UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

MÁSTER EN INGENIERÍA EN INFORMÁTICA Y DE TELECOMUNICACIONES

TRABAJO FIN DE MÁSTER

A GENETIC APPROACH TO THE GRAPH AND SPECTRAL CLUSTERING PROBLEM

Autor: D. Héctor Menéndez Benito
Tutor: Prof. D. David Camacho Fernández

June 6, 2012

Contents

List of Figures

Abstract

The study of the clustering techniques has become an interesting machine learning and data mining fields. These techniques have grown since the introduction of new algorithms based on the continuity of the data, such as Spectral Clustering. This technique uses graph theory to generate the clusters through the spectrum of the graph created by a similarity function between the elements of the analysis. Based on this idea, this study uses genetic algorithms to select the groups using the same similarity graph made by the Spectral Clustering method. The goal is to create a new algorithm which improves the robustness of the Spectral Clustering algorithm and reduce the dependency of the similarity metric parameters. To achieve these goals, different fitness functions inspired in Complex Network analysis (which also uses graph representations) and Graph Theory have been designed and tested. Finally, different synthetic and real datasets have been applied to test the Genetic Graph-based Clustering (GGC) algorithm and the different fitness which could be applied and the results have been compared with classical clustering methods.

1 Introduction

The unsupervised learning methods are mainly based on clustering techniques [14]. These techniques were designed to find hidden information or patterns in a dataset grouping the data with similar properties in clusters. The different methods are divided in three main categories[38]:

- Partitional [52]: consists in a disjoint division of the data where each element belongs only to a single cluster.
- Overlapping [9] (or non-exclusive): allows each element to belong to multiple clusters.
- Hierarchical [47]: nests the clusters formed through a partitional clustering method creating bigger partitions, grouping the clusters by hierarchical levels.

This work is focused on the first category: partitional clustering. If the dataset is defined as $\{x_1, \ldots, x_n\}$ where x_i is a piece of information, then the partitional clustering algorithm assigns each x_i only to one cluster C_j. The set of all clusters is the partition, represented by $C = \{C_1, \ldots, C_k\}$. Partitional clustering have three main approximations[14]:

- Parametric or Model-based clustering [14]: consists on an estimator based on a mixture of probabilities whose parameters are estimated. This estimation fixes the model to the dataset.
- Non-Parametric clustering[74]: there is not an initial probability model or estimator.
- Semiparametric method [14]: a combination of parametric and non-parametric methods.

This work is based on Non-Parametric Partitional Clustering. It is based on the Spectral Clustering algorithm introduced by Ng et al. [61] in 2001. This algorithm is divided in three main steps:

1. A similarity function is applied over an original dataset to construct a Similarity Graph amongst the information.
2. The Spectrum of the Similarity Graph is extracted to study the eigenvectors generated.
3. K-means (or other partitional clustering technique) is applied to the matrix formed by the k-first eigenvectors to discriminate the information and assign the final clusters.

The main problem related to the Spectral Graph is its sensibility to the definition of the similarity function. It produces several problems when there is noisy information as Chang and Yeung exposed in [13]. Some solutions to this problem were based on the improvements of the parameters selection for the similarity function [13]. Other solutions are focused on the selection of the partitional clustering algorithm for the third step of SC [76]. This work develops a new algorithm based on Genetic Algorithms (GA) to improve the robustness of the clusters selection taking the Similarity Graph as a starting point. It looks for new solutions that could aid to manage this problem.

Genetic algorithms have been traditionally used in optimization problems. The complexity of the algorithm depends on the codification and the operations that are used to reproduce, cross, mutate and select the different individuals (chromosomes) of the population [17, 70]. This kind of algorithms applies a fitness function which guides the search to find the best individual of the population.

The fitness function is used as a metric to measure the quality of the clusters. The fitness initially proposed in this work is based on Complex System Analysis and Graph Theory Techniques [21, 58, 78]. Given a network which is represented as a graph, these techniques analyses the different properties of the network through various measures. The principal measures of these systems are the Clustering Coefficient [21] and the Average Distance of the elements. There are several variations of them, such as the Weighted Clustering Coefficient [7], that are studied in this work.

On the one hand, one of the main problems of this kind of fitness function is that it does not consider the continuity of the dataset which is important in this kind of clustering approach. The continuity is the form defined by the data, for example, the path defined by a user in a video-game. Therefore, other different fitness functions based on well-known algorithm (such as K-Nearest Neighbours [18] and Minimal Graph Cut [68]) have also been tested and combined to improve the results.

On the other hand, the codification generates an important convergence problem, specially to the clustering problem. Different approximations of genetic codifications to the clustering problem were deeply studied by Hruschka et al. in [38]. These methods are also divided in the type of clustering: parametric or non-parametric. Here, two different codifications are introduced and analysed to compare their features.

This work presents a Genetic Graph-based Clustering (GGC) algorithm which is inspired on the Spectral Clustering algorithm (it parts from the same Similarity Graph) and improves the robustness of the solutions. The algorithm is experimentally compared with Spectral Clustering, K-means [49] and Expectation Maximization (EM) [49] to test its accuracy. The experimental study is also focused in a comparison between the robustness of the Spectral Clustering and GGC algorithms.

The rest of the work is structured as follows: Section 2 shows a description of the state of art; Section 3 presents the Genetic Graph-based Algorithm designed and developed; Section 4 shows the experimental results. Finally, Section 5 gives the conclusions and future lines of work.

2 State of the Art

This section starts with a general introduction of the Data Mining and Machine Learning techniques. After this brief introduction, it presents the data preprocessing and normalization techniques used in this work. Next, it is focused on the clustering methods, specially in the Spectral Clustering algorithm. Once the clustering methods have been introduced, it focusses the attention on how genetic algorithms have been applied to clustering techniques. Finally, the measures of graph theory and complex network are introduced and defined.

2.1 Data Mining

Data Mining is "the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques" [49]. The Data Mining techniques are divided in 5 main steps:

1. **Data Extraction:** The data extraction problem consists on obtain the datasets which will be analysed. There are several public databases, for example, those which are contained in the UCI Machine Learning Repository [27], that have been widely used to test these techniques.

2. **Data Preprocessing and Normalization:** The data preprocessing methods prepare the data to be analysed. There are three main steps [49]: avoid misclassification, dimensionality reduce (through projections or feature selection techniques) and range normalization.

3. **Model Generation:** This is the most important part of the data analysis. The model is created to find the patterns in the data. It is usual to use Machine Learning or other statistical techniques to generate the model [49].

4. **Model Validation:** Depending on the type of model, the validation process is different. This process gives the confidence of the model. It is usual to use validation with classifiers [49], however, the clustering validation is almost a blind process [80] (which is a consequence of the clustering nature).

5. **Model Application:** The goal of the model is to be applied, for example, to predict the behaviour of new inputs. For example, our previous work was based on the application of clustering models to

predict performance in team games [41], and in the FIFA World Cup Championship [54].

This work is focused on the Model Generation methods. These methods are based in Statistical Inference and Machine Learning techniques. Machine Learning approaches are based on a "machine" which receives a sequence of inputs, called data, and looks for patterns which can be used to predict or explain the behaviour of new inputs. Some applications of these models can be found in [80]: business, biology, music, human behaviour, games... These techniques can be divided in four categories[11]:

- *Supervised Learning:* A sequence of desired outputs is also given with the inputs. The goal of the machine is to learn to produce the correct output given a new input. The output could be a class label (classification) or a real number (regression). Some examples of classical classification methods are [49]: Decision Trees, Support Vector Machines and Neural Networks.

- *Reinforcement Learning:* The machine produces a set of actions which affects the effect of an environment. These effects generates a reward (or punishment) which must be maximized in the future though the machine decisions.

- *Game Theory Learning:* It is a generalization of reinforcement learning. In this case, the environment can contain other machines with the same characteristics.

- *Unsupervised Learning:* The machine simple receives the input. Its goal is to generate the labels of the input set. This work is focused on generate a clustering algorithm for unsupervised learning though genetics algorithms. the main clustering techniques considered in this work will be described in Sections 2.3 and 2.4.

As it was explained in the Introduction, the most common unsupervised data mining method is clustering. Another good example of unsupervised learning is dimensionality reduction[11] which is the process of reducing the number of random variables under consideration in a dataset analysis. These techniques are also known as feature selection methods and are presented in the following subsection.

2.2 Data Preprocessing and Normalization

Data Mining techniques need an intensive phase of data preprocessing. Initially the information must be analysed and stored in some kind of database system, cleaned and separated. This preprocessing phase is used to avoid outliers, missclassifications and missing data. Methods such as histogram and statistical correlation are used to clean the dataset and reduce the number of variables [49]. Projections are also usual in dimension reduction, however, projection methods [22] such as PCA (Principal Component Analysis) or LDA (Lineal Discriminant Analysis) do not offer a complete perspective of the problem we are dealing with. These methods create new variables which are estimated from principal components or lineal projection trying to separate the data and reduce its dimension. Usually, these techniques lose the original information of the features which is unrecoverable once it is projected. It produces a reduction of the human interpretation of Data Mining techniques applied and, sometimes, it is preferable to avoid them.

There are several techniques which reduce the feature sets to avoid projections. These methods apply a guided search among the different attributes looking for the most useful variables for the analysis. These methods are usually known as feature selection methods [45]. Curiel et al. [19] apply genetic algorithms to simplify prognosis of endocarditis using a codification where each individual of the population is based on a set of features. Blum and Langley [10] show some examples of relevant features selections in different datasets and applied them to different machine learning techniques. They define different degrees of relevant features such as strong or weak relevant features. They also study some methodologies such as heuristic search, filters and wrapper approaches which are automatic feature selection methods usually validated by classification techniques. Some of these techniques usually introduced over-fitting to the model and are computationally expensive. Roth and Lange [66] apply these techniques to the clustering problem. Our previous work showed a new feature and simple selection methodology whose goal is to make an intensive reduction of the attributes dimension oriented to clustering analysis. It applies clustering methods to validate the chosen feature set [55].

Finally, the last step is related to normalization. It allows to compare data features with different kind of range of values. Z-Score [12] and Min-Max [35] normalization methods are commonly used for preprocessing the data. Both normalization algorithms takes the attribute records and they find a standard range for them. Min-Max has a fixed range, [0,1] (it is sensitive to outliers), while Z-Score depends on the mean and the standard deviation (it approximates the distribution to a normal distribution, it is usually used to avoid outliers). These algorithms obtain the normalized values from data using the following equations:

- Min-max: It computes maximum and minimum values of the attributes applying:
$$x' = \frac{x - min(X)}{max(X) - min(X)}$$
- Z-Score: It computes mean and standard deviation of the values applying:
$$x' = \frac{x - mean(X)}{SD(X)}$$

Once the data is ready for the analysis, the model generation phase begins. This work is focused on unsupervised learning techniques for model generation, specially clustering techniques, that are presented in the following section.

2.3 Clustering Techniques

Clustering techniques are frequently used in data mining and machine learning methods. A popular clustering technique is K-means. Given a fixed number of clusters, K-means tries to find a division of the dataset [52] based on a set of common features given by distances or metrics that are used to determine how the cluster should be defined. Other approximations, such as Expectation-Maximization (EM) [23], are applied when the number of clusters is unknown. EM is an iterative optimization method that estimates some unknown parameters computing probabilities of cluster membership based on one or more probability distributions; its goal is to maximize the overall probability or likelihood of the data being in the final clusters [59].

Since these techniques fixed the number of clusters a priori, there are validation techniques such as cross-validation [44] which are used to improve the number of clusters selection (through metrics such as the Minimum Sum-of-Squares [56]).

Other research lines have tried to improve these algorithms. For example, some *online* methods have been developed to avoid the K-means convergence problem to local solutions which depend on the initial values [6]. These methods create the clusters adding a new data instance at each step and modifying the cluster structure with this new information. Some other improvements of K-means algorithm are related to deal the different kind of data representation, for example, mixed numerical data [3] and categorical data [67] . There are also some studies comparing methods with different datasets, for example, Wang et al. [77] compare self-organizing maps, hierarchical clustering and competitive learning where establishing molecular data models of large size sets. Other approaches related to genetic algorithms, and directly related to this work, will be described in subsection 2.5.

Machine learning techniques have also been improved through the k-means algorithm, for example, reinforcement learning algorithms[5, 31]; or using topological features of the data set [30, 31] which can also be helpful for data visualization.

The following subsections introduces two of the most classical clustering algorithms: Expectation-Maximization and K-means. The following section presents the Spectral Clustering algorithm which inspired this work.

2.3.1 K-means

K-means [51] is a popular and well known partitional clustering algorithm. It is a straightforward clustering guided method (usually by a heuristic or directly by a human) to classify data in a predefined number of clusters. As it was explained above, given a fixed number of clusters (k), K-means tries to find a division of the dataset [52] based on a set of common features given by distances or metrics that are used to determine what elements belong to each cluster. K-means has also been improved though different techniques, like genetic algorithms [8]. Algorithm 1 shows the pseudo-code for K-means algorithm [49].

2.3.2 Expectation Maximization

Expectation Maximization (EM) [23] is used when the number of clusters is unknown. Initially, it takes the likelihood and tries to maximize it. The process consists on apply the two following steps iteratively until it converges:

- **Expectation step**: Fix a model (θ) and estimate missing labels ($\mathbf{y}$).
- **Maximization step**: Fix missing labels ($\mathbf{y}$)(or a distribution over missing labels) and find the model (θ) which maximizes the likelihood function ($L(\theta)$) of the data.

The likelihood function is defined by:

$$L(\theta) = p(\mathbf{X}, \mathbf{y}|\theta) = \prod_i p(\mathbf{x}_i, y_i|\theta)$$

Algorithm 1 Pseudo-code for K-means algorithm

```
Input: A dataset of $n$ elements $X = \{x_1, \ldots, x_n\}$ and a fix number of clusters $k$.
Output: A set of clusters $C = \{C_1, \ldots, C_k\}$ which partitionate $X$
1: Assign $k$ records to be the initial cluster centroids. We define the set of centroids as $Y = \{y_1, \ldots, y_k\}$ and $Y' = \emptyset$
2: while $Y \neq Y'$ do
3:     Set all $C_j = \emptyset$.
4:     $Y' \leftarrow Y$.
5:     for all $x_i \in X$ do
6:         Calculate the minimal distance centroid to $x_i$. Let $y_j$ be the minimal distance centroid to $x_i$.
7:         Introduce $x_i$ in $C_j$.
8:     end for
9:     Calculate the centroids of $C$ and set $y_i \leftarrow centroid(C_i)$.
10: end while
11: return $C$
```

K-means initially sets the centroids (line 1) and the elements are added to the cluster whose centroid is closer to them (lines 5 to 7). After, it calculates the new centroids position of the clusters (line 9) and again adds the elements to the closer clusters (lines 5 to 7). It continues until the centroid position converge to a fixed point (line 2).

Where θ is a model defining how each instance $\mathbf{x}_i$ is assigned to a label y_j. This algorithm begins with the definition of an initial model $\theta^{(0)}$ and constructs a sequence $\theta^{(0)}$,$\theta^{(1)}$,...,$\theta^{(t)}$,... of models with increasing value of likelihood. To simplify the calculation process, the logarithm of the likelihood function is used:

$$l(\theta) = \log L(\theta) = \sum_i \log p(\mathbf{x}_i, y_i|\theta)$$

EM also used an auxiliary function, which depends on the model and the distribution of missing labels, defined by:

$$Q(\theta|\theta^{(m)}) = \sum_i Q_i(\theta|\theta^{(m)}) = \sum_i E_{y_i|\mathbf{x}_i,\theta^{(m)}}[\log p(\mathbf{x}_i, y_i|\theta)]$$

$$Q(\theta|\theta^{(m)}) = \sum_i \sum_{j=1}^{k} P(y_i = j|\mathbf{x}_i, \theta^{(m)}) \log p(\mathbf{x}_i, y_i|\theta) \qquad (1)$$

This function is used to increment the likelihood of the estimator θ which is taken as a maximum of this function. Algorithm 2 shows the pseudo-code for EM algorithm.

2.4 Spectral Clustering

Spectral clustering methods are based on a straightforward interpretation of weighted undirected graphs as can be seen in [4, 57, 61, 74]. The Spectral Clustering approach is based on a Similarity Graph which can be formulated in three different ways (all of them equivalent [74]) of graphs:

1. **The ϵ-neighbourhood graph**: all the components whose pairwise distance is smaller than ϵ are connected.
2. **The k-nearest neighbour graphs**: the vertex v_i is connected with vertex v_j if v_j is among the k-nearest neighbours of v_i.
3. **The fully connected graph**: all points with positive similarity are connected with each other.

The main problem is how to compute the eigenvector and the eigenvalues of the Laplacian matrix of this Similarity Graph. For example, when large datasets are analysed, the Similarity Graph of the Spectral Clustering algorithm takes too much memory, it makes difficult the eigenvalues and eigenvectors computation. Some works are focused on this problem: von Luxburg

Algorithm 2 Expectation Maximization Pseudo-code [34]

```
Input: A dataset of n elements $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. A convergence error value $\delta$.
Output: A set of clusters $C = \{C_1, \ldots, C_k\}$ which partitionate $X$
 1: Fix a number of cluster $k$ {this value is estimated applying cross-validation and repeating this algorithm with different values of k}
 2: Choose the initial model $\theta^{(0)}$.
 3: Compute the initial log-likelihood $l^{(0)}(\theta^{(0)})$.
 4: repeat
 5:     E-step: Calculate $\{\gamma_{ij}^{(m)}\}$ where $\gamma_{ij} = P(y_i = j|\mathbf{x}_i, \theta^{(m)})$. {For the $m$ iteration}
 6:     M-step: Calculate $\theta^{m+1} = \text{argmax}_\theta \left(Q(\theta|\theta^{(m)})\right)$, where $Q(\theta|\theta^{(m)}) = \sum_i \sum_{j=1}^k \gamma_{ij}^{(m)} \log p(\mathbf{x}_i, y_i|\theta)$. {Extracted from equation 1}
 7:     Convergence check: Calculate $l^{(m+1)}(\theta^{(m+1)})$.
 8: until $|l^{(m+1)} - l^{(m)}| < \delta$
 9: Put $\mathbf{x}_i$ in $C_j$ if $\gamma_{ij} = max(\{\gamma_{iq}\}_{q=1}^k)$
10: return $C$
```

The m-th iteration of the E-step (line 5) produces a guess of the $n \times k$ membership-probabilities of the elements to the clusters $\{\gamma_{ij}\} = P(y_i = j|\mathbf{x}_i, \theta^{(m)})$, where γ_{ij} is the current guess of the probability that sample $\mathbf{x}_i$ came from the j-th cluster. The M-step (line 6) gives a closed-form solution to the new estimates of the estimator θ. It converges if the increment of the likelihood is lower than a value (δ) given.

et al. [74] present the problem, Ng et al.[61] apply an approximation to a specific case, and Nadler et al.[57] apply operators to get better results. The classical algorithms can be found in [74].

The theoretical analysis of the observed good behaviour of SC is justified using the perturbation theory [74, 57], random walks and graph cut [74]. The perturbation theory explains, through the eigengap, the behaviour of Spectral Clustering.

Some of the main problems of Spectral Clustering are related to the consistency of the two classical methods used in the analysis: normalized and un-normalized spectral clustering. A deep analysis about the theoretical effectiveness of normalized clustering over un-normalized can be found in (von Luxburg, 2008)[75].

Part of the present work is inspired by Spectral Clustering because we use a clustering technique which analyse a Similarity Graph. Nevertheless, in our case we are using different methods such as Genetic Algorithms, Graph Theory and Complex Networks analysis to find the clusters, instead of the Laplacian matrix extracted from the Similarity Graph.

2.4.1 The Spectral Clustering Algorithm

Spectral Clustering methods were introduced by Ng et al. in [61]. These methods apply the knowledge extracted from graph spectral theory to clustering techniques. These algorithms are divided in three main steps:

1. The algorithm constructs a graph using the data instance as nodes and applying a similarity measure to define the edges weights (see algorithm 3 line 1). The different types of graphs are explained above. In this work a fully connected graph is used. The measure used in this work is the Radial Basis Function (RBF) Kernel (which is the most usual approach taken in the literature) defined by:

$$s(x_i, x_j) = e^{-\sigma||x_i - x_j||^2} \tag{2}$$

   where σ is used to control the width of the neighbourhood.

2. It studies the graph spectrum calculating the Laplacian Matrix associated to the graph (see algorithm 3 lines 2 and 3) . There are different definitions of the Laplacian Matrix. These definitions achieved different results when they are applied to the Spectral Clustering algorithm. They are used to categorize the Spectral Clustering techniques as follows [74]:

   - **Unnormalized Spectral Clustering** It defines the Laplacian matrix as:
   $$L = D - W$$
   - **Normalized Spectral Clustering** It defines the Laplacian matrix as:
   $$L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$$
   - **Normalized Spectral Clustering (related to Random Walks)** It defines the Laplacian matrix as:
   $$L_{rw} = D^{-1}L = I - D^{-1}W$$

In these formulas I is the identity matrix, D represents the diagonal matrix whose (i,i)-element is the sum of the similarity matrix ith row and W represents the Similarity Graph (see algorithm 3 line 2). Once the Laplacian is calculated (in algorithm 3 the Normalized Spectral Clustering algorithm is used, however, in this case, to simplify, the eigenvalues which are calculated are $1-\lambda_i$ instead of λ_i, the eigenvectors do not change), its eigenvectors are extracted (see lines 4 and 5 of algorithm 3).

3. The eigenvectors of the Laplacian Matrix are considered as points and a clustering algorithm, such as k-means, is applied over them to define the clusters (see algorithm 3 lines 7 and 8).

The Spectral Clustering algorithm which is used in this work is the Normalized Spectral Clustering Algorithm introduced by Ng [61] (see algorithm 3)

Algorithm 3 Normalized Spectral Clustering according to Ng et al. (2001)[61]

```
Input: A dataset of n elements $X = \{x_1, \dots, x_n\}$ and a fix number of clusters $k$.
Output: A set of clusters $C = \{C_1, \dots, C_k\}$ which partitionate $X$
1: Form the affinity matrix $W \in R^{n \times n}$ defined by $W_{ij} = e^{-||x_i - x_j||^2/2\sigma^2}$ if $i \neq j$, and $W_{ii} = 0$.
2: Define $D$ to be the diagonal matrix whose $(i,i)$-element is the sum of the $i$-th row of $W$.
3: Construct the matrix $L = D^{-1/2} W D^{-1/2}$.
4: Find $v_1, \dots, v_k$, the $k$ largest eigenvectors of $L$ (chosen to be orthogonal to each other in the case of repeated eigenvalues) and form the matrix $V = [v_1 v_2 \dots v_k] \in R^{n \times k}$ by stacking the eigenvectors in columns.
5: Form the matrix $Y$ from $V$ by renormalizing each row of $V$ to have unit length (i.e. $Y_{ij} = V_{ij}/(\sum_j V_{ij}^2)^{1/2}$).
6: Apply K-means (or any other algorithm) treating each row of $Y$ as a point in $R^k$.
7: Assign the points $x_i$ to cluster $C_j$ if and only if the row $i$ of the matrix $Y$ was assigned to cluster $j$.
8: return $C$
```

2.4.2 Spectral Clustering Example

This section shows the application of the Spectral Clustering Algorithm. In this case, the dataset is a 2-dimensional dataset composed by three clusters generated by 50 instances o three different Gaussian distributions (Figure 1 shows the dataset):

1. The first Gaussian Distribution has a Mean of 0 and a Standard Deviation of 0,3.

2. The first Gaussian Distribution has a Mean of 1 and a Standard Deviation of 0,3.

3. The first Gaussian Distribution has a Mean of 2 and a Standard Deviation of 0,3.

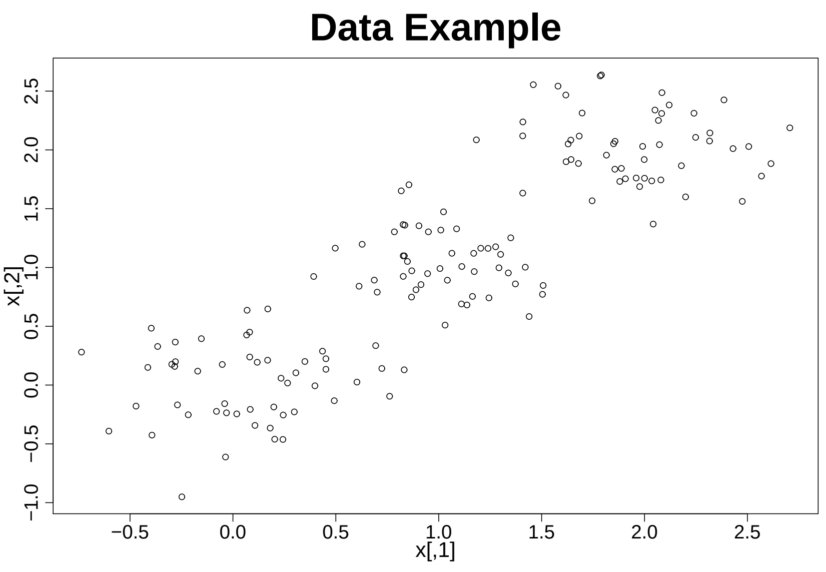


Figure 1: Dataset for the Spectral Clustering application example.

The first step is the generation of the similarity graph. The edge weights of this graph represents the similarity between the points. In this example, the similarity is related to the inverse of the distance between the points (The metric used id the RBF kernel [42], see equation 2 in section 2.4.1).

The similarity graph is represented in figure 2 using a heat-map. The heat-map rows and columns correspond with the adjacency matrix rows and columns of the Similarity Graph. The colours are the similarity values from red (low similarity) to yellow (high similarity). It shows that there are 3 sets whose elements are closer between them.

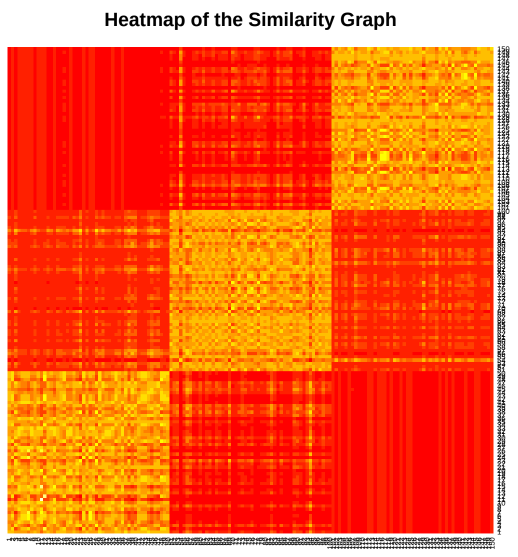
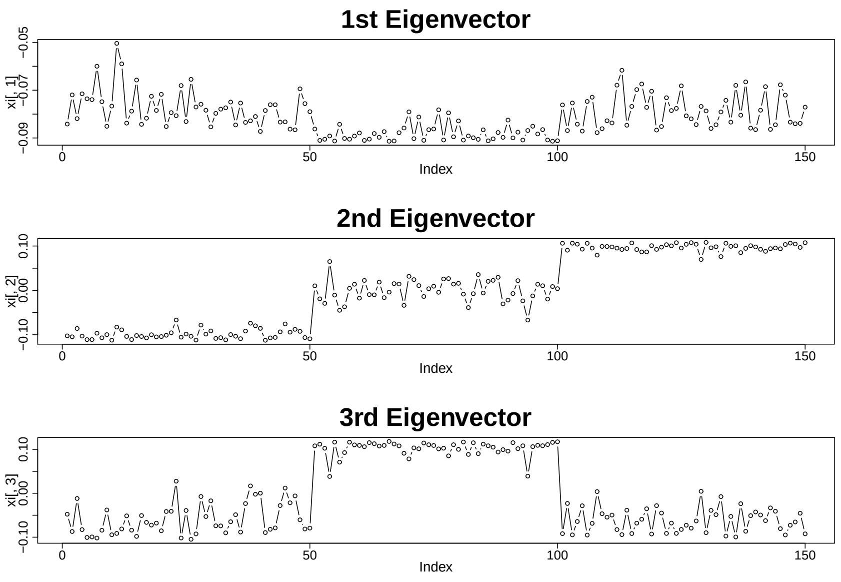


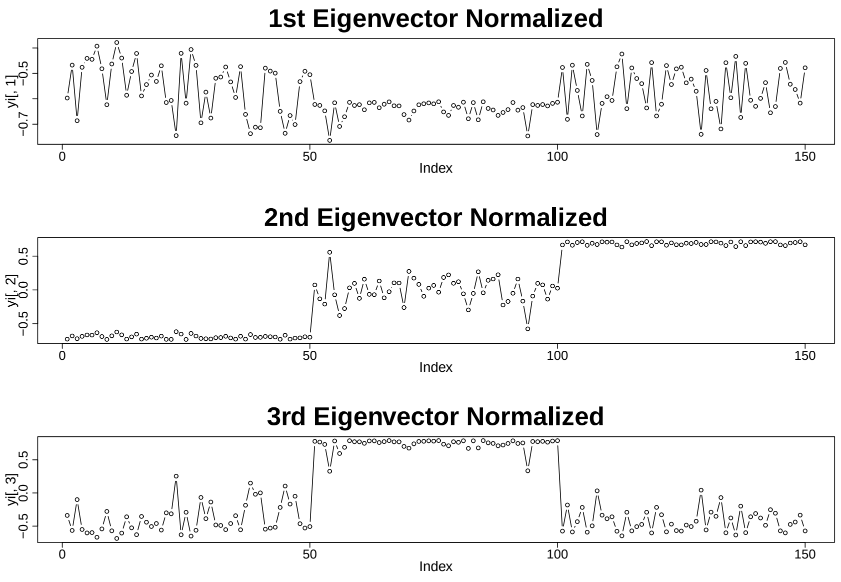
Figure 2: Heat-map of the Similarity Graph generated in the first step of the Spectral Clustering example.

The second step is related to the study of the eigenvector of the Laplacian Matrix of the Similarity Graph. The Laplacian Matrix has a maximum of 150 different eigenvectors. According to the leader eigenvalues (those eigenvalues with the highest values) their associated eigenvectors are chosen. Since this problem has three clusters, the 3 leader eigenvectors are chosen. These eigenvectors are shown in Figure 3. This figure shows the original eigenvectors obtained from the Laplacian Graph (Figure 3(a)) and their normalization (Figure 3(b)). The normalization reduces the distortion of the values.

The last step is the application of a clustering algorithm to the matrix formed by the normalized eigenvectors considering each row of the matrix as a point. Figure 4(a) shows the distribution of the data in the space generated by the chosen eigenvector. In this case, each point could be interpreted as a projection of the original points of Figure 1. The algorithm which is applied is K-means. The results of the application of the algorithm to this data are shown in Figure 4(b). The data in this space is easier to separate than in the original space.



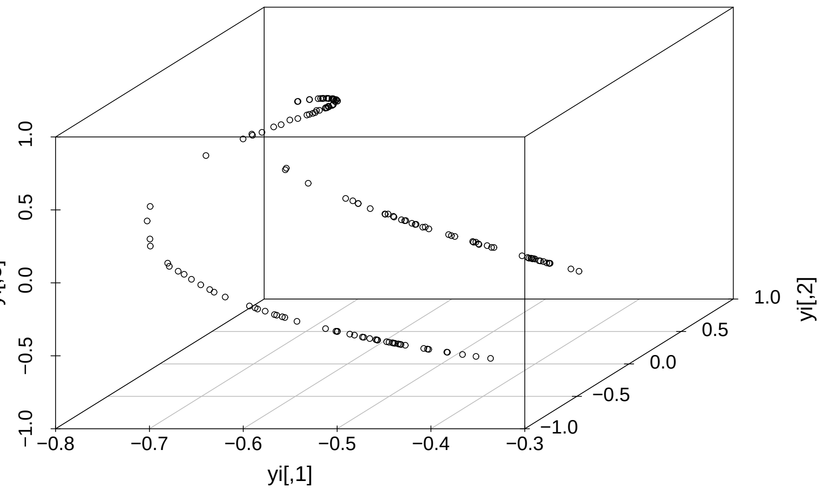
(a) Original Eigenvectors: $V = v_1 v_2 v_3$ (see algorithm 3 line 4)



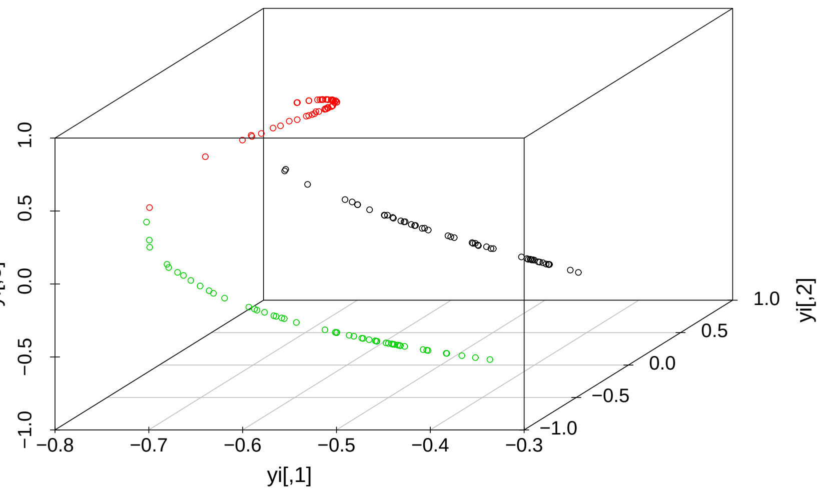
(b) Normalized Eigenvectors: $Y = y_1 y_2 y_3$ (see algorithm 3 line 5)

Figure 3: Eigenvectors and Normalized eigenvectors obtained in the Spectral Clustering example.

Finally, Figure 5 shows the results of the algorithm on the original space.



(a) 3D representation of Y



(b) 3D representation of Y considering the clustering algorithm results

Figure 4: Representation of the points projection over the 3 eigenvectors

2.5 Genetic Algorithms for Clustering

Genetic algorithms have been traditionally used in optimization problems, as was mentioned before. These algorithms have also been used for general data and information extraction [28]. The operators of the genetic algorithms

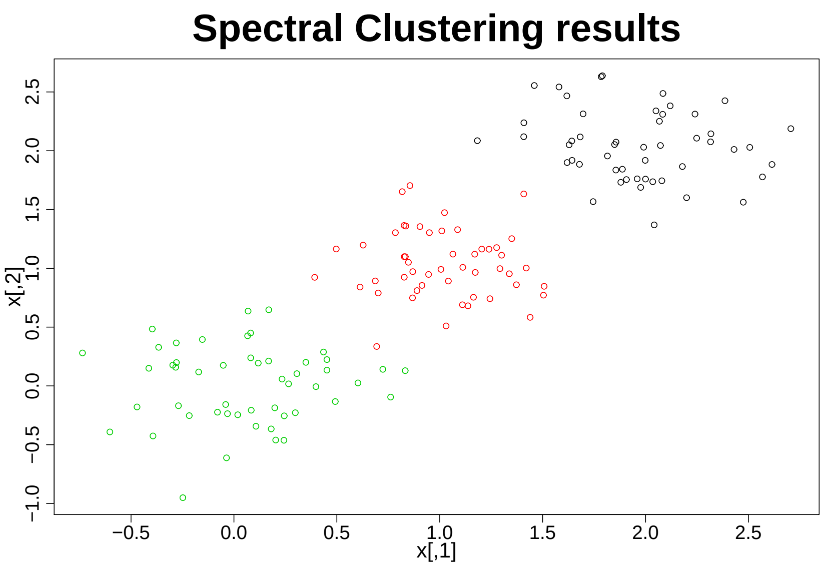


Figure 5: Final results of the Spectral Clustering example

can also be modified. Some examples of these modifications can be found in (Poli and Langdon, 2006)[63] where the algorithm is improved through backward-chaining, creating and evaluating individuals recursively reducing the computation time. Other applications of genetic clustering algorithms can be found in swarm systems [48], software systems [24], file clustering [25] and task optimization [62], amongst others.

The genetic clustering approximation tries to improve the results of the clustering algorithm using different fitness functions to tune up the cluster sets selection. In (Cole, 1998)[16], different approaches of the genetic clustering problem, especially focused in codification and clustering operations, can be found. There is also a deep revision in (Hruschka et al., 2009)[38] which provides a complete up to date state of the art in evolutionary algorithms for clustering.

There are several methods using evolutionary approaches from different perspectives, for example: (Aguilar, 2007)[2] modifies the fitness considering cluster asymmetry, coverage and specific information of the studied case; (Tseng and Yang, 2001)[71] use a compact spherical cluster structure and a heuristic strategy to find the optimal number of clusters; (Maulik and Bandyopadhyay, 2000)[53] use the clustering algorithm for metric optimization trying to improve the cluster centre positions; (Shi et al., 2011)[69] based the search of the genetic clustering algorithm in their Extend Classifier Systems which is a kind of Learning Classifier System, in which a fitness of the classifier is determined by the measure of its prediction's accuracy; (Das and Abraham, 2008)[20] use Differential Evolution, a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality.

Some of those previous methods are based on K-means, for example: (Krishna and Murty, 1999)[46] replace the crossover of the algorithm using K-means as a search operator, and (Wojciech and Kwedlo, 2011)[79] also use differential evolution combined with K-means, where it is used to tune up the individuals obtained from mutation and crossover operators. Finally, other general results of genetic algorithm approaches to clustering can be found in (Adamska, 2005)[1]. There are also other complete studies for multi-objective clustering in (Handl et al., 2004)[36] and for Nearest Neighbour Networks in (Huttenhower et al., 2007)[39].

2.6 Graph Theory and Complex Networks

Graph theory has also proved to be an area of important contribution for research in data analysis, especially in the last years with its application to manifold reconstruction [33] using data distance and graph representation to create a structure which can be considered as an Euclidean space (which is the manifold).

Graph models are useful for diverse types of data representation. They have become especially popular over the last years, being widely applied in the social networks area. Graph models can be naturally used in these domains, where each node or vertex can be used to represent an agent, and each edge is used to represent their interactions. Later, algorithms, methods and graph theory have been used to analyse different aspects of the network, such as: structure, behaviour, stability or even community evolution inside the graph [21, 26, 58, 78].

A complete roadmap to graph clustering can be found in Schaeffer[68] where different clustering methods are described and compared using different kinds of graphs: weighted, directed, undirected. These methods are: cutting, spectral analysis and degree connectivity (an exhaustive analysis of connectivity methods can be found in Hartuv and Shamir[37]), amongst others. This roadmap also provides an overview of computational complexity from a theoretical and experimental point of view of the studied methods.

From previously described graph clustering techniques, a recent and really powerful ones are those based on Spectral Clustering which were previously introduced. In this section, several definitions of Graph Theory and an introduction of the Complex Network approach and the metric extracted from this field are presented.

2.6.1 Basic Definitions from Graph Theory

Defining and selecting an appropriate fitness function is one of the most critical issues in any evolutionary algorithm. Our approach uses concepts and metrics extracted from graph theory. For this reason, and before describing it, some of those basic concepts are briefly introduced.

Definition 2.1 (Graph)**.** A graph $G = (V, E)$ is a set of vertices or nodes V denoted by $\{v_1, \ldots, v_n\}$ and a set of edges E where each edge is denoted by e_{ij} if there is a connection between the vertices v_i and v_j.

Graphs can be directed or undirected. If all edges satisfy the equality $\forall i, j$, $e_{ij} = e_{ji}$, the graph is said to be undirected.

In this work we will represent the graph through its adjacency matrix (the most usual approach) which can be defined as:

Definition 2.2 (Adjacency Matrix)**.** An adjacency matrix of G, A_G, is a square $n \times n$ matrix where each coefficient satisfies:

$$(a_{ij}) = \begin{cases} 1, & \text{if } e_{ij} \in E \\ 0, & \text{otherwise} \end{cases}$$

When it is necessary to work with weighted in the edges, a new kind of graph needs to be defined:

Definition 2.3 (Weighted Graph)**.** G is a weighted graph if there is a function $w : E \to R$ which assigns a real value to each edge.

Any algorithm that works with the vertices of a graph needs to analyse each node neighbours. The neighbourhood of a node is defined as follows:

Definition 2.4 (Neighbourhood)**.** If the edge $e_{ij} \in E$ and $e_{ji} \in E$ we say that v_j is a neighbour of v_i. The neighbourhood of v_i Γ_{v_i} is defined as $\Gamma_{v_i} = \{v_j \mid e_{ij} \in E \text{ and } e_{ji} \in E\}$. Then, the number of neighbours of a vertex v_i is $k_i = |\Gamma_{v_i}|$

Once the most general and simple concepts from graph theory are defined, we can proceed with the definition of some basic measures related to any node in a graph; the Clustering Coefficient (CC) and the Weighted Clustering Coefficient (WCC).

Definition 2.5 (Local Clustering Coefficient (CC) [21])**.** Let $G = (V, E)$ be a graph where E is the set of edges and V the set of vertices and A its adjacency matrix with elements a_{ij}. Let Γ_{v_i} be the neighbourhood of the vertex v_i. If k_i is considered as the number of neighbours of a vertex, we can define the clustering coefficient ***(CC)*** of a vertex as follows:

$$C_i = \frac{1}{k_i(k_i-1)} \sum_{j,h} a_{jh} a_{ij} a_{ih} a_{ji} a_{hi}$$

The Local CC measure provides values ranging from 1 to 0. Where 0 means that the node and its neighbours do not have clustering features, so they do not share connections between them. Therefore, value 1 means that they are completely connected. This definition of CC can be extended to weighted graphs as follows:

Definition 2.6 (Local Weighted CC [7])**.** Following the same assumption of Local Clustering Coefficient definition, let W be the weight matrix with coefficients w_{ij} and A be the adjacency matrix with coefficients a_{ij}, if we define:

$$S_i = \sum_{j=1}^{|V|} a_{ij} a_{ji} w_{ij}$$

Then, the Local Weighted Clustering Coefficient can be defined as:

$$C_i^w = \frac{1}{S_i(k_i-1)} \sum_{j,h} \frac{(w_{ij} + w_{ih})}{2} a_{jh} a_{ij} a_{ih} a_{ji} a_{hi}$$

For this new definition, we are considering the connections between the neighbours of a particular node, but now we add information about the weights related to the original node. This new measure calculates the distribution of the weights of the node that we are analysing, and shows how good the connections of that cluster are. The following theorem proves that the weighted CC has the same value than the CC when all the weights are set to the same value:

Theorem 2.1. *Let G be a graph, A its adjacency matrix and W its weight matrix. If we set $w_{ij} = \omega\ \forall i, j$, them $C_i = C_i^w$.*

Proof. Following the definition of C_i^w we have:

$$C_i^w = \frac{1}{S_i(k_i - 1)} \sum_{j,h} \frac{2\omega}{2} a_{jh} a_{ij} a_{ih} a_{ji} a_{hi}$$

Where $S_i = \sum_{j=1}^{|V|} a_{ij} a_{ji} \omega$. Replacing S_i, we have:

$$C_i^w = \frac{1}{\sum_{j=1}^{|V|} a_{ij} a_{ji} \omega (k_i - 1)} \sum_{j,h} \omega a_{jh} a_{ij} a_{ih} a_{ji} a_{hi}$$

$$C_i^w = \frac{1}{\sum_{j=1}^{|V|} a_{ij} a_{ji} (k_i - 1)} \sum_{j,h} a_{jh} a_{ij} a_{ih} a_{ji} a_{hi}$$

We also know that following the neighbour definition and the adjacency matrix definition: $k_i = \sum_{j=1}^{|V|} a_{ij} a_{ji} = |\Gamma_{v_i}| = |\{v_j \mid e_{ij} \in E \ and \ e_{ji} \in E\}|$ And finally:

$$C_i^w = \frac{1}{k_i(k_i - 1)} \sum_{j,h} a_{jh} a_{ij} a_{ih} a_{ji} a_{hi} = C_i$$

Which proves theorem 1. □

As a corollary to this theorem, if $C_i^w = 1 \Rightarrow C_i = 1$.

Finally, if we want to study a general graph, we should study its Global CC:

Definition 2.7 (Global CC [21, 7])**.** The clustering coefficient of a graph can be defined as:

$$C = \frac{1}{|V|} \sum_{i=0}^{|V|} C_i$$

Where $|V|$ is the number of vertices.

The Global Weighted Clustering Coefficient is:

$$C^w = \frac{1}{|V|} \sum_{i=0}^{|V|} C_i^w$$

The main difference between Local CC, Local Weighted CC and Global CC is that, the first one can be used to represent how connected is a node locally in a graph, the second one is used to calculate the density of these connections using the edge weights, and the last one provides us with global information about of the connectivity in a graph. In real complex problems only the two initial measures can be used, whereas the third one is usually estimated [72].

2.6.2 Complex Networks Analysis

In network analysis, is common to use a graph representation, especially for the social network approach where users are connected by affinities or behaviours. This approximation has been studied in some of the small world networks based on two main variables: the average distance between elements and the clustering coefficient of the graph [21, 58, 78].

The present work is closed to the network approach because our algorithm looks for sub-graphs in a graph whose elements share similar features. In an initial study of the problem [8], an evolutionary approach was adopted based on the K-means algorithm applied to community finding approach (which is also a clustering problem applied to a graph representation).

Other similar approximations related to the finding-community problem can be found in Reichardt and Bornholdt[65] where different statistical mechanics for community detection are used. Pons and Latapy[64] use random walks to compute the communities. However, we decided to use genetic algorithms because we are interested in optimization methods for tuning up the definition of our clusters, allowing to adapt the size and membership of these clusters using metrics and features selected from graph characteristics.

Finally, another work based on metrics used to measure the quality of the communities can be found in (Newman and Girvan, 2004)[60], and metrics that can be used to find the structure of a community in very large networks in (Clauset et al., 2004)[15]. Genetic algorithms have also been applied to find communities or clusters through agglomerative genetic algorithms [50] and multi-objective evolutionary algorithms [43] amongst others.

3 The Genetic Graph-based Clustering (GGC) Algorithm

This section introduces the Genetic Graph-based Clustering algorithm, that is mainly based on Genetic Algorithms (GA) and Graph Theory.

In this approach it is necessary to feed the algorithm with an initial set of individuals (clusters) like in Spectral Clustering or K-means algorithms. Our technique searches the best sub-graphs which might define a clear partition of the original graph. It is generated by a similarity function such as in the Spectral Clustering algorithm. The population is a set of possible solutions (partitions) which evolves until the best solution is found or a maximum number of generations is reached. The fitness function is a metric which is used to measure how close are the individuals to be the best solution. The algorithm will maximize this metric.

3.1 The GGC Codifications

The genetic algorithm has been constructed using two different codifications. The first codification is a simple vector codification while the second one is based on set theory. These two codifications have been selected to compare their computational effort in the experimental phase and choose the fastest for the final experiments (the comparison is made in Section 4.1.1). They provide different representations for the same valid individuals (this makes them equivalent).

3.1.1 Vector-based Codification

This codification is a simple numerical, and straightforward, representation. The set of nodes of the graph defines the length of the chromosome and the numbers of the individuals represents the membership of each node to a determined cluster. The following example illustrates this codification:

Example 3.1. Let n be the number of data instances (in this case, we set $n = 9$). Let k be the number of clusters (we set $k = 3$). The chromosomes of Figure 6 shows three possible correct representations. The first chromosome of Figure 6, represents the clusters selection shown in Figure 7.

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	1	1	2	2	2	3	3	3
Chromosome 2	1	2	1	2	3	2	3	3	3
Chromosome 3	2	3	1	2	1	2	1	3	3

Figure 6: Vector-based codification of the GGC algorithm

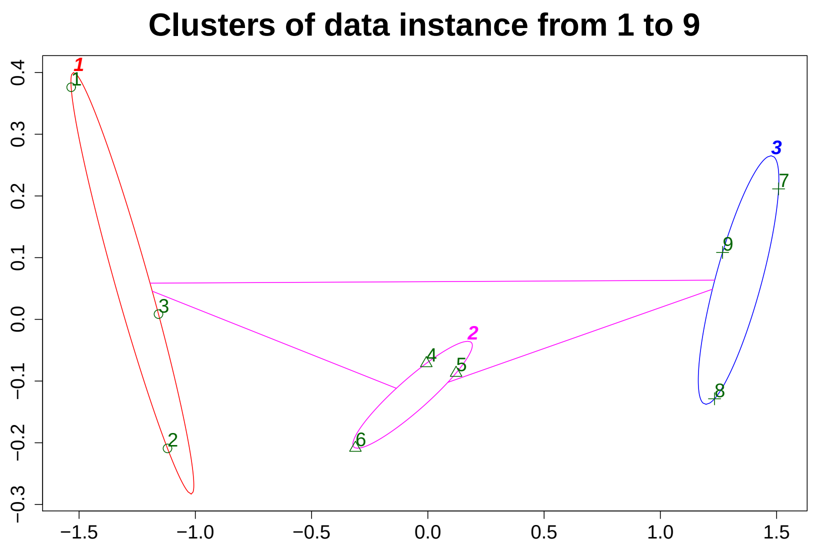


Figure 7: Representation of the clusters for the data instances of chromosome 1 of Figures 6 and 8

3.1.2 Set-based Codification

The second codification is based on sets. Each set represents a cluster and the elements of the sets are the data instances which compose each cluster.

Example 3.2. The chromosomes of example 3.1 can be represented with this new codification as in Figure 8.

3.1.3 Invalid elements

The operations (mutation and crossover) of the genetic algorithm can create invalid elements. These elements are solutions where one or more clusters

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1,2,3\}$	$\{4,5,6\}$	$\{7,8,9\}$
Chromosome 2	$\{1,3\}$	$\{2,4,6\}$	$\{5,7,8,9\}$
Chromosome 3	$\{3,5,7\}$	$\{1,4,6\}$	$\{2,8,9\}$

Figure 8: Set-based codification of the GGC algorithm

could have no elements. In the partitional clustering problem these solutions are not valid because the number of clusters is initially given. To avoid invalid elements, the fitness value assigned to these chromosomes is 0. This value prevents that the elements pass to the next generation.

Some examples of invalid elements for each codification are shown in the following example:

Example 3.3. Figure 9 shows two chromosomes which are invalid elements for the vector-based codification while Figure 10 shows the same elements using the set-based codification and an invalid element which can only be generated using this codification (the chromosome 3). If $k = 3$ and $n = 9$ the first individual has missed cluster 3 and the second cluster 1 (in both figures). In partitional clustering, all the clusters need to have at least one element. Chromosome 3 of Figure 10 repeats the assignation of one element (1) omitting other element (2).

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	1	1	1	1	2	2	2	2
Chromosome 2	2	2	2	3	3	3	2	2	2

Figure 9: Invalid chromosomes of the vector-based codification

3.2 The GGC Operations

This section defines the operations which are used between the chromosomes for each codification. The classical operations have been used. These opera-

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1, 2, 3, 4, 5\}$	$\{6, 7, 8, 9\}$	$\emptyset$
Chromosome 2	$\emptyset$	$\{1, 2, 3, 7, 8, 9\}$	$\{4, 5, 6\}$
Chromosome 3	$\{1, 1, 3, 9\}$	$\{4, 5, 6\}$	$\{7, 8\}$

Figure 10: Invalid chromosomes of the set-based codification

tions are: selection, reproduction, crossover and mutation.

3.2.1 Selection

For both codifications, the selection process selects a subset of the best individuals. These chromosomes are reproduced and also pass to the next generation. It is called a $(\mu + \lambda)$ selection, where μ represents those chromosomes which are chosen, and λ the new chromosomes generated.

3.2.2 Reproduction

For both codifications, the reproduction randomly selects two individuals (using the classical wheel algorithm), and apply the crossover operation to them creating two new individuals.

First, this process calculates the fitness of each individual and the total sum. Next, it calculates the reproduction probability of each chromosome which is equal to its fitness value over the total sum. Finally, each chromosome is randomly selected to be reproduced depending on its reproduction probability.

3.2.3 Crossover

The main problem of crossover is those individuals which have different representation with respect to their codification but represents the same chromosome (see Figures 11 and 12, and the real representation of their chromosomes in Figure 13). For these reason, it is recommendable to relabel the individuals before the application of the crossover. The criteria followed for this relabelling process is to maximize the similarity between the chromosomes which are crossed. It also improves the convergence of the algorithm and reduce the generation of invalid elements. The similarity measure is defined as follows:

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	1	1	2	2	2	3	3	3
Chromosome 2	2	2	2	3	3	3	1	1	1

Figure 11: These two chromosomes represent the same solution, but the name of the clusters appears different using the vector-based codification.

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1, 2, 3\}$	$\{4, 5, 6\}$	$\{7, 8, 9\}$
Chromosome 2	$\{7, 8, 9\}$	$\{4, 5, 6\}$	$\{1, 2, 3\}$

Figure 12: These two chromosomes represent the same solution, but the name of the clusters appears different using the set-based codification.

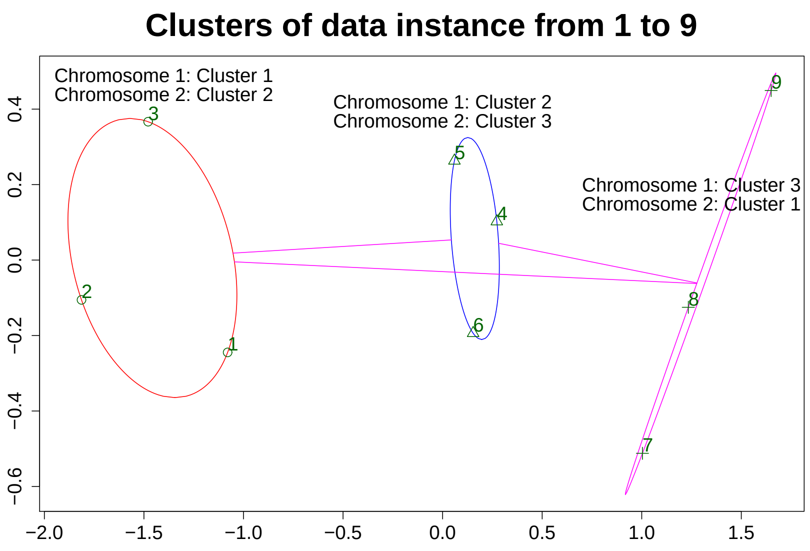


Figure 13: Cluster representation of chromosomes of Figures 11 and 12

Definition 3.1 (Cluster Similarity measure)**.** Let $\{x_1, \ldots, x_n\}$ be a set of elements, and C_i, C_j the clusters which are compared. Their similarity measure is defined by:

$$sim(C_i, C_j) = \frac{1}{2}\left(\frac{\sum_{q=1}^{n} \delta_{C_i}(x_q)\delta_{C_j}(x_q)}{|C_i|} + \frac{\sum_{q=1}^{n} \delta_{C_i}(x_q)\delta_{C_j}(x_q)}{|C_j|}\right) \tag{3}$$

where $|C_i|$ is the number of elements of cluster C_i and $\delta_{C_i}(x_q)$ is the Kronecker δ defined by:

$$\delta_{C_i}(x_q) = \begin{cases} 0 & \text{if } x_q \notin C_i \\ 1 & \text{if } x_q \in C_i \end{cases}$$

The relabelling process can be divided in three fundamental steps:

1. The similarities between the clusters are calculated.
2. The similarities are sorted (decremental order).
3. The second chromosome is relabelled maximizing the similarity with the first chromosome.

Example 3.4 shows the application of this process for the chromosomes shown in Figures 14 and 15. The crossover of the vector-based codification exchanges strings of numbers between two chromosomes (both strings have the same length). In the set-based codification, it keeps the similar elements of both chromosomes and the different elements are randomly distributed amongst the clusters, creating two new elements. Example 3.5 shows the crossover process for each codification.

Example 3.4. This example shows the application of the relabelling process. First, it takes two chromosomes (see Figure 14 for the vector-based codification and Figure 15 for the set-based codification) and calculates the similarities between the clusters. The results are shown in table 1. Once the similarities are calculated, the clusters of chromosome 2 are relabelled from the most similar cluster:

- Cluster 3 is relabelled to 2 (similarity of 83,33%).
- Cluster 1 is relabelled to 3 (similarity of 66,67%).
- Cluster 2 is relabelled to 1 (similarity of 58,33%).

See Figures 16 and 17 to see the results of the relabelling process applied to nodes

	1	2	3	4	5	6	7	8	9
Chromosome 1	1	3	1	2	1	2	3	1	3
Chromosome 2	2	2	2	3	3	3	1	1	1

Figure 14: Example of two chromosomes for the relabelling process using the vector-based codification

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1, 3, 5, 8\}$	$\{4, 6\}$	$\{2, 7, 9\}$
Chromosome 2	$\{7, 8, 9\}$	$\{1, 2, 3\}$	$\{4, 5, 6\}$

Figure 15: Example of two chromosomes for the relabelling process using the set-based codification

Chromosome 1	**Chromosome 2**	**Sim. Calculus**	**Sim. Percentage**
Cluster 1	Cluster 1	$\frac{1}{2}\left(\frac{1}{4}+\frac{1}{3}\right)=\frac{7}{24}$	$29,17\%$
Cluster 1	Cluster 2	$\frac{1}{2}\left(\frac{2}{4}+\frac{2}{3}\right)=\frac{7}{12}$	**58,33%**
Cluster 1	Cluster 3	$\frac{1}{2}\left(\frac{1}{4}+\frac{1}{3}\right)=\frac{7}{24}$	$29,17\%$
Cluster 2	Cluster 1	$\frac{1}{2}\left(\frac{0}{2}+\frac{0}{3}\right)=0$	0%
Cluster 2	Cluster 2	$\frac{1}{2}\left(\frac{0}{2}+\frac{0}{3}\right)=0$	0%
Cluster 2	Cluster 3	$\frac{1}{2}\left(\frac{2}{2}+\frac{2}{3}\right)=\frac{5}{6}$	**83,33%**
Cluster 3	Cluster 1	$\frac{1}{2}\left(\frac{2}{3}+\frac{2}{3}\right)=\frac{2}{3}$	**66,67%**
Cluster 3	Cluster 2	$\frac{1}{2}\left(\frac{1}{3}+\frac{1}{3}\right)=\frac{1}{3}$	$33,33\%$
Cluster 3	Cluster 3	$\frac{1}{2}\left(\frac{0}{3}+\frac{0}{3}\right)=0$	0%

Table 1: Similarities from chromosomes shown in Figure 14

nodes

	1	2	3	4	5	6	7	8	9
Chromosome 1	1	3	1	2	1	2	3	1	3
Chromosome 2	1	1	1	2	2	2	3	3	3

Figure 16: Example of the two chromosomes of figure 14 after the relabelling process applied to chromosome 2 using the vector-based codification

	Cluster 1	Cluster 2	Cluster 3
Chromosome 1	$\{1, 3, 5, 8\}$	$\{4, 6\}$	$\{2, 7, 9\}$
Chromosome 2	$\{1, 2, 3\}$	$\{4, 5, 6\}$	$\{7, 8, 9\}$

Figure 17: Example of the two chromosomes of figure 15 after the relabelling process applied to chromosome 2 using the vector-based codification

Example 3.5. Figures 18 and 19 shows the crossover process. The first figure exemplifies the vector-based codification. In this example, the exchange is between two sections of the chromosomes. This sections are randomly selected. In this case, the interval is from 4 to 7 (both numbers included). The second figure shows the crossover of the set-based codification. In this case, the common parts of both chromosomes are kept and the rest of the data instances are randomly distributed by all the clusters.

	nodes								
	1	2	3	4	5	6	7	8	9
Chromosome 1	1	1	1	**2**	**2**	**2**	**3**	3	3
Chromosome 2	1	2	1	**2**	**3**	**2**	**3**	3	3
New Chromosome 1	1	1	1	**2**	**3**	**2**	**3**	3	3
New Chromosome 2	1	2	1	**2**	**2**	**2**	**3**	3	3

Figure 18: Crossover using the vector-based codification after relabelling

3.2.4 Mutation

In this algorithm an adaptive mutation has been developed for both codification. It works as follows:

1. For each chromosome, it randomly chooses if the mutation is applied (the mutation probability is fixed at the beginning of the algorithm)-

2. When a chromosome is chosen, it decides the alleles which are mutated. The decision considers the probability of the allele to belong to the cluster which have assigned. If the probability is high, the allele has a low

	Clusters 1	Clusters 2	Clusters 3
Chromosome 1	$\{1,2,3\}$	$\{4,5,6\}$	$\{7,8,9\}$
Chromosome 2	$\{2,3\}$	$\{5,6,7\}$	$\{1,4,8,9\}$
Intersection	$\{2,3\}$	$\{5,6\}$	$\{8,9\}$
New Chromosome 1	$\{2,3,\mathbf{4}\}$	$\{5,6,\mathbf{1}\}$	$\{8,9,\mathbf{7}\}$
New Chromosome 2	$\{2,3,\mathbf{1},\mathbf{7}\}$	$\{5,6,\mathbf{4}\}$	$\{8,9\}$

Figure 19: Crossover using the set-based codification

probability of mutate and vice versa. In this algorithm, this probability is calculated applying the metric defined in the fitness function to one allele.

3. The alleles are mutated depending on the codification:
    - The vector-based codification changes the allele value. The new value is a random number between 1 and the number of clusters.
    - The set-based codification moves the allele to other cluster. It randomly chooses the new cluster which will contain the allele.

Example 3.6. Figure 20 shows the mutation of the vector-based codification. It is focused on a chromosome which have been selected to be mutated. The first and seventh alleles have been randomly chosen to be changed. Figure 21 shows the same process applied to the set-based codification. In this case, third and eighth alleles have been moved from first and third clusters to third and second, respectively.

nodes

	1	2	3	4	5	6	7	8	9
Chromosome	**1**	1	1	2	2	2	**3**	3	3
	↓						↓		
Chromosome Mutated	**2**	1	1	2	2	2	**2**	3	3

Figure 20: Mutation of two alleles in a chromosome using the vector-based codification

	Cluster 1	Cluster 2	Cluster 3
Chromosome	$\{1, 2, \mathbf{3}\}$	$\{4, 5, 6\}$	$\{7, \mathbf{8}, 9\}$
Chromosome Mutated	$\{1, 2\}$	$\{4, 5, 6, \mathbf{8}\}$	$\{7, 9, \mathbf{3}\}$

Figure 21: Mutation of two alleles in a chromosome using the set-based codification

3.3 The GGC Fitness Functions

In this section, the two different fitness functions which have been designed are described. Theoretical and experimental results which have been concluded from the experiments with these fitness (for example, the weight clustering coefficient fitness) are shown in the following section.

3.3.1 The Weighted Clustering Coefficient Fitness Function

The first fitness function is the measure of the Global Weight Clustering Coefficient, previously described in section 2.6.1.

It uses the following metric which is applied to the Similarity Graph (we suppose an undirected weight graph):

$$C_i^w = \frac{\sum_{j,h} \frac{w_{ij}+w_{ih}}{2} a_{ij} a_{ih} a_{jh}}{S_i(k_i - 1)}$$

where $S_i = \sum_j w_{ij}$ and k_i is the number of neighbours of the node i.
This fitness looks for individuals which have high similarity with their neighbours and whose neighbours also have high similarity between them.

3.3.2 KNN-Minimal Cut fitness

This fitness function is a combination of the classical K-Nearest Neighbourhood (KNN) [49] algorithm and the Minimal Cut measure [68]. KNN assigns an element to a cluster if the neighbours are in the same cluster. It is useful to guarantee the continuity condition which is frequent in the Spectral Clustering solutions. To control the separation between the elements of the clusters, the Minimal Cut measure is used. It guarantees that those elements which clearly belongs to different clusters are not assigned to the same cluster. The K value for KNN is initially given.

Algortihm 4 shows the pseudo-code of the fitness: KNN covers all the nodes and check if the K-closest elements are in the same cluster (lines 9 to

12). The fitness value of this metric is the mean of the percentage of well-classified neighbours of all the individuals in a cluster (lines 10 and 13). The Minimal Cut measure calculates the average value edge weights which have been removed (lines 11 and 14). The final value of the fitness if the product of the KNN metric and the subtraction between one and the Minimal Cut metric (line 16), both metrics have the same range: [0,1]. Therefore, the algorithm maximizes the value of $\frac{TotalKNN}{|C|} \times \left(1 - \frac{TotalMC}{|C|}\right)$ (line 16) where:

$$TotalMC = \sum_{x \in C} \frac{\sum_{y \notin C_x} w_{xy}}{|\{y | y \notin C_x\}|}$$

$$TotalKNN = \sum_{x \in C} \frac{|\{y | y \in \Gamma(x) \wedge y \in C_x\}|}{|\Gamma(x)|}$$

In these formulas, C represents the set of clusters and $\Gamma(x)$ represents the neighbourhood of the element x. It reduces the weight values of the edges which are cut and improve the proximity of the neighbours.

Algorithm 4 Pseudo-code of the KNN-Minimal Cut Fitness Function

Input: A n-vector of elements with values between 0 and k where k is the number of clusters and a variable $neighbours$ which represents the number of neighbours for the KNN measure.

Output: A value between 0 and 1 which corresponds with the fitness achieved.

```
 1: TotalKNN = 0.;
 2: TotalMC = 0.;
 3: Generate the set of $k$ Clusters: $C$.
 4: for all $C_a \in C$ do
 5:     if $C_a = \emptyset$ then
 6:         return 0
 7:     end if
 8:     SumKNN = 0; SumMC = 0.
 9:     for all $ind \in C_a$ do
10:         SumKNN += $PofKNN(neighbours, ind)$ {It calculates the
            percentage of neighbours for the individual $ind$ which are as-
            signed to the same cluster.}
11:         SumMC += $AvEdWCut(ind)$ {It calculates the average value
            of the edge weights which have been cut from $ind$.}
12:     end for
13:     TotalKNN += SumKNN / $|C_a|$; {$|C_a|$ represents the number of
        elements of $C_a$.}
14:     TotalMC += SumMC / $|C_a|$;
15: end for
16: return $\frac{TotalKNN}{|C|} \times \left(1 - \frac{TotalMC}{|C|}\right)$
```

4 Experimentation

This section shows the different experiments which have been applied using the GGC algorithm. These experiments are both synthetic and real-world experiments. First, a comparison between the two codifications and the fitness functions introduced in section 3 is presented. Second, the metrics used in the experiments are explained. Third, the experiments analyse the robustness of the GGC Algorithm. Next, the GGC Algorithm results are compared with other algorithm (K-means, EM and Spectral Clustering) using synthetic datasets. Finally, these algorithm are also tested with real datasets.

4.1 Comparison of Codifications and Fitness Functions

This section compares the advantages and disadvantages of the two codifications and the two fitness functions.

4.1.1 Codifications comparison

The two codifications are similar and can be applied to the problem with the same results, because the fitness function is the responsible to guide the algorithm to the best solutions. However, they presents the following differences:

- Omitting the relabelling process, the vector-based crossover operation is faster than the set-based crossover. In the vector-based case, the crossover is $O(n)$ because only one loop is necessary to swap the values of two vectors. For the set-based case, the crossover is $O(n^2)$ because two nested loops are necessary to find the common elements of two sets.

- The mutation effort of the two algorithms is almost the same, although the vector-based codification is the fastest. In the mutation of any kind of chromosome, all of its alleles are visited in both cases, but in the vector-based codification the value changes instantly when the mutation is applied while in the set-based codification the value is extracted from one set and introduced in another set.

- Both codifications can use the relabelling process, however the set-based codification simplifies the similarity calculus with the intersection operation.

- The algorithm has a probably to fall in a local maximum value of fitness (which is a common problem when heuristic methods are used). This convergence problem has not been deeply study in the GGC algorithm, however these problems depend on the operations of the genetic algorithm. To compare the codifications convergence, the Spirals dataset [42] has been tested with the two codifications. Figure 22 shows a graphic of the average convergence velocity (for 50 runs of the algorithm per codification) with the following parameters of the genetic algorithm:

- Population: 200
- Generations: 2000
- Crossover probability: 0.3
- Mutation probability: 0.5 [1]
- Selection $(\mu + \lambda)$: The 50 best individual are selected from the previous generation.

In this case the vector-based codification converges faster than the set-based codification.

Owing to the facts that have been exposed above, the vector-based based codification reduce the computation effort. Because of this, it has been chosen to carry out the experiments.

4.1.2 Fitness Functions comparison

The experimental results show that the fitness function of the Weight Clustering Coefficient in some cases obtains always the maximum value. The analysis of this problem shows that it was when the Similarity Graph was fully connected (it means that all the weights are bigger than 0). To understand this situation, it has been mathematically proved that this problem is a "metric mistake". The following theorem shows the proof:

Theorem 4.1. *Suppose that G is a graph (with 3 elements or more) and W is the matrix of the weights of the graph. If $w_{ij} > 0$ $\forall i, j$ then $CCW_i = 1 \forall i$.*

[1]The mutation operation is adaptive (see Section 3.2.4), an allele only changes if it has a low probability value to belong to the cluster which has been assigned for it. Hence, the mutation helps the algorithm to converge faster, it is not a complete random process such as in the classical Genetic Algorithms.

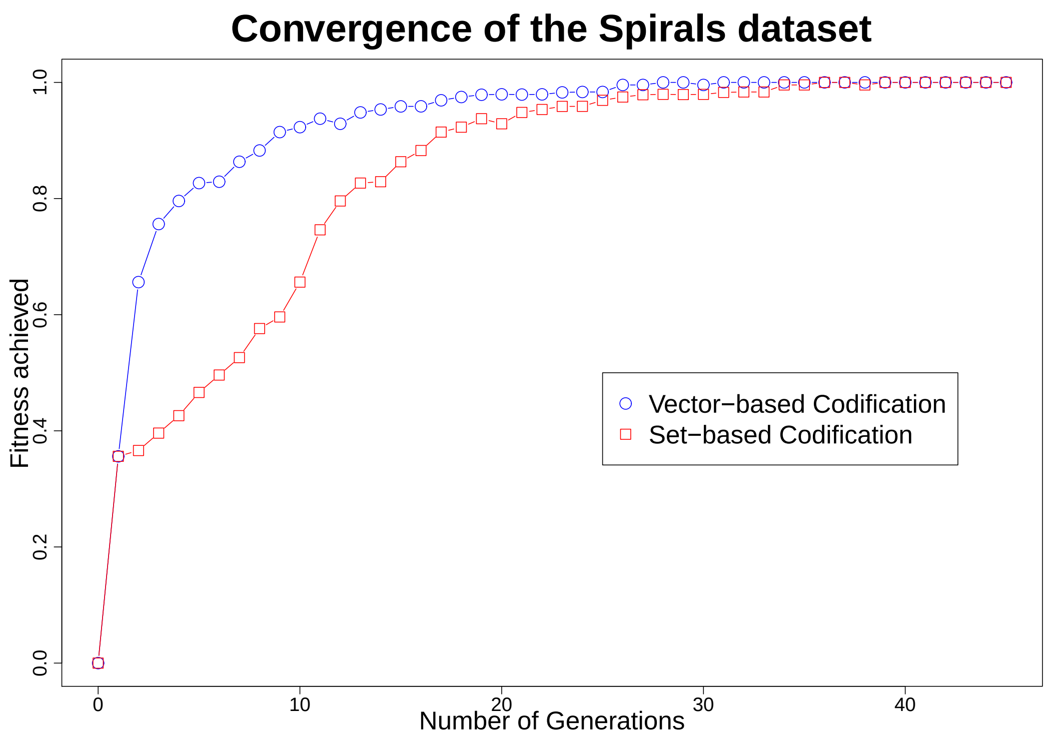


Figure 22: Convergence of the genetic algorithm. The convergence is produced in the 30 generation

Proof. We choose a random element i which has n neighbours. Let $x_1, \dots, x_n$ the weight values from the node i to its n neighbours. From the definition of the C_i^w we have:

$$C_i^w = \frac{\sum_{j,h} \frac{w_{ij}+w_{ih}}{2} a_{ij} a_{ih} a_{jh}}{S_i(k_i - 1)}$$

If we calculate S_i we have:

$$S_i = x_1 + \dots + x_n$$

In this case $a_{ij} = 1 \ \forall i, j$ and $k_i = n$, then:

$$C_i^w = \frac{\sum_{j,h} \frac{x_j+x_h}{2}}{S_i(n-1)}$$

If we sort the sum elements we have the following:

$$\begin{array}{ccccccccccc}
0 & + & \frac{x_1+x_2}{2} & + & \frac{x_1+x_3}{2} & + & \frac{x_1+x_4}{2} & + & \dots & + & \frac{x_1+x_n}{2} \\
\frac{x_2+x_1}{2} & + & 0 & + & \frac{x_2+x_3}{2} & + & \frac{x_2+x_4}{2} & + & \dots & + & \frac{x_2+x_n}{2} \\
\frac{x_3+x_1}{2} & + & \frac{x_3+x_2}{2} & + & 0 & + & \frac{x_3+x_4}{2} & + & \dots & + & \frac{x_3+x_n}{2} \\
\frac{x_4+x_1}{2} & + & \frac{x_4+x_2}{2} & + & \frac{x_4+x_3}{2} & + & 0 & + & \dots & + & \frac{x_3+x_n}{2} \\
\vdots & + & \vdots & + & \vdots & + & \vdots & + & \dots & + & \vdots \\
\frac{x_n+x_1}{2} & + & \frac{x_n+x_2}{2} & + & \frac{x_n+x_3}{2} & + & \dots & + & \frac{x_n+x_{n-1}}{2} & + & 0
\end{array}$$

If we consider the symmetries of the sum, and we sum the elements which are symmetric, then we have:

$$\begin{array}{ccccc}
(x_1+x_2)+(x_1+x_3)+(x_1+x_4)+\dots+(x_1+x_n) & = & (n-1)x_1 & + & x_2+\dots+x_n \\
(x_2+x_3)+(x_2+x_4)+(x_2+x_5)+\dots+(x_2+x_n) & = & (n-2)x_2 & + & x_3+\dots+x_n \\
(x_3+x_4)+(x_3+x_4)+(x_3+x_5)+\dots+(x_3+x_n) & = & (n-3)x_3 & + & x_4+\dots+x_n \\
\vdots & = & \vdots & + & \vdots \\
(x_{n-1}+x_n) & = & (1)x_{n-1} & + & (1)x_n
\end{array}$$

In this case, if we sum, for example, the x_2 that is left in the first sum to $(n-2)x_2$ we have $(n-1)x_1$, if we do the same with the x_3 left in the first and second sum to $(n-3)x_3$ we have $(n-1)x_3$. If we continue until x_n we have $(n-1)x_i\ \forall i$. Then:

$$C_i^w = \frac{(n-1)(x_1+\dots+x_n)}{S_i(n-1)}$$

We know that $S_i = x_1+\dots+x_n$ then:

$$C_i^w = \frac{(n-1)(x_1+\dots+x_n)}{(x_1+\dots+x_n)(n-1)} = 1$$

□

Since the Similarity Graph construction that was chosen is the fully connected graph (see section 2.4.1), the only fitness that has been applied in the experiments is the KNN-Minimal Cut fitness (the other fitness always achieves the maximum value for the fully connected graph because its weights are always bigger than 0). The fully connected approximation was chosen because the GGC algorithm tries to maximized the robustness of the clustering selection with respect to the metrics (which is explained in the following subsection). If the ϵ-neighbourhood graph or the k-nearest neighbour graph are chosen (see section 2.4) the Similarity Graph increments the number of zero similarities which is not desirable when all the elements could have a non-zero similarity between them. It could reduce the robustness of the algorithm and supposes a higher dependency to parameters (in this case, the Similarity Graph generation parameters: the ϵ value of the ϵ-neighbourhood graph and the k value of the k-nearest neighbour graph).

4.2 Metrics

All the techniques use the metrics which have been mentioned before: K-means and EM use the Euclidean Distance Metric and Spectral Clustering and GGC use the Radian Basis Function (with the σ parameter optimized in the Spectral Clustering case). There are not applications with other metrics or kernels because the goal of the GGC algorithm is to be robust enough to separate the dataset without orders of magnitude problems. It tries to give the same results if two different metrics give the same relative distance between the objects, that is, if there are, for example, three objects: o_1, o_2, o_3, and the distances between them are $d_{12}^{M_1}$, $d_{23}^{M_1}$, $d_{13}^{M_1}$, $d_{12}^{M_2}$, $d_{23}^{M_2}$, $d_{13}^{M_2}$ where M_1 and M_2 are the metrics and the two metric distances satisfies the same order relationship:

$$d_{12}^{M_1} < d_{23}^{M_1} < d_{13}^{M_1}$$

$$d_{12}^{M_2} < d_{23}^{M_2} < d_{13}^{M_2}$$

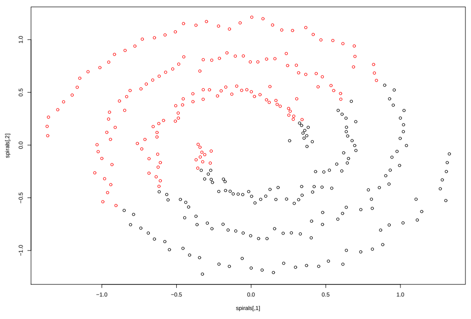
Then the clustering results should be almost the same (except, for example, when some of these distances are infinity or zero).

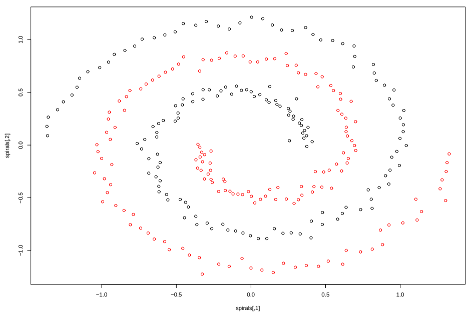
4.3 Comparing SC and GGC Robustness

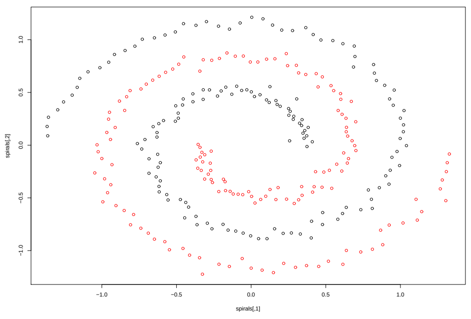
An important problem of the Spectral Clustering algorithm is its dependency to the parameters of the Similarity Function. The GGC algorithm has been designed to avoid this problem. The KNN metric which is applied in the fitness step provides a higher robustness to the algorithm compared to the traditional Spectral Clustering algorithm. It does not depend on the order of magnitude of the distances calculated by the metric. Figure 23 shows a clear example. In this case, the Spectral Clustering algorithm (implemented in the "kernlab" package of CRAN [42]) is compared with the GGC algorithm. In the "kernlab" package, Karatzoglou et al. implements the Random Walks Normalized Spectral Clustering algorithm. They use the Gaussian RBF Kernel to set the similarity graph. It is defined by:

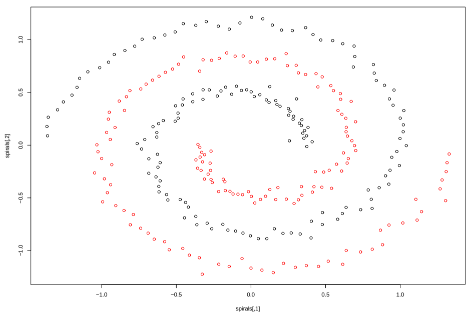
$$K_{ij} = e^{-\sigma||x_i - x_j||^2} \tag{4}$$

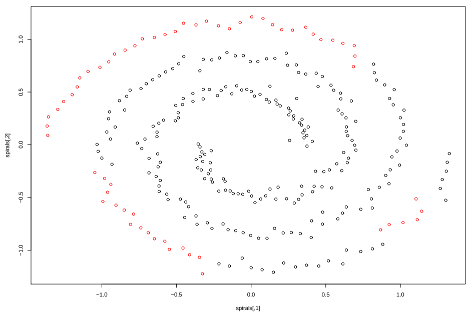
Where K is the Similarity Graph, x_i, x_j are data instances, and σ is the parameter which changes the order of magnitude. The experimental results show that the clustering technique clearly depends on the σ parameter. Figure 23 shows the different clustering results of the Spectral Clustering and the GGC algorithm modifying the σ parameter.


(a) Spectral Clustering results of Spirals for $\sigma = 2$


(b) GGC results of Spirals for $\sigma = 2$


(c) Spectral Clustering results of Spirals for $\sigma = 500$


(d) GGC results of Spirals for $\sigma = 500$


(e) Spectral Clustering results of Spirals for $\sigma = 2000$

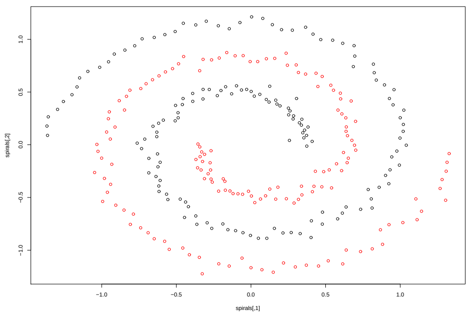

(f) GGC results of Spirals for $\sigma = 2000$

Figure 23: Spectral Clustering and GGC results for the Spirals[42] dataset with $\sigma = 2$, $\sigma = 500$, $\sigma = 2000$, respectively

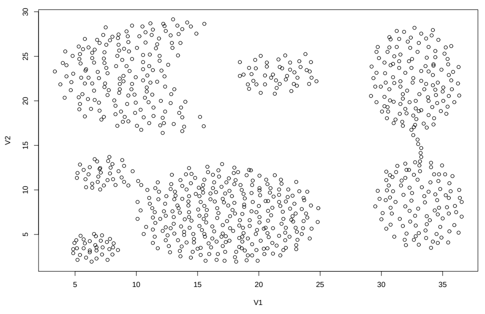
4.4 Experiments on Synthetic data

In this section the different datasets which have been considered for the experimentation are explained and analysed. These datasets have been extracted from different clustering works which study the behaviour of different approaches similar to Spectral Clustering [13, 29, 32, 40, 73, 81]

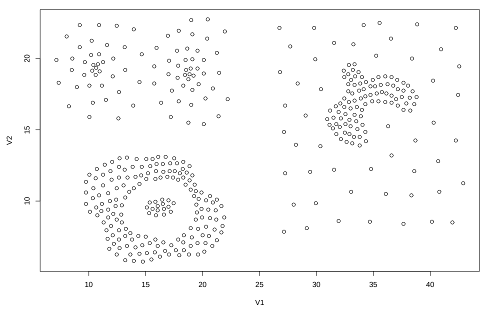
4.4.1 Data description

K-means, EM and SC have been tested with different datasets. These datasets are 2-Dimensional data which can be separated by human intuitions but are problematic for the classical clustering algorithms. We have analysed the following datasets (figure 24 shows the original datasets):

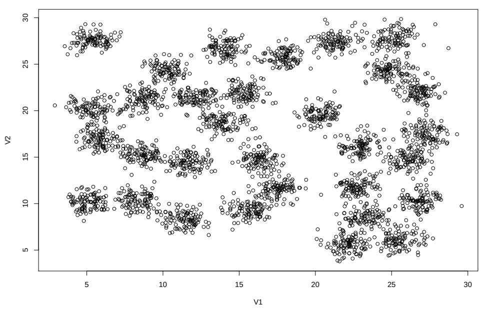
- Aggregation[32]: This dataset is composed by 7 clusters, some of them can be separated by parametric clustering.
- Compound[81]: There are 6 clusters which are only separable by non-parametric methods (or special kernels if parametric clustering is applied).
- D31 [73]: This data has 31 clusters with a high level of noise.
- Flame [29]: This dataset have three ideal clusters: the first is the base of the figure, the second is the top and the last are two outliers at the top-left of the image.
- Jain [40]: This dataset is composed by two surfaces with different density and a clear separation.
- PathBased[13]: This dataset have 2 clusters which can be separated by a parametric method and another cluster which can only be separated by a non-parametric method. This example is problematic for algorithms such as Spectral Clustering because this algorithm is sensitive to the noisy points.
- R15: [73]: Similar to D31, this dataset is divided in 15 clusters which are clearer separated.
- Spiral: [13]: In this case, there are 3 spirals close to each other.



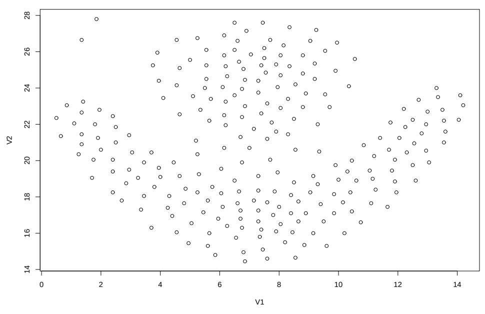
(a) Original Aggregation dataset



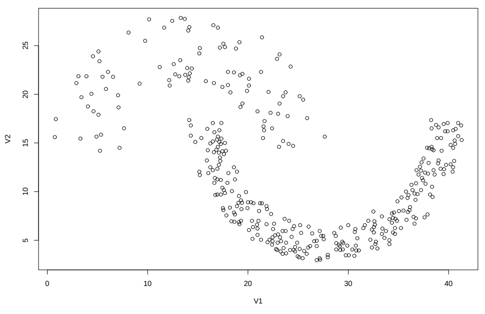
(b) Original Compound dataset



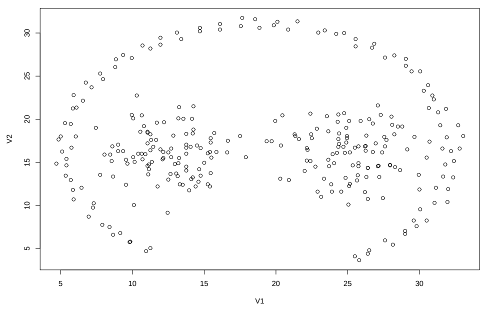
(c) Original D31 dataset



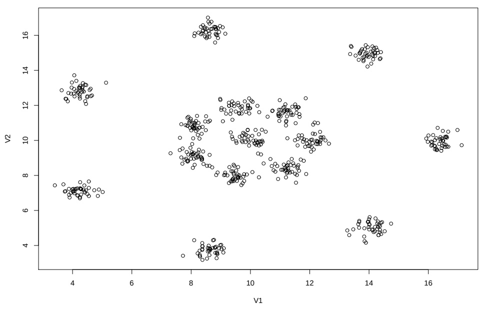
(d) Original Flame dataset



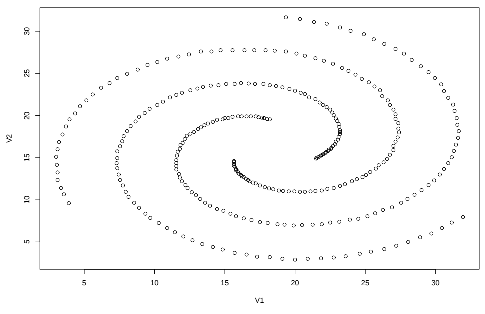
(e) Original Jain dataset



(f) Original PathBased dataset

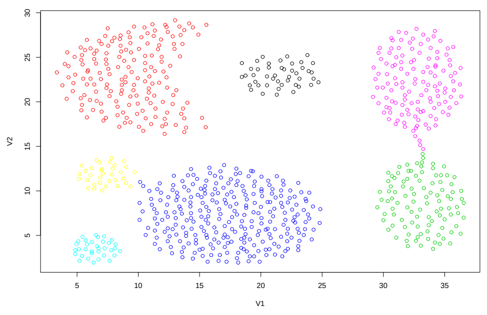


(g) Original R15 dataset

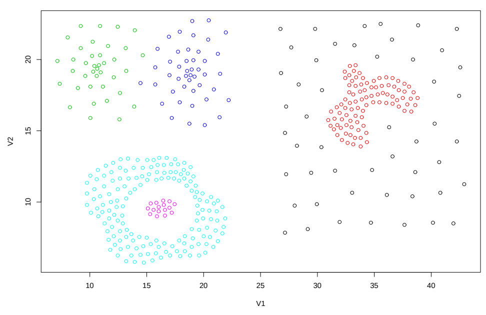


(h) Original Spiral dataset

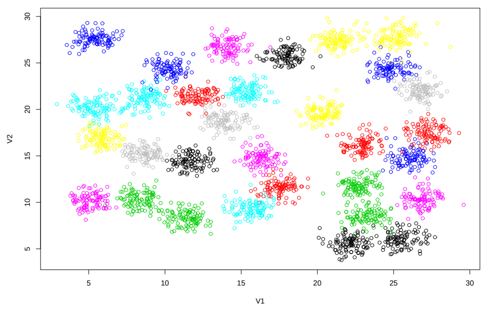
Figure 24: The original images of the synthetic datasets



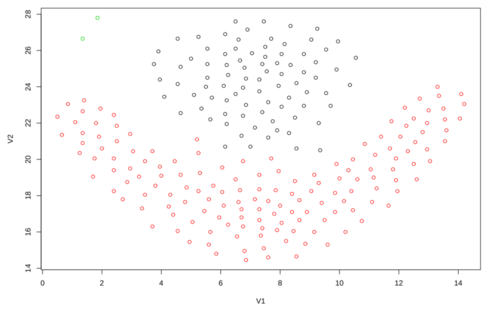
(a) Ideal human separation for Aggregation



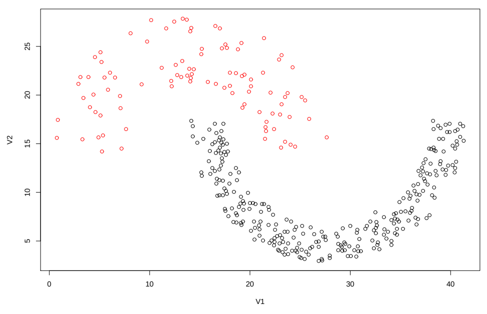
(b) Ideal human separation for Compound



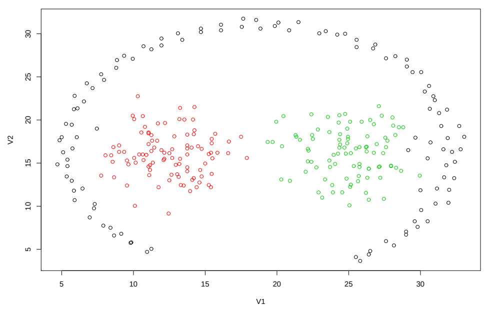
(c) Ideal human separation for D31



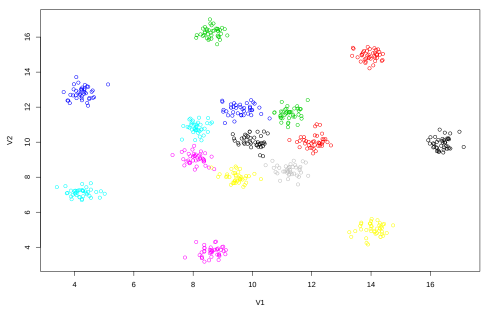
(d) Ideal human separation for Flame



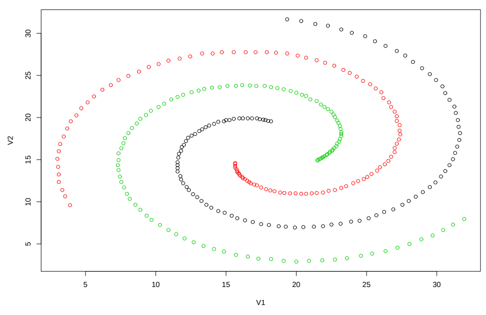
(e) Ideal human separation for Jain



(f) Ideal human separation for PathBased

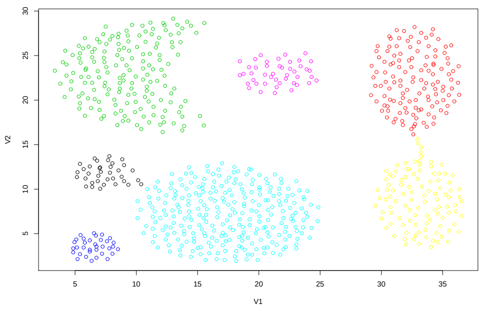


(g) Ideal human separation for R15

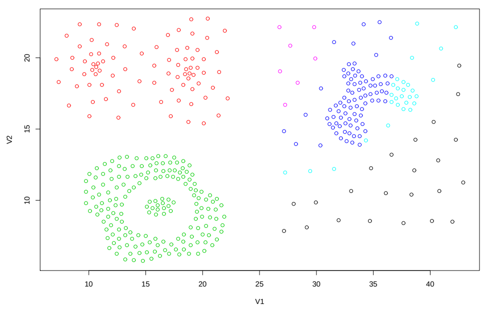


(h) Ideal human separation for Spiral

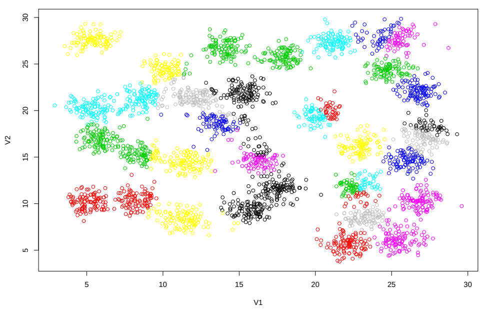
Figure 25: The ideal human separation of the synthetic datasets



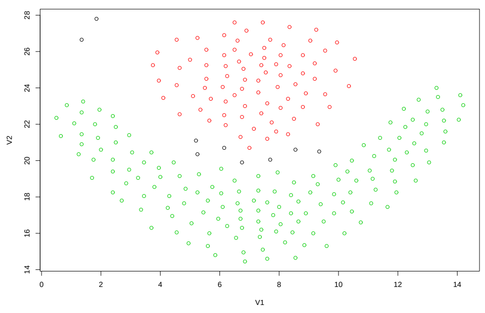
(a) SC results for Aggregation



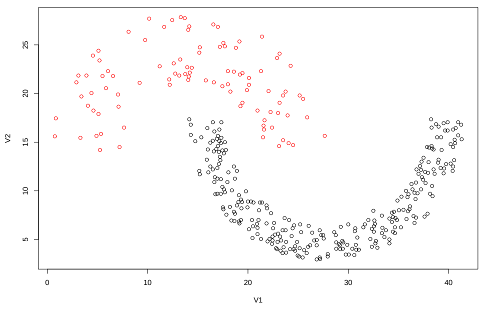
(b) SC results for Compound



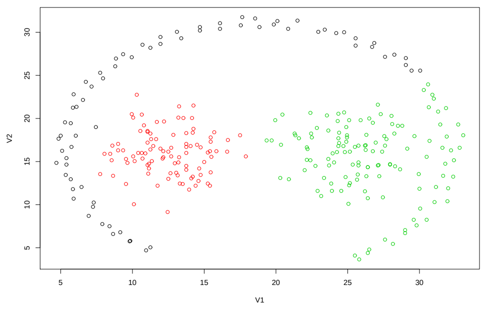
(c) SC results for D31



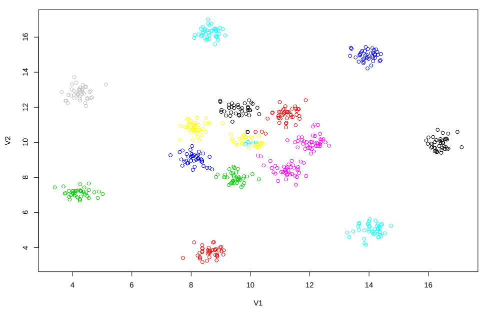
(d) SC results for Flame



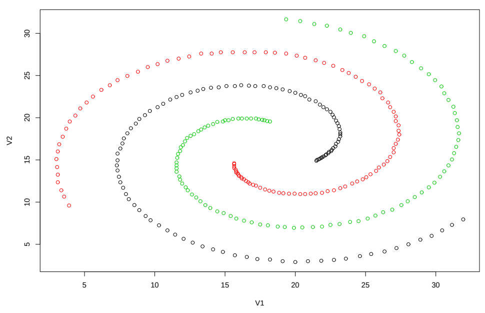
(e) SC results for Jain



(f) SC results for PathBased

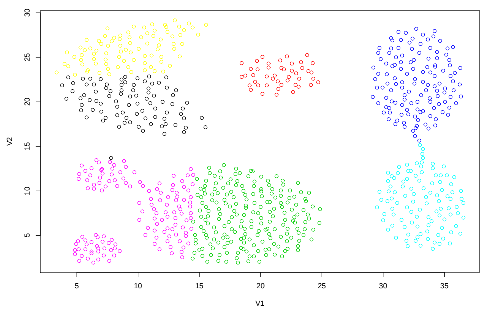


(g) SC results for R15

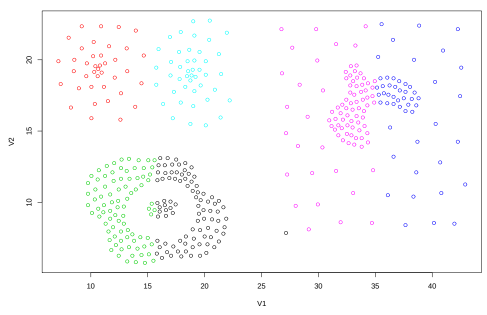


(h) SC results for Spiral

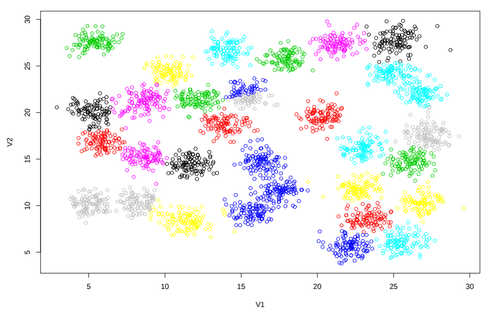
Figure 26: Spectral Clustering results for the synthetic datasets. The algorithm has problems with Compound, D31, Flame, PathBased and R15



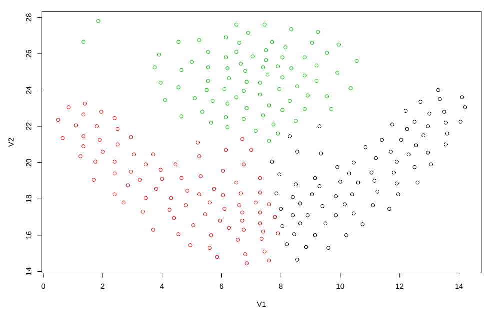
(a) K-means results for Aggregation



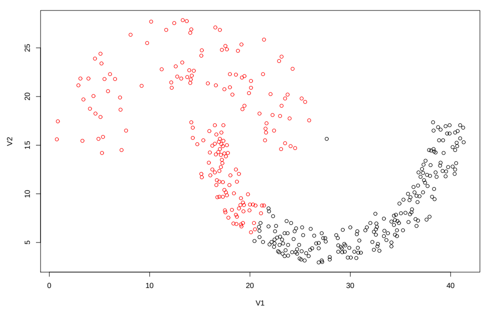
(b) K-means results for Compound



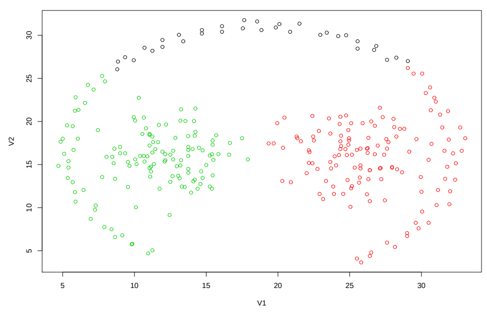
(c) K-means results for D31



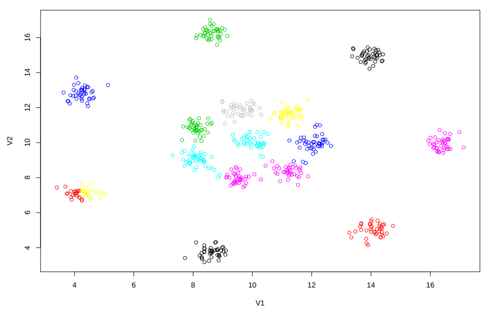
(d) K-means results for Flame



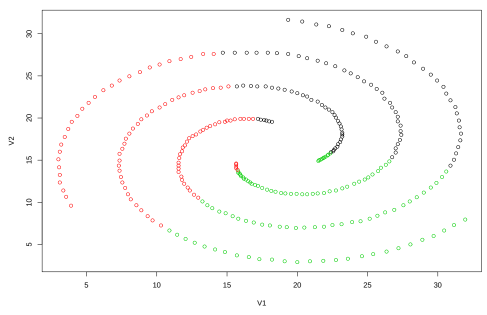
(e) K-means results for Jain



(f) K-means results for PathBased

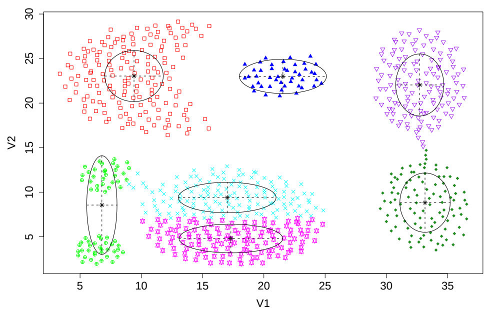


(g) K-means results for R15

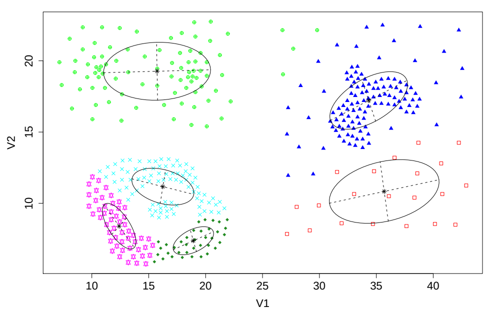


(h) K-means results for Spiral

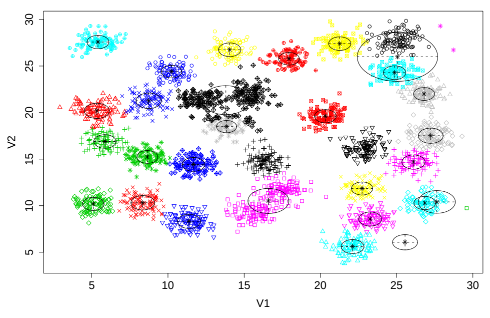
Figure 27: K-means results for the synthetic datasets. The algorithm has problems with all the datasets



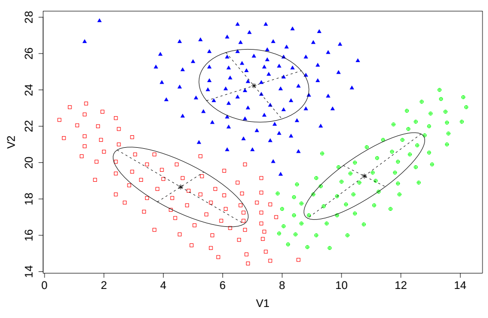
(a) EM results for Aggregation



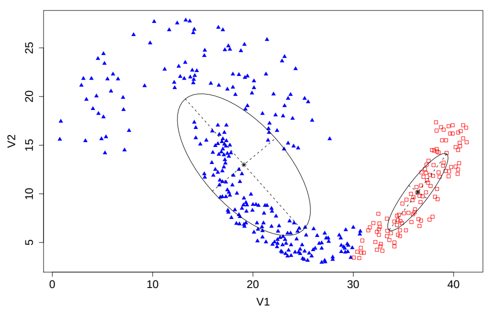
(b) EM results for Compound



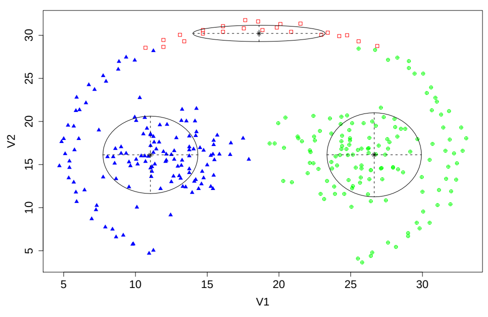
(c) EM results for D31



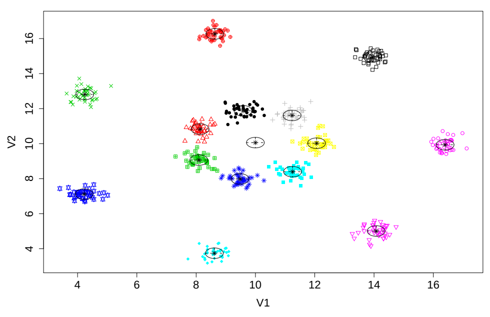
(d) EM results for Flame



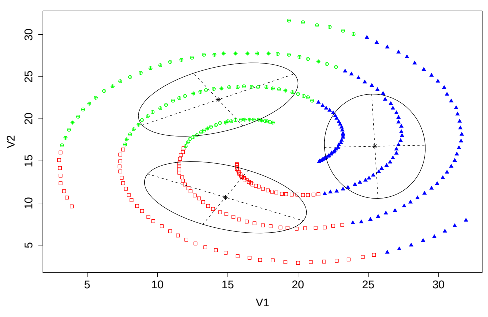
(e) EM results for Jain



(f) EM results for PathBased

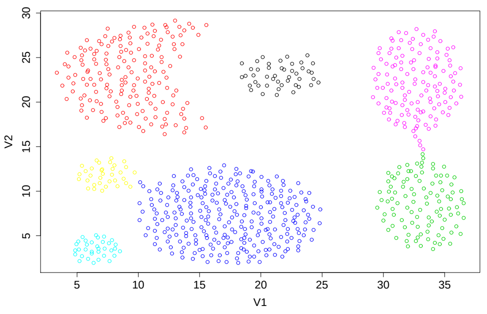


(g) EM results for R15

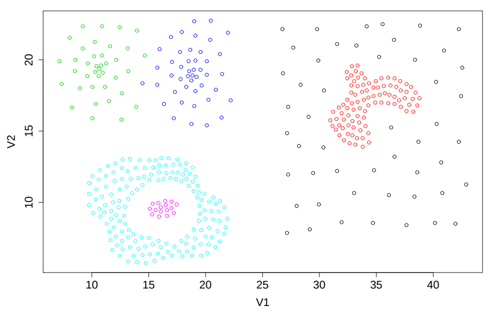


(h) EM results for Spiral

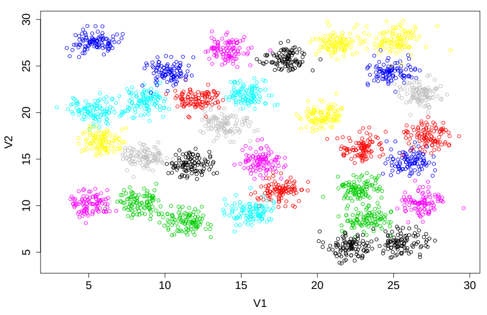
Figure 28: EM results for the synthetic datasets. The algorithm has problems with all the datasets except R15



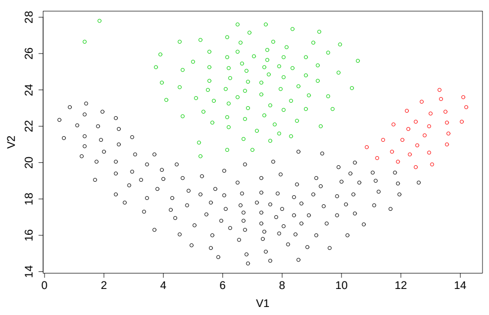
(a) GGC results for Aggregation



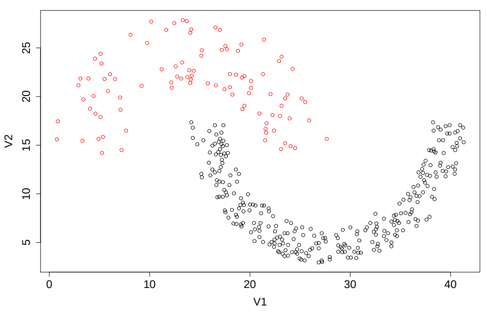
(b) GGC results for Compound



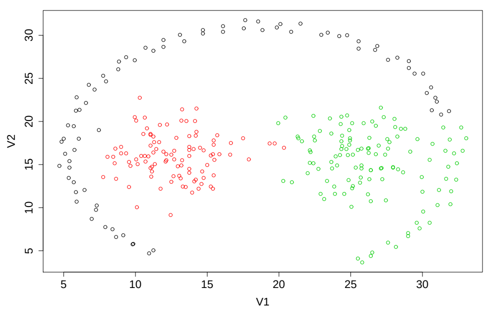
(c) GGC results for D31



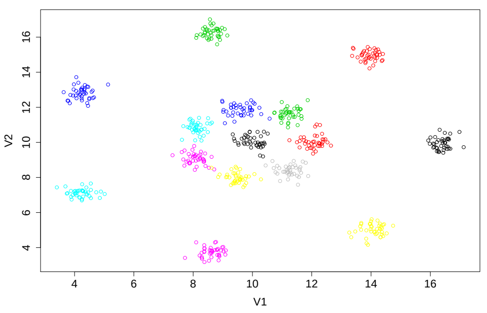
(d) GGC results for Flame



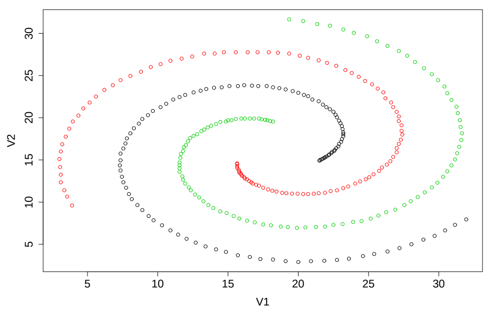
(e) GGC results for Jain



(f) GGC results for PathBased



(g) GGC results for R15



(h) GGC results for Spiral

Figure 29: GGC results for the synthetic datasets. The algorithm has problems with Flame and PathBased

4.4.2 Results obtained from the clustering algorithms using synthetic data

The selected clustering algorithms (Kmeans, Em using a Gaussian Mixture model estimator, Spectral Clustering and the GGC algorithm) have been applied to the previous described datasets. All the algorithms have been executed 50 times and their best results have been selected. Figure 24 shows the original datasets and Figure 25 the ideals clusters. The analysis of the rest of figures shows different problems for the clustering techniques:

- Figure 26 shows the results of Spectral Clustering applying the algorithm of Ng [61] to find the optimal σ. The figure shows that Aggregation, Jain and Spirals are not problematic for the Spectral Clustering algorithm. However, Compound, Flame, PathBased, D31 and R15 are more problematic. Compound is difficult to classify for the Spectral Clustering algorithm because the distribution of the data is highly heterogeneous. In the case of Flame, there is not a clear boundary between the clusters. It makes difficult the application of the algorithm. D31 and PathBased have noisy information, it produces several desviations for the SC algorithm. Finally, R15 has also noisy information in the central clusters.

- Figure 27 shows the results of Kmeans using the euclidean distance metric. It shows that K-means is useless for almost all the cases. It is understandable since Kmeans is a parametric algorithm where the parameter is a centroid whose position is optimized by the algorithm. In the case of Compound, for example, the clusters of the top-left position of the image are well classified, however it is impossible, with these conditions, that the algorithm classifies correctly the bottom-left two clusters because one cluster surrounds the other. It hav the same problem with Jain, Spirals, PathBased and Flame. In the case of Aggregation, the worst misclassification is related to the three clusters of the bottom-left. In this case, the different sizes of the clusters influence in the selection. The D31 and R15 misclassification is a consequence of a local minimum convergence of the algorithm caused by the noisy information.

- Figure 28 shows the results of EM also using the Euclidean distance metric. It shows that EM is better than K-means but is also useless for theses cases. It has similar problems to Kmeans. It achieves better results for R15 although the rest of the datasets are misclassified.

- Figure 29 shows the results of the genetic algorithm. Table 2 shows the fitness value achieved by the GGC algorithm for the results obtained in each cluster selection. Table 2 also shows the parameters selection of the genetic algorithm for each case. The results show that the genetic algorithm only have problems with the noisy cases: Flame and Path-Based. The reason is related to a boundary problem. It is difficult for the algorithm determined the limit of the clusters when it is not clear.

Dataset	**Population**	**Generations**	**Crossover**	**Mutation**	**Selection**	**Max. Fitness**
Aggregation	100	2000	0.4	0.01	50	0.9928
Compound	200	2000	0.5	0.01	50	0.9552
Flame	100	2000	0.4	0.01	50	0.9828
Jain	100	500	0.4	0.2	50	1.0
Pathbased	100	2000	0.4	0.01	50	1.0
R15	200	2000	0.5	0.3	50	0.9850
Spiral	100	500	0.4	0.01	50	1.0
D31	200	5000	0.7	0.4	50	0.9445

Table 2: Best parameter selection found of the GGC algorithm for the different synthetic datasets and the fitness achieved by the GGC algorithm. The K value of the KNN-Minimal Cut fitness is always set to 2

4.5 Experiments on Real-World data

In this section the experiments are focused on real-world datasets which have been previously classified by humans. Here, the accuracy of the algorithm is tested.

4.5.1 Dataset Description

The experiments have been also applied on three real datasets extracted from the UCI Machine Learning Repository [27]:

- **Iris**: This dataset is a well-know dataset. It has 150 instance of 3 different classes (50 in each class). Each class refers to a type of iris plant. The classes are: Iris Setosa, Iris Versicolour and Iris Virginica. Each instance has 4 attributes which are: Sepal length in cm, Sepal width in cm, Petal length in cm and Petal width in cm. It does not have missing values.

- **Wine**: This dataset has 178 instances. Each instance has 13 attributes and can belongs to 1 of the three different classes. Each class refers to

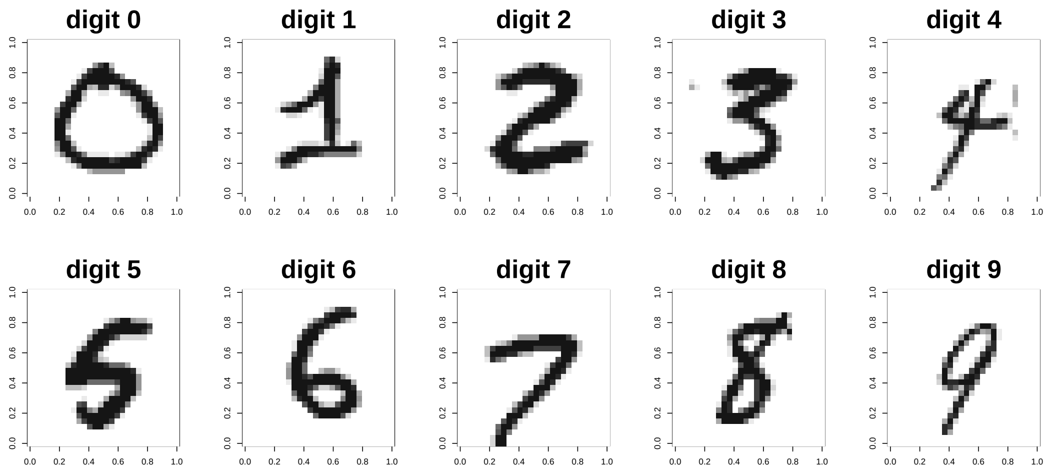


Figure 30: Example of the digits dataset

a type of wine. The first class has 59 instances, the second one has 71 and the third one has 48. It does not have missing values.

- **Handwriting**: This dataset is based on digits handwriting. It has 60000 train instance and 10000 of test instances. Each instance has a vector of 784 elements which represents a 28x28 matrix where each element is a pixel in grayscale ranged from 0 to 256. There is also a column for the labels numbered from 0 to 9. Figure 30 shows an example of 10 instances. It does not have missing values. In this work only 6000 instances of this dataset have been analysed because the Similarity Graph generated by the Spectral Clustering algorithm is bigger than the memory available [2].

4.5.2 Preprocessing and Normalizing the Data

This first analysis prepares the datasets for the clustering analysis. The preprocessing process is divided in two principal steps:

- The first step has been the study of the available variables through histograms and correlation diagrams which were used for dimension

[2]The computer used has 4 Gbytes of RAM memory and 1 Gbytes of Virtual Memory, in the generation of the Similarity Graph it is necessary to generate a matrix of 6000×6000 of double values. If a double variable requires 8 bytes, then the whole matrix requires $6000 \times 6000 \times 8 \approx 288$ Mbytes. However, if the 60000 data instance are used, the memory required is $60000 \times 60000 \times 8 \approx 28.8$ Gbytes.

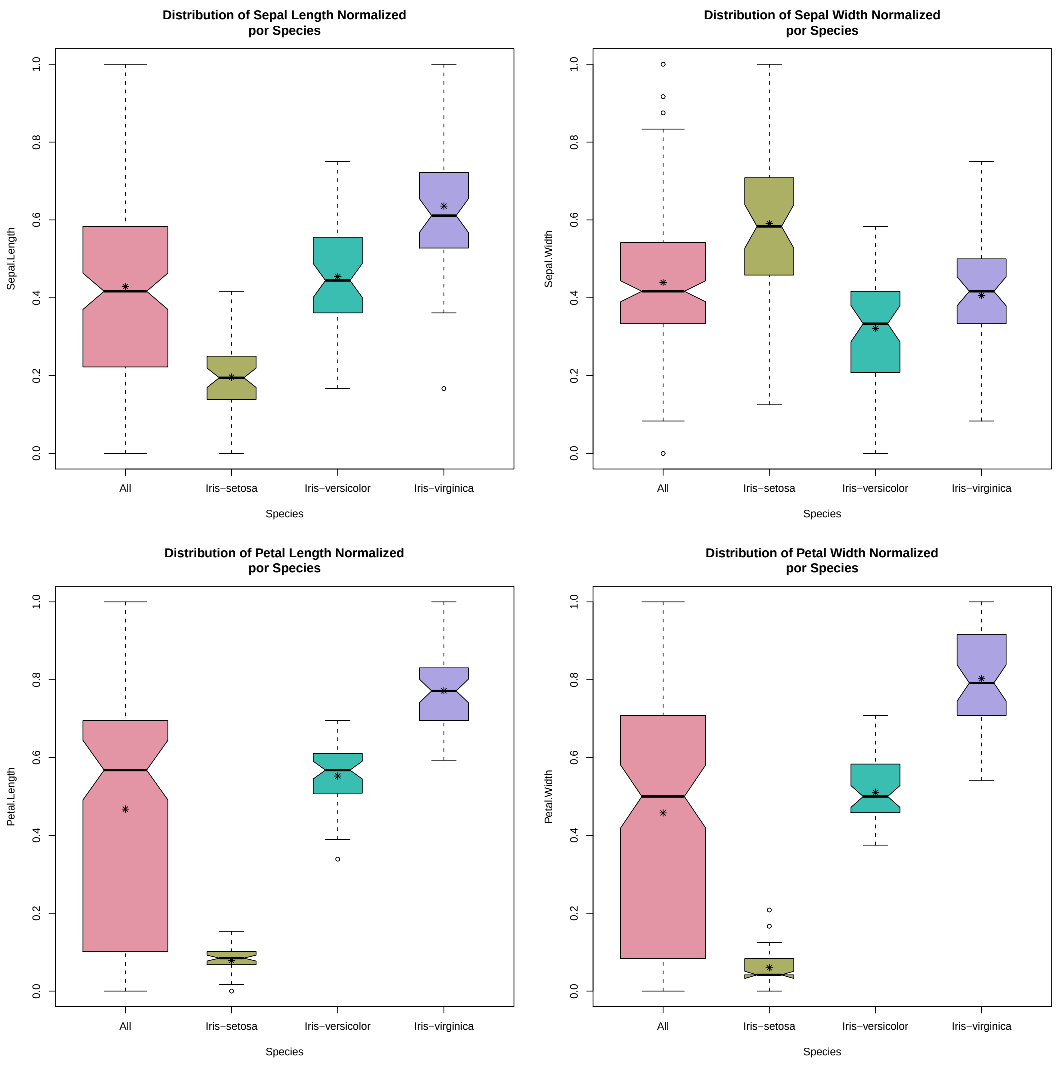


Figure 31: Boxplot of the Iris Dataset. Petal-length and Petal-width are the variables which better discriminates the three classes. Versicolour and Virginica classes are more difficult to discriminate than Setosa class. The number of outliers is low.

reduction. The information provided by this phase shows the values which are useless because, for example, are constants or have a high correlation (more than 0.8 if we consider that the correlation values is in range $[0, 1]$) with other variables. This means that they may variate the clustering results, if they are not eliminated, with redundant information.

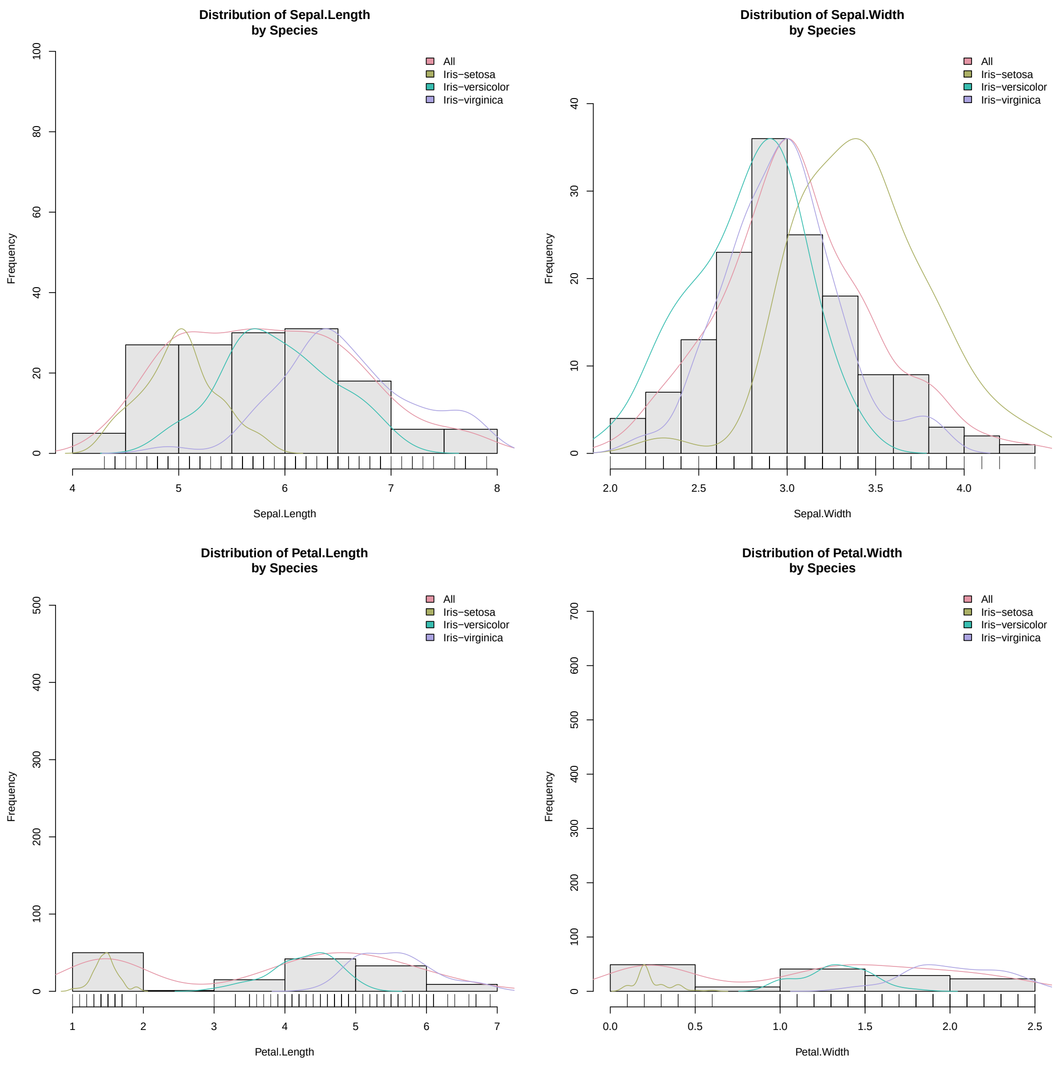


Figure 32: Histograms of the Iris Dataset. The three Gaussian functions which represent the classes distribution are also clearer separated for Patel-length and Petal-width variables. Versicolour and Virginica classes are too close together.

- The second preprocessing phase consists on the normalization of the variables. First, the attributes with outliers are recentralized. After, the same range is applied for all. As was described in section 2.2, we combine Z-score to recentralized the distribution and avoid outliers and MinMax to fixed the range of all the values between 0 and 1.

Dataset	Population	Generations	Crossover	Mutation	Selection	Max. Fitness
Iris	1000	2000	0.1	0.8	50	0.9946
Wine	100	20000	0.4	0.01	50	1
Handwriting	20	20000	0.9	0.2	5	0.8995

Table 3: Best parameter selection found of the GGC algorithm for the different real-world datasets and the fitness achieved by the GGC algorithm. The K value of the KNN-Minimal Cut fitness is always set to 2

Figures 31 and 32 show the boxplots and histograms of the Iris dataset for all its variables and classes. In the Iris and Wine datasets, there are a few number of instances and attributes, it implies that the reduction is not necessary. However, in the case of the handwriting dataset there are a lot of attributes (pixels) which do not contribute to the analysis (those pixels which have always the same value, for example). Also there are features which have a high correlation between them. These attributes have been reduced in the first step leaving 195 attributes for the analysis. All the attributes of the datasets have been normalized applying the techniques of the second step.

4.5.3 Experiment Results

The experiments have followed the same procedure that they followed with the synthetic datasets experiments (see Table 3 for the parameters selection). The value of σ has been approximated to 100. Table 4 shows the accuracy percentages of the different clustering algorithms. The results for the Iris show that EM is the best classifier (with an accuracy of the 96,67 %) and the GGC algorithm is the second (92%). The results for the Wine datasets show that all the algorithm obtained high accuracy values (bigger than the 95 %), and the GGC algorithm obtain a perfect classification with the maximum fitness value (see Table 3). Finally, the results of the Handwriting show that Spectral Clustering and GGC obtain the best classification results (73,55% and 99%, respectively). These results are a consequence of the data distribution. Iris dataset has instances of different classes which are closed to each

	Iris	Wine	Handwriting
K-Means best classification	89.33%	95.50 %	50.83 %
EM best classification	96.67%	97.19%	35.43 %
Spectral Clustering best classification	89.33%	95.50%	73.55%
GGC best classification	92%	100%	99%

Table 4: Experimental results obtained using the UCI datasets

other, the GGC algorithm has problems to discriminate the boundary of the clusters specially when there are intersections between the clusters. The fitness value of the Iris is the highest that the algorithm has achieved, it shows that there are instance which belongs to different cluster but are closed to each other. In the case of the Wine dataset, the classes are clearer separated (as the different clustering techniques show). It improves the results of the GGC algorithm, because the boundaries are clearer. It must be also similar in the Handwriting case, however, the value of the fitness shows that there are some instances which are in the clusters boundary and are difficult to assign to a determine cluster.

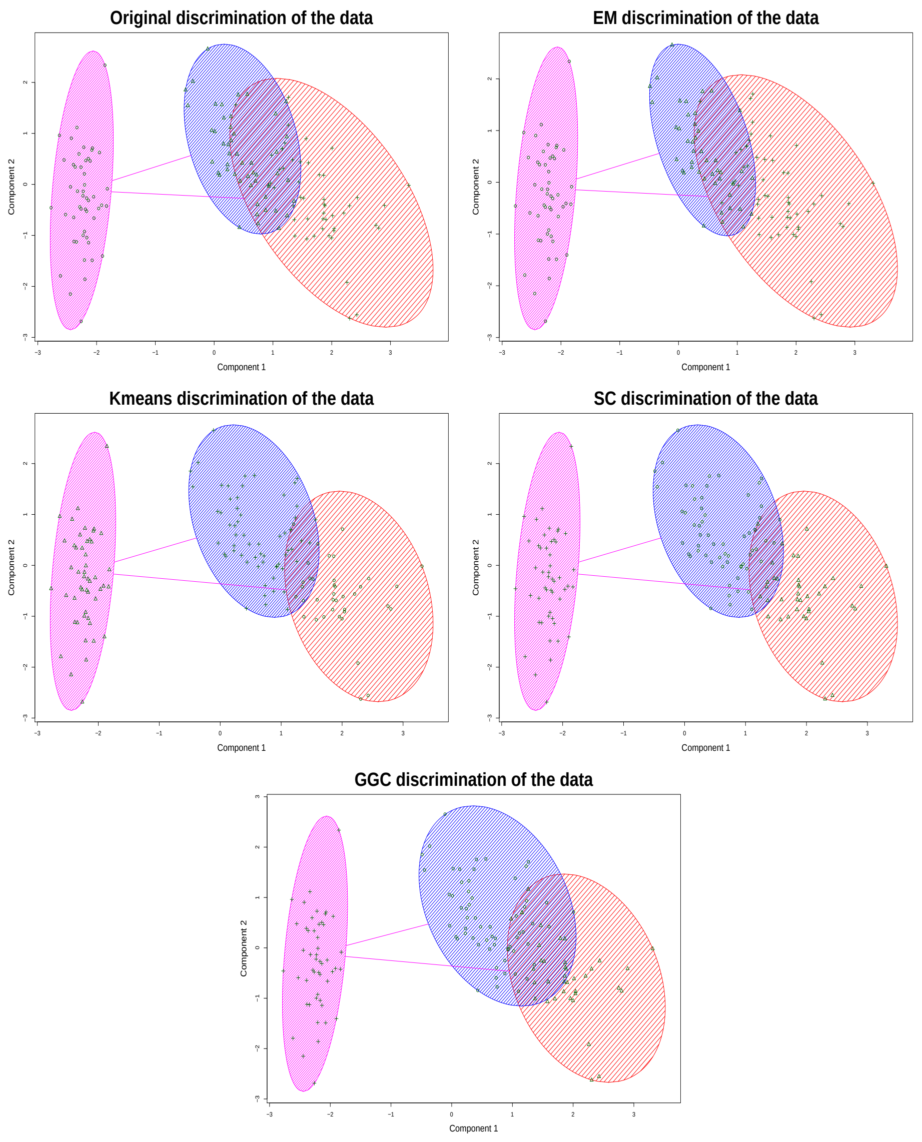


Figure 33: Discrimination of the data obtained by the different algorithms (from left to right and from top to bottom: The original Iris classification, the results of EM, the results of Kmeans, the results of Spectral Clustering and the results of the Genetic Algorithm) projected over 2 principal components which explained the 95.81 % of the point variability

5 Conclusions

This work presents a new clustering method inspired in the Spectral Clustering algorithm and based on Genetic Algorithm. The GCC algorithm is defined using simple codifications and traditional Genetic Algorithm operations. The main contribution of the algorithm is the fitness function. GGC is based on KNN and Minimum Cut measures. It is applied to the Similarity Graph which is generated in the first step of the Spectral Clustering algorithm. The combination of these measures improves the robustness of the algorithm giving a higher independence of the parameters of the Similarity Function. The results of Section 4 (Experimentation) show that the new algorithm obtains good results for both, synthetic and real-world datasets. However there are several problems which will be studied in a future work:

- Large datasets, such as the whole Handwriting dataset, are problematic because the Similarity Graph generated is computationally high. This problem is inherit from the Spectral Clustering algorithm. It is necessary to find other techniques to keep t he information of the Similarity Graph.
- When the number of clusters is high, the algorithm converges slower to the solution. It is also a problem which should be studied.
- When the boundary of the clusters is not clear, the algorithm also has problems to define this section.

The future work will be focused on several improvements that could be made to the GGC algorithm. The effects of noisy information could be deeply analysed. The number of clusters could be automatically selected using strategies such as cross-validation. Finally, other fitness functions which could improve the convergence, and the clusters quality, of the GGC algorithm will be studied.

6 Contributions

During the development of these work the following contributions have been generated:

- G. Jiménez-Díaz, H. D. Menéndez, D. Camacho and P. A. González-Calero. **Predicting performance in team games**. In I. I. for systems, C. Technologies of Information, and Communication, *editors, ICAART 2011 - Proceedings of the 3ed International Conference on Agents and Artificial Intelligence*, volume Vol 1, pages pages 401 - 406, 2011.

- G. Bello, H. Menéndez and D. Camacho. **Using the clustering coefficient to guide a genetic-based communities finding algorithm**. In H. Yin, W. Wang, and V. Rayward-Smith, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2011*, volume 6936 of Lecture Notes in Computer Science, pages 160-169. Springer Berlin / Heidelberg, 2011.

- Héctor Menéndez, Gema Bello Orgaz and David Camacho. **Features Selection from High-Dimensional Web Data using Clustering Analysis**. *In Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS '12* (accepted).

- Héctor Menéndez, Gema Bello Orgaz and David Camacho.**Extracting Behavioural Models from 2010 FIFA World Cup**. *Journal of Systems Science and Complexity, JSSC 2012* (sent).

- H. Menéndez and D. Camacho. **A Genetic Graph-based Clustering Algorithm**. In H. Yin, W. Wang, and V. Rayward-Smith, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, (conditional accepted).

References

[1] K. Adamska. Cluster analysis of genetic algorithms results. *Inteligencia Artificial, Revista Iberoamericana de IA*, 9(28):25–32, 2005.

[2] J. Aguilar. Resolution of the clustering problem using genetic algorithms. *International Journal of Computers*, 1(4):237 – 244, 2007.

[3] A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering*, 63(2):503 – 527, 2007.

[4] F. Bach and M. Jordan. Learning Spectral Clustering, With Application To Speech Separation. *Journal of Machine Learning Research*, 7:1963 – 2001, Oct. 2006.

[5] W. Barbakh and C. Fyfe. Clustering with reinforcement learning. In H. Yin, P. Tino, E. Corchado, W. Byrne, and X. Yao, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, volume 4881 of *Lecture Notes in Computer Science*, pages 507–516. Springer Berlin / Heidelberg, 2007.

[6] W. Barbakh and C. Fyfe. Online clustering algorithms. *International Journal of Neural Systems*, 18(3):185–194, 2008.

[7] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, March 2004.

[8] G. Bello, H. Menéndez, and D. Camacho. Using the clustering coefficient to guide a genetic-based communities finding algorithm. In H. Yin, W. Wang, and V. Rayward-Smith, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2011*, volume 6936 of *Lecture Notes in Computer Science*, pages 160–169. Springer Berlin / Heidelberg, 2011.

[9] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing (The Handbooks of Fuzzy Sets)*. Springer, 1 edition, Mar. 2005.

[10] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97:245–271, December 1997.

[11] O. Bousquet, U. von Luxburg, and G. Rätsch, editors. *Advanced Lectures on Machine Learning, ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, volume 3176 of *Lecture Notes in Computer Science*. Springer, 2004.

[12] S. R. Carroll and D. J. Carroll. *Statistics Made Simple for School Leaders*. Rowman & Littlefield, 2002.

[13] H. Chang and D.-Y. Yeung. Robust path-based spectral clustering. *Pattern Recogn.*, 41(1):191–203, Jan. 2008.

[14] K. J. Cios, R. W. Swiniarski, W. Pedrycz, L. A. Kurgan, K. J. Cios, R. W. Swiniarski, W. Pedrycz, and L. A. Kurgan. Unsupervised learning: Clustering. In *Data Mining*, pages 257–288. Springer US, 2007.

[15] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111+, Dec. 2004.

[16] R. M. Cole. Clustering with Genetic Algorithms. Master's thesis, Nedlands 6907, Australia, 1998.

[17] Coley. *An Introduction to Genetic Algorithms for scientists and engineers*. World Scientific Publishing, 1999.

[18] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21 –27, january 1967.

[19] L. Curiel, B. Baruque, C. Dueñas, E. Corchado, and C. Pérez-Tárrago. Genetic algorithms to simplify prognosis of endocarditis. In *Proceedings of the 12th international conference on Intelligent data engineering and automated learning*, IDEAL'11, pages 454–462, Berlin, Heidelberg, 2011. Springer-Verlag.

[20] S. Das, A. Abraham, and A. Konar. Automatic clustering using an improved differential evolution algorithm. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(1):218 –237, jan. 2008.

[21] M. Dehmer, editor. *Structural Analysis of Complex Networks*. Birkhäuser Publishing, 2010. in press.

[22] K. Delac, M. Grgic, and S. Grgic. Independent comparative study of PCA, ICA, and LDA on the FERET data set. *International Journal of Imaging Systems and Technology*, 15(5):252–260, 2005.

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[24] D. Doval, S. Mancoridis, and B. S. Mitchell. Automatic Clustering of Software Systems using a Genetic Algorithm. In *IEEE Proceedings of the 1999 Int. Conf. on Software Tools and Engineering Practice (STEP'99)*, pages 73–91, 1999.

[25] V. Fernandez, R. G. Martinez, R. Gonzalez, and L. Rodriguez. Genetic algorithms applied to clustering. In *In Proceedings of the Winter Simulation Conference*, pages 1307–1314, 1997.

[26] S. Fortunato, V. Latora, and M. Marchiori. Method to find community structures based on information centrality. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(5):056104, 2004.

[27] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[28] A. A. Freitas. A review of evolutionary algorithms for data mining. In *In: Soft Computing for Knowledge Discovery and Data Mining*, pages 61–93, 2007.

[29] L. Fu and E. Medico. Flame, a novel fuzzy clustering method for the analysis of dna microarray data. *BMC Bioinformatics*, 8, 2007.

[30] C. Fyfe. Topographic maps for clustering and data visualization. In J. Fulcher and L. Jain, editors, *Computational Intelligence: A Compendium*, volume 115 of *Studies in Computational Intelligence*, pages 111–153. Springer Berlin - Heidelberg, 2008.

[31] C. Fyfe and W. Barbakh. Immediate reward reinforcement learning for clustering and topology preserving mappings. In M. Biehl, B. Hammer, M. Verleysen, and T. Villmann, editors, *Similarity-Based Clustering*, volume 5400 of *Lecture Notes in Computer Science*, pages 35–51. Springer Berlin - Heidelberg, 2009.

[32] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007.

[33] A. N. Gorban and A. Zinovyev. Principal manifolds and graphs in practice: From molecular biology to dynamical systems. *International Journal of Neural Systems*, 20(3):219 – 232, 2010.

[34] M. R. Gupta and Y. Chen. Theory and use of the em algorithm. *Foundations and Trends in Signal Processing*, 4(3):223–296, 2010.

[35] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[36] J. Handl, J. H, and J. Knowles. Evolutionary multiobjective clustering. In *In Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pages 1081–1091. Springer, 2004.

[37] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4–6):175–181, 2000.

[38] 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*, 39(2):133 – 155, march 2009.

[39] C. Huttenhower, A. Flamholz, J. Landis, S. Sahi, C. Myers, K. Olszewski, M. Hibbs, N. Siemers, O. Troyanskaya, and H. Coller. Nearest Neighbor Networks: clustering expression data based on gene neighborhoods. *BMC Bioinformatics*, 8(1):250+, 2007.

[40] A. Jain and M. Law. Data clustering: A user's dilemma. In S. Pal, S. Bandyopadhyay, and S. Biswas, editors, *Pattern Recognition and Machine Intelligence*, volume 3776 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin / Heidelberg, 2005.

[41] G. Jiménez-Díaz, H. D. Menéndez, D. Camacho, and P. A. González-Calero. Predicting performance in team games. In I. I. for systems, C. Technologies of Information, and Communication, editors, *ICAART 2011 - Proceedings of the 3ed International Conference on Agents and Artificial Intelligence*, volume Vol 1, pages pages 401 – 406, 2011.

[42] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

[43] K. Kim, R. B. McKay, and B.-R. Moon. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of*

the 12th annual conference on Genetic and evolutionary computation, GECCO '10, pages 1179–1186, New York, NY, USA, 2010. ACM.

[44] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[45] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, December 1997.

[46] K. Krishna and M. N. Murty. Genetic K-means Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 29(3):433–439, 1999.

[47] G. N. Lance and W. T. Williams. A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. *The Computer Journal*, 9(4):373–380, Feb. 1967.

[48] W. Langdon and R. Poli. Evolving problems to learn about particle swarm and other optimisers. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 81 –88 Vol.1, sept. 2005.

[49] D. T. Larose. *Discovering Knowledge in Data.* John Wiley & Sons, 2005.

[50] M. Lipczak and E. Milios. Agglomerative genetic algorithm for clustering in social networks. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1243–1250, New York, NY, USA, 2009. ACM.

[51] D. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

[52] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[53] U. Maulik. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.

[54] H. Menendez, G. Bello-Orgaz, and D. Camacho. Extracting behavioural models from 2010 fifa world cup. In *Journal of Systems Science and Complexity*, JSSC 2012, page (Accepted, 2012.

[55] H. Menendez, G. Bello-Orgaz, and D. Camacho. Features selection from high-dimensional web data using clustering analysis. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, page (Accepted, New York, NY, USA, 2012. ACM.

[56] O. d. Merle, P. Hansen, B. Jaumard, and N. Mladenovic. An interior point algorithm for minimum sum-of-squares clustering. *SIAM J. Sci. Comput.*, 21(4):1485–1505, Dec. 1999.

[57] B. Nadler, S. Lafon, R. Coifman, and I. G. Kevrekidis. Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators. pages 955 – 962, 2005.

[58] M. C. V. Nascimento and A. C. P. L. F. Carvalho. A graph clustering algorithm based on a clustering coefficient for weighted graphs. *J. Braz. Comp. Soc.*, 17(1):19–29, 2011.

[59] G. Nathiya, S. C. Punitha, and M. Punithavalli. An analytical study on behavior of clusters using k means, em and k* means algorithm. *CoRR*, abs/1004.1743, 2010.

[60] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.

[61] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, pages 849–856. MIT Press, 2001.

[62] P. Pokorný and P. Dostál. Cluster analysis and genetic algorithms. In *In: Management, Economics and Business Development in the New European Conditions*, pages 1–9, 2008.

[63] R. Poli and W. B. Langdon. Backward-chaining evolutionary algorithms. *Artificial Intelligence*, 170(11):953 – 982, 2006.

[64] P. Pons and M. Latapy. Computing communities in large networks using random walks (long version). *ArXiv Physics e-prints*, Dec. 2005.

[65] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006.

[66] V. Roth and T. Lange. Feature selection in clustering problems. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[67] D. K. Roy and L. K. sharma. Genetic kmeans clustering algorithm for mixed numeric and categorial data sets. *International Journal of Artificial intelligence and Applications(IJAIA)*, 1(2):23 – 28, 2010.

[68] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[69] L.-D. Shi, Y.-H. Shi, Y. Gao, L. Shang, and Y.-B. Yang. Xcsc:: A novel approach to clustering with extended classifier system. *International Journal of Neural Systems*, 21(1):79 – 93, 2011.

[70] M. Srinivas and L. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656 –667, apr 1994.

[71] L. Y. Tseng and S. B. Yang. A genetic approach to the automatic clustering problem. *Pattern Recognition*, 34(2):415 – 424, 2001.

[72] A. Tsonis, K. Swanson, and G. Wang. Estimating the clustering coefficient in scale-free networks on lattices with local spatial correlation structure. *Physica A: Statistical Mechanics and its Applications*, 387(21):5287–5294, 2008.

[73] C. Veenman, M. Reinders, and E. Backer. A maximum variance cluster algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(9):1273 – 1280, sep 2002.

[74] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec. 2007.

[75] U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586, Apr. 2008.

[76] H. WANG, J. CHEN, and K. GUO. A genetic spectral clustering algorithm. *Journal of Computational Information Systems*, 7(9):3245–3252, 2011.

[77] L. Wang, M. Jiang, Y. Lu, M. Sun, and F. Noe. A comparative study of clustering methods for molecular data. *International Journal of Neural Systems*, 17(6):447 – 458, 2007.

[78] D. J. Watts. *Small worlds : the dynamics of networks between order and randomness.* 1999.

[79] Wojciech and Kwedlo. A clustering method combining differential evolution with the k-means algorithm. *Pattern Recognition Letters*, 32(12):1613 – 1621, 2011.

[80] R. Xu and D. Wunsch. *Clustering (IEEE Press Series on Computational Intelligence)*. Wiley-IEEE Press, illustrated edition edition, Oct. 2008.

[81] C. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, C-20(1):68 – 86, jan. 1971.