsity of the data and the number of dimensions.

For the **Electricity** dataset the results are slightly better. VARMOG and MOGCLA, and SC+N obtain the best results (61.3% and 60%, respectively). Besides, VARMOG successfully identifies the correct number of classes in the data. The rest of the algorithms obtain the same results (52.2%), which are significantly worse, according to the Wilcoxon test, to the results obtained by VARMOG.

Statlib dataset is the most challenging for all the algorithms, it is very likely that this dataset is not suitable for clustering problems. In this case, VARMOG is the worst algorithm of the whole set. It fails to identify the number of classes (it identifies 12 out of 30) and its accuracy is around a 5% while the rest of the algorithms have accuracy levels between 8 and 10%. This is probably a consequence of the high number of classes.

The final dataset, **Block** shows opposite results to the previous one. In this case both, VARMOG and MOGCLA are able to identify a good discrimination of the clusters (86.9%) and VARMOG can successfully identify the number of clusters. For the rest of the algorithms, only SC+N is able to identify patterns in the data (76.9% of accuracy), the rest drop close to 50%, which is the worst possible result for a two classes dataset.

VARMOG is competitive for the real-world datasets and can discriminate the number of clusters when the divisions are clearer. It has also shown that in datasets with high number of classes it has problems discriminating them.

V. Conclusions and Future Work

This work proposes VARMOG, an extension of MOGCLA algorithm to select the number of clusters automatically. VARMOG uses a manifold identification approach based on two levels. The low level summarizes the original space into a set of centroids, by applying the Map-Reduce version of K-means. The high-level identifies the manifold's shape by joining these centroids via a co-evolutionary approach. This co-evolution is also able to decide the proper number of clusters based on the Pareto Front of the search process. It obtains good results for large datasets. It is also competitive against 6 state of the art clustering algorithms. VARMOG has similar (or better) clustering results than the results obtained by using SC and Nyström and Online Clustering algorithms.

The future work will focus on several improvements of VARMOG. On the one hand, the effects of noisy information should be deeply analysed, whereas, on the other hand, I want to extend the algorithm to work directly on the data stream, giving it the ability to process the data online. I will study other fitness functions that can improve VARMOG convergence and the clusters quality. Finally, I want to extend this work to different paradigms, specially to Ant Colony Optimization, to improve the performance of known ACO clustering algorithms as MACOC [23], [24].

References

[1] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology*. ACM, 2011, pp. 530–533.

[2] G. Bello, H. Menendez, and D. Camacho, "Using the clustering coefficient to guide a genetic-based communities finding algorithm," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2011, pp. 160–169.

[3] G. Bello, H. Menendez, S. Okazaki, and D. Camacho, "Extracting collective trends from twitter using social-based data mining," in *International Conference on Computational Collective Intelligence*. Springer, 2013, pp. 622–630.

[4] H. D. Menendez, "A tutorial on manifold clustering using genetic algorithms," in *Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on*. IEEE, 2015, pp. 1–6.

[5] H. D. Menéndez, D. F. Barrero, and D. Camacho, "A genetic graph-based aprroach for partitional clustering," *International Journal of Neural Systems*, vol. 24, no. 03, p. 1430008, 2014.

[6] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007. [Online]. Available: http://www.kyb.mpg.de/publications/attachments/Luxburg06_TR_%5B0%5D.pdf

[7] H. D. Menéndez, D. F. Barrero, and D. Camacho, "A multi-objective genetic graph-based clustering algorithm with memory optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 3174–3181.

[8] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nystrom method," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 2, pp. 214–225, 2004.

[9] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in *Foundations of computer science, 2000. proceedings. 41st annual symposium on*. IEEE, 2000, pp. 359–366.

[10] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *The Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[11] T. Murata and H. Ishibuchi, "Moga: Multi-objective genetic algorithms," in *Evolutionary Computation, 1995., IEEE International Conference on*, vol. 1. IEEE, 1995, p. 289.

[12] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Cloud Computing*. Springer, 2009, pp. 674–679.

[13] H. D. Menendez and D. Camacho, "Mogcla: A multi-objective genetic clustering algorithm for large data analysis." in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1437–1438.

[14] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[15] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The clustree: indexing micro-clusters for anytime stream mining," *Knowledge and information systems*, vol. 29, no. 2, pp. 249–272, 2011.

[16] V. Guillemin and A. Pollack, *Differential topology*. American Mathematical Soc., 2010, vol. 370.

[17] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: applications and algorithms," *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.

[18] E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[19] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho, "A survey of evolutionary algorithms for clustering," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 2, pp. 133 –155, march 2009.

[20] O. Beckonert, M. E Bollard, T. Ebbels, H. C. Keun, H. Antti, E. Holmes, J. C. Lindon, and J. K. Nicholson, "Nmr-based metabonomic toxicity classification: hierarchical cluster analysis and k-nearest-neighbour approaches," *Analytica Chimica Acta*, vol. 490, no. 1, pp. 3–15, 2003.

[21] M. D. Buhmann, "Radial basis functions," *Acta Numerica 2000*, vol. 9, pp. 1–38, 2000.

[22] E. A. Gehan, "A generalized wilcoxon test for comparing arbitrarily singly-censored samples," *Biometrika*, vol. 52, no. 1-2, pp. 203–223, 1965.

[23] H. D. Menéndez, F. E. Otero, and D. Camacho, "Medoid-based clustering using ant colony optimization," *Swarm Intelligence*, vol. 10, no. 2, pp. 123–145, 2016.

[24] ——, "Macoc: a medoid-based aco clustering algorithm," in *Swarm Intelligence*. Springer, 2014, pp. 122–133.

Acknowledgment

This work has been supported by the projects: InfoTestSS EP/P006116/1 and GGGP EP/M025853/1 from EPSRC. I also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.