International Journal of Neural Systems, Vol. 0, No. 0 (April, 2000) 00–00

A GENETIC GRAPH-BASED APPROACH FOR PARTITIONAL CLUSTERING

Héctor D. Menéndez
Computer Science Department, Universidad Autónoma de Madrid
28049, Madrid, Spain
E-mail: hector.menendez@uam.es

David F. Barrero
Departamento de Automática, Universidad de Alcalá
28801, Alcalá de Henares, Madrid, Spain
E-mail: david@aut.uah.es

David Camacho*
Computer Science Department, Universidad Autónoma de Madrid
28049, Madrid, Spain
E-mail: david.camacho@uam.es

Received (to be inserted
Revised by Publisher)

Abstract

Clustering is one of the most versatile tools for data analysis. In the recent years, clustering that seeks the continuity of data (in opposition to classical centroid-based approaches) has attracted an increasing research interest. It is a challenging problem with a remarkable practical interest. The most popular continuity clustering method is the Spectral Clustering algorithm, which is based on graph cut: It initially generates a Similarity Graph using a distance measure and then studies its Graph Spectrum to find the best cut. This approach is sensitive to the parameters of the metric, and a correct parameter choice is critical to the quality of the cluster. This work proposes a new algorithm, inspired by Spectral Clustering, that reduces the parameter dependency while maintaining the quality of the solution. The new algorithm, named Genetic Graph-based Clustering (GGC), takes an evolutionary approach introducing a Genetic Algorithm to cluster the Similarity Graph. The experimental validation shows that GGC increases robustness of Spectral Clustering and has competitive performance in comparison with classical clustering methods, at least, in the synthetic and real dataset used in the experiments.
Keywords: Machine Learning, Clustering, Spectral Clustering, Graph Clustering, Genetic Algorithms.

1. Introduction

Classical clustering algorithms like K-means [51] or Expectation-Maximization (EM) [24] estimate a set of parameters to build a data model [16]; other algorithms do not construct such model. A well-known algorithm that belongs to the latter is Spectral Clustering (SC) [69]; this algorithm seeks the continuity of the data, instead of the centroids. This characteristic makes SC well suited for a variety of relevant problems, such as pattern recognition and artificial vision [69].

In few words, SC [57] first builds a Similarity Graph based on distance measures (or Similarity Function). Then, the algorithm computes the

*Corresponding author.

eigenvectors of the Laplacian Matrix (Spectrum) extracted from the Similarity Graph and finally they are clustered using a classical algorithm such as K-means. This approach has been quite successful, and the algorithm is widely used. However, SC has some practical problems, probably the main problem is related to its robustness, it highly depends on the parameters chosen in the Similarity Function. This dependence generates a collection of undesirable effects that has a negative impact on the algorithm performance, for example, when the dataset is noisy [13].

Several authors have proposed solutions to the robustness problem. Some solutions aim to optimize the parameter setting of the Similarity Function [13]; other solutions focus on the selection of the clustering algorithm which groups the data projected through the eigenvectors in SC [72]. In this paper we introduce a new solution to this problem that consists of modifying the SC algorithm to eliminate the dependence on the distance definition. Then we propose a new algorithm inspired by SC named Genetic Graph-based Clustering (GGC). Our proposal uses a Genetic Algorithm (GA) [2,40] and is based on Graph Theory. Similarly to SC, GGC constructs a Similarity Graph according to a Similarity Function but, instead of computing its Laplacian Matrix, it looks for the clusters in the Similarity Graph using a GA.

Two essential elements in any GA design are the fitness function and the encoding of the individuals. The first one is used to guide the evolutionary search [19,66], the second one determines the search space and landscape [62]. In GGC the fitness function is a metric that measures the quality of the clusters and it is based on Complex System Analysis and Graph Theory [23,55,74].

Given a network which is represented as a graph, previous techniques analyse the different properties of the network through various measures. The principal measures are the Clustering Coefficient [23] (CC) and the Average Distance (AD) of the elements. There are several variations of them, such as the Weighted Clustering Coefficient [9] which considers weights in the edges of the graph. In order to support a good election for the fitness function, two measures are analysed in detail.

One of the main problems of AD or other Complex Network measures is that they do 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, an object form into an image. Therefore, other different fitness functions based on well-known algorithms (such as K-Nearest Neighbours [20] or Minimal Graph Cut [64]) have also been tested and combined to improve the results.

As it was mentioned above, the second critical element in the design of a GA is the encoding. In this particular case of clustering, the encoding has a convergence problem. The literature has addressed this issue and there are many approximations to the encoding of clustering in GA, (see [38] for a complete analysis of this problem). Given the importance of this topic to the success of GGC, we study two possible encodings, comparing their main features and subsequent performance.

In order to assess the performance of GGC, we have carried out a collection of experiments that compare GGC performance and robustness to those of some classical algorithms, like K-means, EM and, of course, SC.

The rest of the work is structured as follows. It first introduces in detail SC and surveys the state of art in GAs and Graph Theory for clustering. Then, Section 3 describes the GGC algorithms and the two fitness functions and encodings selected to be studied in detail. An analysis of GGC follows in Section 4, which includes some experimental work and theoretical results. Section 5 assesses GGC performance comparing it to K-means, EM and SC with synthetic and real datasets. Finally, Section 6 outlines some conclusions and future lines of work.

2. Related Work

This section starts with a general introduction of the clustering methods, specially SC. Once the clustering methods have been introduced, it focusses the attention on how GAs have been applied to clustering techniques. Finally, some measures and metrics based on Graph Theory and Complex Networks are introduced and defined.

2.1. *Clustering Algorithms*

Clustering is frequently used in Data Mining

and Machine Learning. The most popular clustering technique is K-means. Given a fixed number of clusters, K-means tries to find a division of the dataset [5,51] based on a set of common features given by distances (or metrics) that are used to determine how the cluster could be defined.

Other approximations, such as EM [24], 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 [24].

The most recent approaches combine classification techniques with clustering algorithms to improve the results quality, for example, Hsu uses Neural Networks applied to brain-computer interface systems [39], Kodogiannis et al. use Neural Networks and fuzzy clustering for short-term load forecasting [45] and Davis et al. combine segmentation and classification for hand radiography [22].

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 [8]. These methods create the clusters by adding a new instance at each step and modifying the cluster structure with this new information. Some other improvements of K-means algorithm are related to deal with different kinds of data representation, for example, mixed numerical data [4] and categorical data [63] . There are also some studies comparing methods, for example, Wang et al. [73] compare self-organizing maps, hierarchical clustering and competitive learning when establishing molecular data models of large size sets.

2.2. *Graph-based and Spectral Clustering*

Other clustering techniques are related to Graph Clustering. A well-known algorithm is SC, which is based on a straightforward interpretation of weighted undirected graphs as can be seen in [7,54,57,69]. SC starts building a Similarity Graph through a Similarity Function applied to the data instances. The Similarity Graph can be formulated in three different ways[69]:

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 non-zero similarity are connected with each other.

This work takes the most common approach in the literature, which is the fully connected graph with the Similarity Function named Radial Basis Function (RBF) Kernel [32]. RBF is defined by

$$s(x_i, x_j) = e^{-\sigma||x_i - x_j||^2} \tag{1}$$

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

It is important to remark that the Similarity Function parameters are highly connected with the SC results. Small changes in these parameters produce high changes in the solution reducing the algorithm robustness (this is deeply studied in Section 4.3). Chang and Yeung exposed these problems in [13].

Once the Similarity Graph is constructed, the second step of the algorithm is the extraction of its Spectrum or Laplacian Matrix. There are different definitions of the Laplacian Matrix that affect the performance of SC. Let I be the identity matrix and D the diagonal matrix whose (i,i)-element is the sum of the Similarity Graph (represented as a weighted Matrix) ith row, and let W be the Similarity Graph, then we can define the following three Laplacian Matrices [69]:

- **Unnormalized Spectral Clustering**. It defines the Laplacian Matrix as

$$L = D - W \tag{2}$$

- **Normalized Spectral Clustering**. It defines the Laplacian Matrix as

$$L_{sym} = D^{-1/2} L D^{-1/2} \tag{3}$$

- **Normalized Spectral Clustering (related to Random Walks [69])**. It defines the Laplacian Matrix as

$$L_{rw} = D^{-1} L \tag{4}$$

The Laplacian Matrix or SC algorithm used in this work is the Normalized Spectral Clustering Algorithm, which is the most classical technique

in the literature [69]. The three Laplacian Matrices have been deeply studied in the related literature [69,54,70]. They are connected to the *graph cut problem*, which looks for the best way to cut a graph keeping a high connectivity amongst the elements which belongs to each partition, and a low connectivity between the elements of different partitions.

The graph cut problem is closely related to clustering. In the graph cut literature this problem has two classical solutions[69]: RadioCut and NCut. Von Luxburg et al. [69] describe the connection between the different approaches of SC (focused on the Laplacian Matrices), RadioCut and NCut. They also show that Unnormalized Spectral Clustering converges to RadioCut and the Normalized methods converge to NCut. On the other hand, a deep analysis about the theoretical effectiveness of Normalized clustering over Unnormalized can be found in [70].

Finally, in the third step, the eigenvectors of the Laplacian Matrix are considered as data points and a clustering algorithm, such as K-means, is applied over them to define the clusters. The main problem is how to compute the eigenvectors and the eigenvalues of the Laplacian Matrix of the Similarity Graph avoiding the huge memory that it consumes. For example, when large datasets are analysed, the Similarity Graph of the SC algorithm requires a high memory storage and it makes extremely hard the eigenvalues and eigenvectors computation (using a dataset with 100,000 instances, the fully-connected Similarity Graph will use a matrix of 10^{10} elements and 8 bytes per element; the total size is almost 80Gb).

New Spectral Clustering methods are focused on some practical improvements. These refinements have been centred on big data processing such as, for example, distribute the algorithm execution [15] or real-time processing of data streams [14].

Our work is inspired by SC because our approach calculates a Similarity Graph, but in our case we use a different search algorithm, such as a GA, and borrow concepts from Graph Theory and Complex Networks analysis to find the clusters, instead of the Laplacian Matrix extracted from the Similarity Graph.

2.3. *Genetic Algorithms for Clustering*

GAs have been traditionally used in optimization problems [12,30,60], but given their extraordinary flexibility, GAs are used to solve a wide range of problems in many domains; clustering is a good example. They can be tuned in many ways, some examples can be found in [59] , where the algorithm is improved through backward-chaining, creating and evaluating individuals recursively reducing the computational time. Other applications of GAs in clustering are swarm systems [47], software systems [25], file clustering [27] and task optimization [58] or information extraction [30], among others.

Cole [18] shows different approaches of the genetic clustering problem, especially focused on the encoding and clustering operations. Hruschka et al. [38] provide a deep revision with 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 [3] modifies the fitness function considering cluster asymmetry, coverage and specific information of the studied case; Tseng and Yang [67] use a compact spherical cluster structure and a heuristic strategy to find the optimal number of clusters; Maulik and Bandyopadhyay [52] use the clustering algorithm for metric optimization trying to improve the cluster centre positions; Shi et al. [65] base the search for the clusters in their Extend Classifier Systems (a kind of Learning Classifier System), in which the fitness is determined by the measure of its prediction accuracy; Das and Abraham [21] use Differential Evolution.

Some of the methods previously described are based on K-means. For instance, Krishna and Murty [46] replace the crossover of the algorithm using K-means as a search operator and Wojciech and Kwedlo [75] also use Differential Evolution combined with K-means. Finally, Adamska [1] introduces other general results of evolutionary approaches to clustering. There are also other complete studies for multi-objective clustering developed by Handl et al. [36] and for Nearest Neighbour Networks by Huttenhower et al. [41].

In this work, we have used different encodings and fitness functions to look for new methods and algorithms in the domain of graph-based clustering

problems [53].

2.4. *Graph Theory and Clustering*

Graph theory has made significant contributions to data analysis, especially over the last few years, with its application to manifold reconstruction [6,34] using data distance and graph representation to create a structure which can be considered as an Euclidean space (or manifold).

Graph models are useful to represent a large number of problems in different domains. They have become especially popular over the last few years, being widely applied in Social Networks analysis. Graph models can be naturally used in these domains, where each node or vertex can be used to represent a network element, 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 the structure, behaviour, stability or even community evolution inside the graph [23,28,55,74].

Schaeffer[64] describes a complete roadmap of graph clustering, including a comparison of the three types of graphs: weighted, directed and undirected. The methods that Schaeffer compares are cutting, spectral analysis and degree connectivity amongst others. An exhaustive analysis of connectivity methods can be found in Hartuv and Shamir[37].

In network analysis, it is common to represent graphs, especially in the study of social networks, 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 [23,55,74].

The present work is closer 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 [10], 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[61], where different statistical mechanics for community detection are used. However, we decided to use GAs because we are mainly 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, Newman and Girvan [56] provide another work that measures the quality of the communities with graph metrics. Clauset et al. [17] show metrics that can be used to find the structure of a community in very large networks. GAs have also been applied to find communities or clusters through Agglomerative Genetic Algorithms [49] and multi-objective Evolutionary Algorithms [44] amongst others.

In this work, our approach uses metrics from Graph Theory -the K-Nearest Neighbour and the MinCut metrics- and Complex Networks -the Weight Clustering Coefficient- to guide the heuristic-based search of the genetic algorithm through the fitness function.

3. The Genetic Graph-based Clustering Algorithm (GGC)

GGC is an algorithm motivated to avoid the strong dependence between SC and its metric parameters, and in particular the Similarity Function that generates the Similarity Graph. Even though GGC takes an evolutionary approximation to clustering and uses some concepts from Graph Theory, it is strongly inspired by SC. This section describes in detail GGC and presents the two encodings and fitness functions that were studied in order to design the algorithm.

The algorithm first initializes the number of clusters, like in SC and K-means. Our technique looks for the best sub-graphs of the Similarity Graph which might define a clear partition. The Similarity Graph is generated by a Similarity Function like in the SC algorithm. The population is a set of potential solutions (named partitions) which evolve until a good enough solution is found, or a maximum number of generations is reached. The fitness function is a metric used to assess the potential solutions. The algorithm will try to maximize the fitness value. In the following, we describe the evolutionary components of GGC.

3.1. *The GGC Encodings*

The GA has been constructed using two classical integer encodings, well known in cluster-based genetic algorithms [30]. The first encoding is a simple vector encoding (label-based) while the second one is based on set theory (medoid-based). These two encodings have been selected to compare their computational effort and performance to choose the best encoding for our algorithm (the experimental comparison is in Section 4.1).

3.1.1. *Label-based and Medoid-based Encodings*

We examined two encodings to choose the one with better performance. The first one follows the philosophy of what the literature named label-based [30] encoding. Each gene in the chromosome represent an x_i of the dataset, and its value indicates the cluster that it belongs to. This is a naïve encoding, genes contain an integer that identifies one cluster. The number of nodes in the graph determines the chromosome length. The Fig. 1 a) shows an example of this encoding scheme with a chromosome containing the partition drawn in Fig. 2.

The second encoding is based on sets. In this case the chromosome is divided in several variable-length chunks, each one associated to a cluster. The chunks contain the data instances (medoids [30]) which compose each cluster. Probably, it can be better understood looking at Fig. 1 b), which shows the partition of Fig. 2 using this encoding.

	nodes								
	1	2	3	4	5	6	7	8	9
Chr.	1	1	1	2	2	2	3	3	3

a) Label-based encoding.

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

b) Medoid-based encoding.

Fig. 1. Example of label-based and medoid-based encoding in GGC.

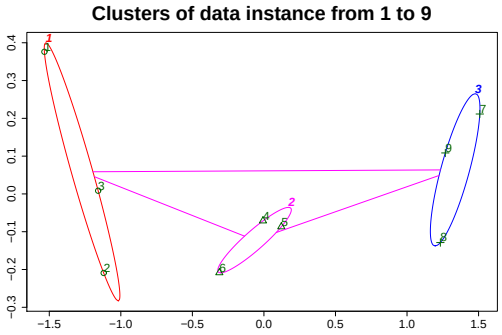


Fig. 2. Examples of the clusters considered in Fig. 1

3.1.2. *Invalid elements*

The genetic operations (mutation and crossover) of the GA might create invalid chromosomes. Using these encodings, it only happens when a chromosome contains one or more empty clusters. In partitional clustering, these solutions are invalid because the number of clusters is initially given, and therefore each cluster must contain at least one element. To avoid invalid chromosomes, the fitness value assigned to these chromosomes is 0. This value prevents that the elements passes to the next generation.

	nodes								
	1	2	3	4	5	6	7	8	9
Chr.	1	1	1	1	1	2	2	2	2

a) Label-based encoding.

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

b) Medoid-based encoding.

Fig. 3. Example of invalid chromosomes in GGC.

Some examples of invalid elements for each encoding are shown in Fig. 3. In this case, if $k = 3$ and $n = 9$, the first individual has missed cluster 3 (in both, label-based encoding and first chromosome for medoid-based encoding). In the medoid-based encoding, the second chromosome repeats

one element. In partitional clustering, all the clusters need to have at least one element and each element can only be assigned to one cluster.

3.2. *GGC Genetic Operators*

This section describes the genetic operators which are used between the chromosomes for each encoding. The classical operators (selection, crossover and mutation) have been used.

3.2.1. *Selection*

Regardless of the encoding used, the selection operator selects a subset of chromosomes to reproduce and breed the offspring. These chromosomes are selected using a tournament [71]. In few words, a tournament selects randomly n chromosomes, assesses them using the fitness function, and then takes the fittest one. It is called a $(\mu + \lambda)$ selection, where μ represents the number of bred chromosomes, and λ the new chromosomes generated.

3.2.2. *Crossover*

Any of the two encoding schemes induces a phenotypic space smaller than the genotypic space, and therefore different genotypes correspond to the same phenotype (see Fig. 4). This is a problem from the perspective of the recombination operator, because it destroys the correlation between phenotypic and genotypic spaces [62]. For this 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 is focused on the convergence improvement of the algorithm by reducing the search space and the number of invalid elements. To this end we define the following similarity measure.

Definition 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:*

$$s(C_i, C_j) = \frac{1}{2}\left(\frac{\sum_q \delta^q_{C_i}\delta^q_{C_j}}{|C_i|} + \frac{\sum_q \delta^q_{C_i}\delta^q_{C_j}}{|C_j|}\right) \quad (5)$$

where $|C_i|$ is the number of elements of cluster C_i and $\delta^q_{C_i}$ is the Kronecker δ defined by:

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

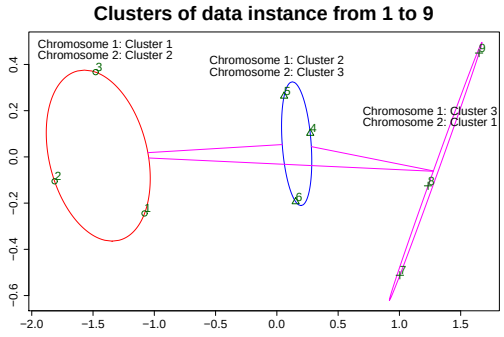
nodes

	1	2	3	4	5	6	7	8	9
Chr. 1	1	1	1	2	2	2	3	3	3
Chr. 2	2	2	2	3	3	3	1	1	1

a) Label-based encoding

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

b) Medoid-based encoding



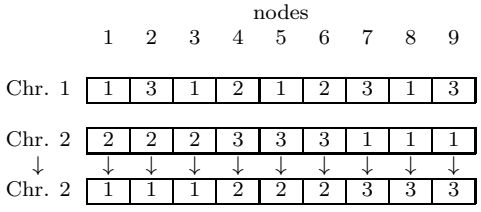
c) Original Representation

Fig. 4. These two chromosomes represent the same solution (for both kind of encodings), but the name of the clusters appears different using the label-based encoding.

The *relabelling process* can be divided in three fundamental steps:

1. The similarities between the clusters are calculated, using Equation (5).
2. The similarities are sorted using a decremental order.
3. The second chromosome is relabelled maximizing the similarity with the first chromosome.

Fig. 5 shows an example of two chromosomes, which represent the same solution, before the relabelling process for each encoding and the result of this process.

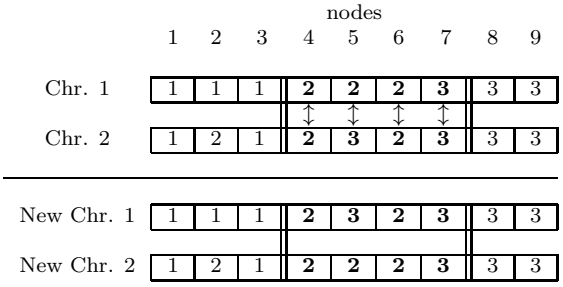


a) Label-based encoding.

	Cluster 1	Cluster 2	Cluster 3
Chr. 1	$\{1,3,5,8\}$	$\{4,6\}$	$\{2,7,9\}$
Chr. 2	$\{7,8,9\}$	$\{1,2,3\}$	$\{4,5,6\}$
↓ Chr. 2	$\{1,2,3\}$	$\{4,5,6\}$	$\{7,8,9\}$

b) Medoid-based encoding.

Fig. 5. Example of the relabelling process applied to two chromosomes with the label-based and medoid-based encodings.



a) Crossover for Label-based encoding.

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

b) Crossover for Medoid-based encoding.

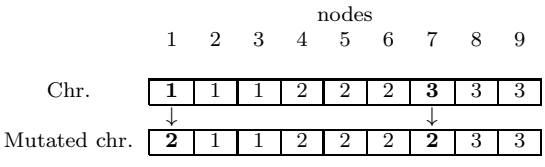
Fig. 6. Crossover using the label-based and medoid-based encodings after relabelling process.

The crossover of the label-based encoding exchanges strings of numbers between two chromosomes. It is straightforward since both strings have the same length (see Fig. 6 a)). In the medoid-based encoding, it keeps the similar elements of both chromosomes and the different elements are randomly distributed amongst the clusters, creating two new elements (see Fig. 6 b)).

3.2.3. *Mutation*

GGC uses an adaptive mutation for both encodings that works as follows:

1. A chromosome is randomly chosen to be mutated according to a mutation probability p_m, that is fixed at the beginning, with $p_m \in [0,1]$.
2. When a chromosome is chosen, the alleles which will be mutated are selected. The decision considers the probability of the allele to belong to the cluster which have been assigned. If the probability is high, the allele has a low probability of mutation and vice versa. In our algorithm, this probability depends on the metric defined in the fitness function. This means that even if the mutation probability is high and an allele is chosen to mutate, if the chromosome is close to the solution it could not mutate.
3. Finally, the alleles are mutated depending on the encoding:
   - The label-based encoding changes the allele value. The new value is a random number between 1 and the number of clusters.
   - The medoid-based encoding moves the allele to other cluster. It randomly chooses the new cluster which will contain the allele.



a) Label-based encoding.

	Cluster 1	Cluster 2	Cluster 3
Chr.	$\{1,2,\boldsymbol{3}\}$	$\{4,5,6\}$	$\{7,\boldsymbol{8},9\}$
Mutated chr.	$\{1,2\}$	$\{4,5,6,\boldsymbol{8}\}$	$\{7,9,\boldsymbol{3}\}$

b) Medoid-based encoding.

Fig. 7. Mutation of two alleles in a chromosome.

Fig. 7 shows the mutation process. In the label-based encoding, the first and seventh alleles have been randomly chosen to be changed. In the medoid-based encoding, the third and eighth alleles have been moved from first and third clusters to third and second respectively.

3.3. *The GGC Fitness Functions*

This section describes the two fitness functions designed in the context of GGC; these functions have been chosen to satisfy the continuity condition of the clusters. The first fitness is the Weight Clustering Coefficient [9] which looks for "strong triangles" formed between neighbours in the graph. The second is a combination of the K-Nearest Neighbour [48] and the Mincut methods [64].

3.3.1. *The Weighted Clustering Coefficient Fitness Function*

The first fitness function uses Global Weight Clustering Coefficient [9] as the fit value for the population. Supposing an undirected weight graph, it applies the following metric C_i^w, which is defined as:

$$C_i^w = \frac{\sum_{j,h} \frac{w_{ij}+w_{ih}}{2} a_{ij} a_{ih} a_{jh}}{S_i(k_i - 1)} \quad (6)$$

where w_{ij} are the weights of the matrix, a_{ij} is 1 if the edge from i to j exists and 0 otherwise, $S_i = \sum_j w_{ij}$ and k_i is the number of neighbours of the node i. The denominator $S_i(k_i - 1)$ defines a normalization factor to range the value between $[0, 1]$. 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*

The second fitness function under study is a combination of the classical K-Nearest Neighbourhood (KNN) [48] and the Minimal Cut[64] algorithms. KNN 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 belong to different clusters are not assigned to the same cluster. The K value for KNN is initially given by the user, nevertheless, in this work we have fixed it to 2 because it is the minimal value to guarantee the continuity, in a similar way than the Clustering Coefficient, and additionally it avoids over-fitting.

KNN covers all the nodes and checks if the K-closest elements (related to the metric) are in the same cluster. The fitness value of this measure is the mean of the percentage of well-classified neighbours of all the individuals in a cluster. The Minimal Cut measure calculates the average value edge weights which have been removed. The final value of the fitness is the product of the KNN metric and the subtraction between one and the Minimal Cut metric; both metrics have the same range: [0,1]. Therefore, the algorithm maximizes the value of $\frac{TotalKNN}{|C|} \times \left(1 - \frac{TotalMC}{|C|}\right)$ where:

$$TotalMC = \sum_{x \in C} \frac{\sum_{y \notin C_x} w_{xy}}{|\{y | y \notin C_x\}|} \quad (7)$$

$$TotalKNN = \sum_{x \in C} \frac{|\{y | y \in \Gamma(x) \wedge y \in C_x\}|}{|\Gamma(x)|} \quad (8)$$

In these formulas, w_{xy} represents the weight of edge $x \to y$, 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 improves the proximity of the neighbours.

3.4. *The Algorithm Steps*

The GGC algorithm can be divided in three main steps:

1. **Similarity Graph generation:** a Similarity Function (usually based on a kernel) is applied to the data instances (i.e., the domain concepts), connecting all the points with each other. It generates the Similarity Graph.
2. **Genetic search:** Giving an initial number of clusters k, the GA generates an initial population of possible solutions and evolves them using a fitness function to guide the algorithm to find the best solution. It stops when a good solution is found, or a maximum number of generations is reached.
3. **Clustering association:** The solution with the highest fitness value is chosen as a solution of the algorithm and the data instances

are assigned to the k clusters according to the solution chosen.

4. GGC Analysis

This section shows an analysis of the GGC algorithm, including the two encodings introduced in Section 3.1 and the metrics associated with the fitness functions previously described. Finally, the robustness of the GGC algorithm is evaluated against and compared to the robustness of the SC algorithm.

4.1. *Comparison of GGC Encodings*

The two encodings used in this work are equivalent and can be applied to any problem with similar results, however, they present the following differences:

- Omitting the relabelling process, the **label-based** crossover operation is faster than the medoid-based crossover. In the label-based case, the crossover is $O(n)$ because only one loop is necessary to swap the values of two vectors. For the medoid-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 **label-based** encoding is slightly faster because in the label-based encoding the value changes instantly when the mutation is applied, while in the medoid-based encoding the value is extracted from one set and introduced in another set and a swap process is needed.
- Both encodings can use the relabelling process, however the **medoid-based** encoding simplifies the similarity calculus using the intersection operation.
- GGC algorithm presents, as any other heuristic-based search method, a local maximum convergence problem. This problem has not been deeply studied in the GGC algorithm, however it depends on the GA operators. To compare both encodings convergence behaviour, the Spirals dataset [43] has been tested against them. Fig. 8 plots the convergence results for this dataset (for 50 runs of the algorithm per encoding and fitness function) using the parameters shown in Table 1, the KNN value set to 2 (as is explained in Section 3.3.2) and the tournament value also set to 2. The algorithm uses an adaptive mutation (see Section 3.2.3), the value (0.5) is the initial value for the mutation and (10^{-4}) the final value. In this case, the label-based encoding converges faster than the medoid-based encoding.

Dataset	**Pop.**	**Gen.**	**Cross.**	**Mut.**	**Eli.**	**Fit.**
Spirals	200	2000	0.3	0.5-10^{-4}	50	1.0

Table 1. Parameter setting with population, generations, crossover probability, mutation probability and elitism size used with the Spirals datasets. The table also includes the fitness value achieved.

The **label-based** encoding reduces the computation effort (see Table 2). Therefore, it has been chosen to carry out the rest of the experiments.

Encoding	Process			
	Cross.	Mut.	Relab	Convergence
Label-based	X	X		X
Medoid-based			X	

Table 2. Comparison for both encoding related to genetic operations in GGC. 'X' shows the encoding which achieves the best results with respect to computational effort and speed.

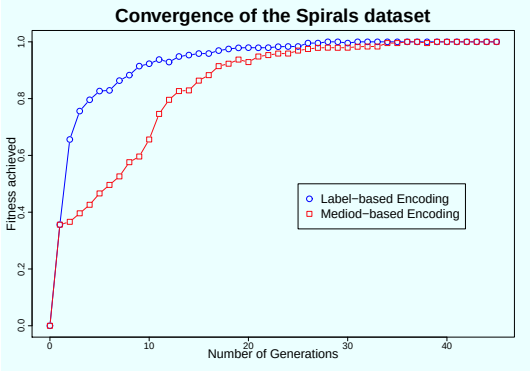


Fig. 8. GGC convergence for Spirals dataset. The convergence for the label-based encoding is reached in the 30 generation, using the medoid-based encoding is reached in the 40 generation.

4.2. *GGC Fitness Functions Analysis*

During the experimental stage **to test the behaviour** of our fitness functions, we detected that

the Weight Clustering Coefficient fitness obtained the maximum value even when the solution was incorrect. The analysis of this problem shows that it only happens when the Similarity Graph was fully connected (i.e., all the weights are bigger than 0). We analyzed this fact in more detail in an attempt to explain it, concluding that there is an issue with this approximation: It can be mathematically proved that this problem is a "metric mistake" [†]. The following theorem shows the proof:

Theorem 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 $C_i^w = 1\ \forall i$.*

Proof. We choose a random element i which has n neighbours. Let $x_1, \ldots, x_n$ be 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:

$$S_i = x_1 + \ldots + x_n$$

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:

$$\begin{array}{ccccccc} 0 & + & \frac{x_1+x_2}{2} & + & \ldots & + & \frac{x_1+x_n}{2} \\ \frac{x_2+x_1}{2} & + & 0 & + & \ldots & + & \frac{x_2+x_n}{2} \\ \vdots & + & \vdots & + & \ldots & + & \vdots \\ \frac{x_n+x_1}{2} & + & \ldots & + & \frac{x_n+x_{n-1}}{2} & + & 0 \end{array}$$

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

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

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:

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

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

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

□.

Since the Similarity Graph construction that was chosen is the fully connected graph (see Section 2.2), the only fitness that has been applied in the experiments is the KNN-Minimal Cut fitness to avoid this problem.The fully connected approximation was chosen because the GGC algorithm tries to maximize the robustness of the clustering selection related to the metrics, as is explained in the following subsection. Therefore, if the ϵ-neighbourhood graph or the k-nearest neighbour graph are chosen (see Section 2.2), 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 on 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.

4.3. *Robustness of the GGC algorithm*

An important problem related to SC is its dependency on the parameters of the Similarity Function. The GGC algorithm has been designed to alleviate this problem. The KNN metric which is applied in the fitness calculation provides a higher robustness to the algorithm compared to the SC algorithm, it does not depend on the order of distance magnitude calculated by the metric.

To compare the sensitivity of SC and GGC to the parameters of the metric, the RBF kernel has been used to carry out the experiments. This kernel is defined by: $K_{ij} = e^{-\sigma||x_i - x_j||^2}$, where K is the similarity matrix, x_i, x_j are data instances, and σ is the parameter which changes the order of magnitude. The experimental results show that SC clearly depends on σ parameter. Fig. 9 shows the different clustering results obtained using the

[†]This metric has also been used in several works about Weighted Complex Networks [9] and it is an important reference in the literature.

SC and the GGC algorithms modifying the σ parameter between 1 and 4000.

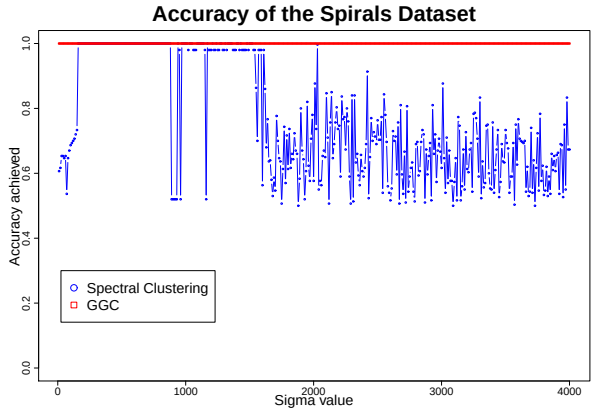


Fig. 9. SC and GGC results for the spirals[43] dataset with σ values from 1 to 4000, respectively. The red straight line in the top represents the robustness of GGC over SC.

These experimental results show that the parameters used in the definition of the kernel are critical (see the evolution of σ in Fig. 9) because these parameters define the degree of the similarity. Ng et al. introduced a method to calculate the optimal σ in [57], however, as Fig. 10 shows, this technique is not always enough. GGC always obtains the same results because it has been designed to be robust to the modification of the metric parameters, when this modification keeps the order relationship between the elements of the dataset and only changes the distance magnitude. The next section will show the experiments carried out using the GGC algorithm. The accuracy of the algorithm is tested using synthetic and real datasets.

5. Experimental Results

This section compares the GGC algorithm with other classical clustering algorithms (K-means, EM and SC) using synthetic and real datasets. The accuracy value is calculated using the similarity metric defined in Equation (5).

5.1. *Experiments on Synthetic Data*

Eight datasets have been extracted from the state of the art in clustering research area which study the behaviour of different algorithms similar to SC [13,31,33,42,68,76]

5.1.1. *Data Description*

The initial datasets considered are 2-dimensional data which can be separated by human intuition, but are problematic to classical clustering algorithms. We have analysed the following datasets:

Data	Instances	Clusters	Structure
Ag	788	7	Parametric
Cp	399	6	Mixture
D31	3100	31	Noisy Parametric
Fl	241	3	Continuity
Jn	373	2	Continuity
Pb	300	3	Noisy Continuity
R15	600	15	Parametric
Sp	312	3	Continuity

Table 3. Synthetic datasets, and their features, used to evaluate the GGC algorithm performance.

- Aggregation[33] (Ag): This dataset is composed of 7 clusters, some of them can be separated by parametric clustering methods.
- Compound[76] (Cp): There are 6 clusters which are only separable by non-parametric methods (or using special kernels if parametric clustering is applied).
- D31[68]: This data has 31 clusters with a high level of noise.
- Flame[31] (Fl): This dataset has three ideal clusters: the first one is the base of the figure, the second one is the top and the last one are three outliers at the top-left of the image.
- Jain[42] (Jn): This dataset is composed of two surfaces with different density and a clear separation.
- PathBased[13] (Pb): This dataset has 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 noisy data.
- R15[68]: Similar to D31, this dataset is divided in 15 clusters which are clearly separated.
- Spiral[13] (Sp): In this case, there are 3 spirals close to each other.

Table 3 summarizes the features of the datasets.

5.1.2. *Experimental Results*

The selected clustering algorithms (K-means, EM using a Gaussian Mixture Model estimator, SC and the GGC algorithm) have been applied to the previous described datasets. We carried out an experiment executing the algorithms 50 times and taking their best results. We selected best fitness as performance measure because of two reasons. First, the goal is to maximize the fitness to achieve the best cluster discrimination, i.e., what Eiben and Jelasity named design domain, and therefore the best fitness is a better choice [26]. Secondly, in our experiments, and on the contrary than other authors observations [50], we observed that similar fitness values are associated to quite different genotypes. It follows the same reasoning of [26] where Eiben and Jelasity explain when these two approaches should be used in GA. Table 4 shows the best accuracy results, and Table 5 shows the parameters and the best fitness values achieved by the GGC algorithm for these datasets. GGC and SC use the RBF kernel [32]. EM and K-means use the Euclidean distance [24]. These metrics are well-known in the literature.

Table 4 shows that Aggregation, Jain and Spirals are not problematic for SC (we are using the Ng [57] version of the algorithm). However, Compound, Flame, PathBased, D31 and R15 are more problematic. Compound is difficult to classify for SC 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 (see Fig. 10), it produces several deviations for the SC algorithm. R15 has also noisy information in the central clusters. Finally, the standard deviation (see Table 4) shows that SC is generally unstable, probably caused by the robustness problem mentioned in Section 4.3.

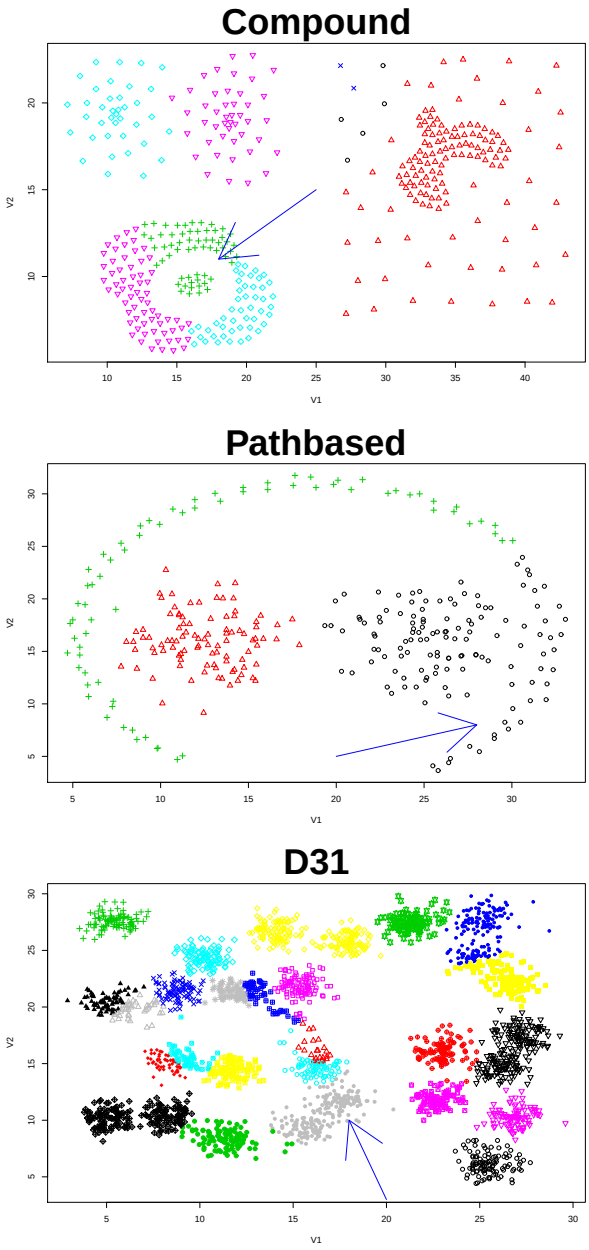


Fig. 10. Three experimental results applying **SC**, using the approach given by Ng et al. [57]. From top to bottom: "Compound", "Pathbased" and "D31". The arrows point the problematic sections.

K-means, as a parametric technique, does not obtain good general results. The reason is that the parameter is a set of centroids optimized by the algorithm. In the case of Compound, for example, the clusters of the top-left position of the image (see Fig. 11) are well classified, however it is impossible, with these conditions, that the algorithm classifies correctly the bottom-left two clusters be-

cause one cluster surrounds the other (see Fig. 11). The same problem appears with Jain, Spirals, PathBased and Flame. In the case of Aggregation, the worst misclassification is related to the three clusters of the bottom-left and the two clusters of the right (see Fig. 11). In this case, the different sizes of the clusters influence the selection process. The D31 and R15 misclassification might be a consequence of a local minimum convergence of the algorithm caused by the noisy information. This algorithm is also unstable according to the standard deviation (see Table 4) due to its local minimum convergence.

Data	SC (%)	GGC (%)	EM (%)	K-M (%)
Pb	**89 ± 8.8**	*88 ± 3.8*	71 ± 5.6	74 ± 6.7
Ag	*96 ± 5.8*	**100 ± 2.1**	79 ± 7.8	86 ± 9.2
D31	85 ± 7.5	**99 ± 2.2**	*90 ± 7.4*	82 ± 6.5
Cp	*77 ± 7.7*	**100 ± 1.3**	57 ± 8.9	72 ± 7.1
R15	81 ± 8.2	**100 ± 2.0**	*100 ± 9.9*	81 ± 8.3
Jn	**100 ± 4.8**	**100 ± 1.8**	57 ± 9.7	78 ± 6.3
Sp	**100 ± 5.3**	**100 ± 1.5**	35 ± 7.3	35 ± 7.6
Fl	**99 ± 5.1**	*99 ± 1.7*	69 ± 9.9	70 ± 8.7

Table 4. Results (and standard deviation) of the different datasets applying K-means, Expectation Maximization, Spectral Clustering and the GGC algorithm. The best results are remarked in bold and the second in italic.

Data	Pop.	Gen.	Cross.	Mut.	Eli.	Fit.
Ag	100	2000	0.4	$0.01\text{-}10^{-4}$	50	0.99
Cp	200	2000	0.5	$0.01\text{-}10^{-4}$	50	0.96
Fl	100	2000	0.4	$0.01\text{-}10^{-4}$	50	0.98
Jn	100	500	0.4	$0.2\text{-}10^{-4}$	50	1.0
Pb	100	2000	0.4	$0.01\text{-}10^{-4}$	50	1.0
R15	200	2000	0.5	$0.3\text{-}10^{-4}$	50	0.99
Sp	100	500	0.4	$0.01\text{-}10^{-4}$	50	1.0
D31	200	5000	0.7	$0.4\text{-}10^{-4}$	50	0.94

Table 5. Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in the GGC algorithm and the best fitness value. The K value of the KNN-Minimal Cut fitness is always set to 2. The tournament size is also 2.

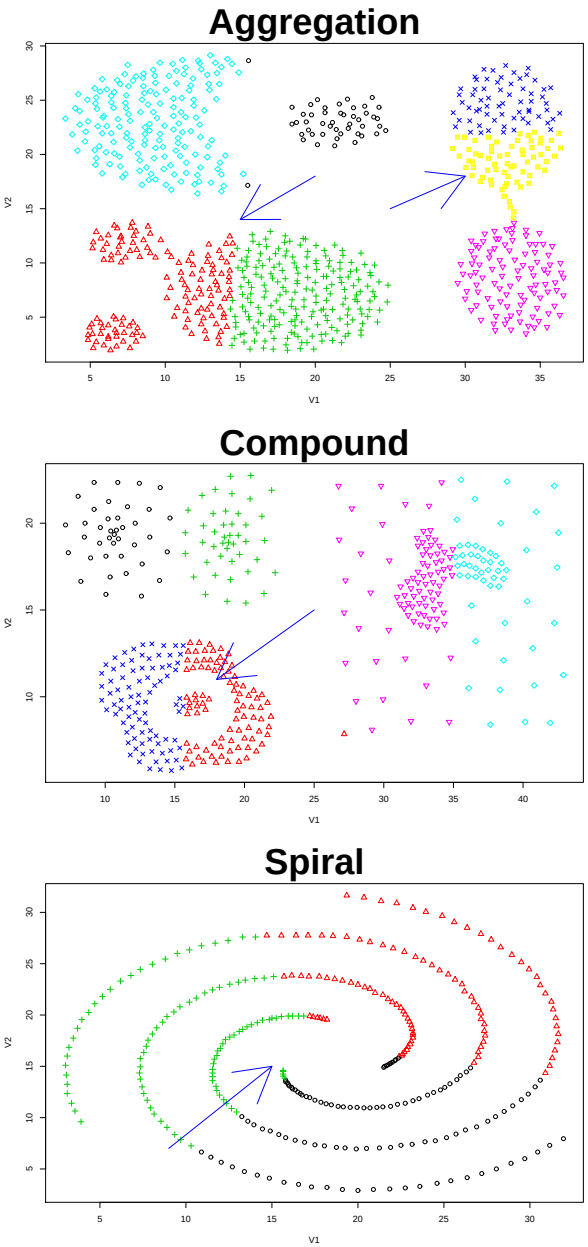


Fig. 11. Three experimental results applying **K-means**, using the classical algorithm [51]. From top to bottom: "Aggregation", "Compound" and "Spiral". The arrows point the problematic sections.

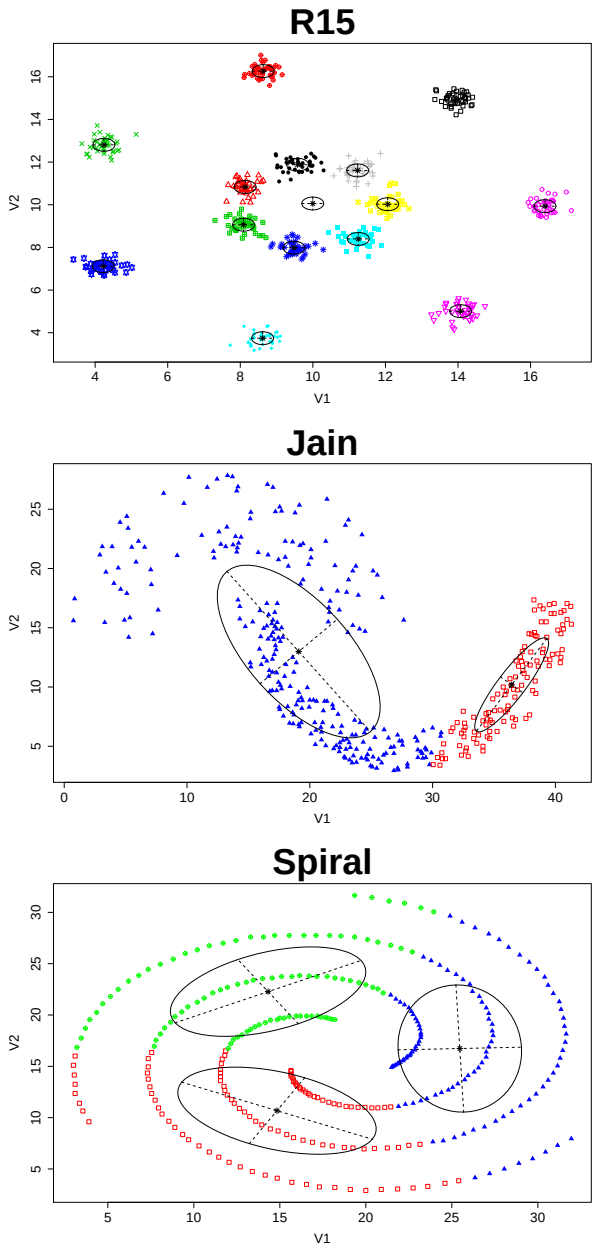


Fig. 12. Three experimental results applying **EM**, using a Gassian Mixture Model [24]. From top to bottom: "D31", "Jain","Spiral".

EM obtains better results than K-means but it also has problems with other datasets. It achieves better results for R15 although the rest of the datasets are misclassified (see Fig. 12). Nevertheless, this algorithm also has stability problems according the standard deviation.

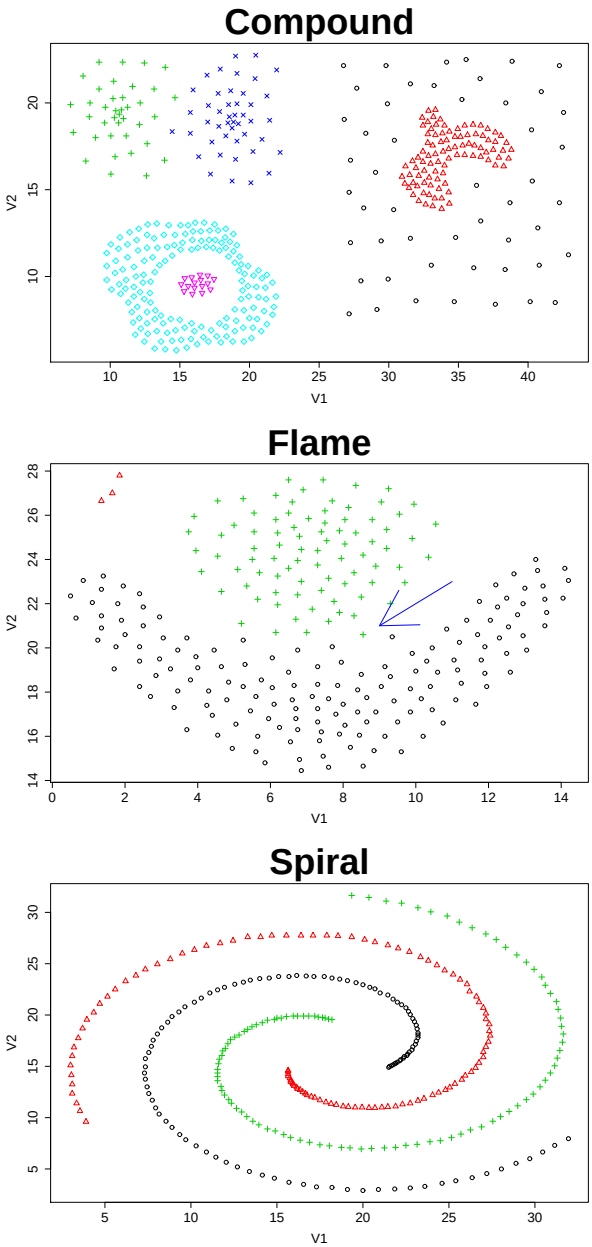


Fig. 13. Three experimental results applying **GGC**, using a Gassian Mixture Model [24]. From top to bottom: "Compound", "Flame","Spiral".

Finally, the GGC algorithm achieves good results in almost all the cases (see Fig. 13). Table 5 shows the parameters selection of the GA for each case. The results show that the GGC algorithm only has problems with the most noisy cases: Flame, Pathbased and D31. The reason is related to a boundary problem. It is difficult for the algorithm (using the RBF metric in the generation of the Similarity Graph), to determine the limits

of the clusters when they are not clear. Also, even if the algorithm has achieved the maximum accuracy values, there are some cases where the fitness function does not obtain the maximum value of its range. It is usual that hard problems such as Compound or D31 do not permit the fitness to find a max-range solution, even if the final cluster selection achieved by the algorithm is closed to the human selection. Finally, GGC is the most robust algorithm according its standard deviation.

5.2. *Experiments on Real Data*

Finally, some experiments have been focused on real datasets which have been previously classified by humans.

5.2.1. *Dataset Description*

The experiments have been applied on three real datasets extracted from the UCI Machine Learning Repository [29]:

- **Iris** (Ir): This dataset is a well-known dataset. It has 150 instances of 3 different classes (50 per class). Each class refers to a type of iris plant: Iris Setosa, Iris Versicolour and Iris Virginica. Each instance has 4 attributes which are Sepal length, Sepal width, Petal length and Petal width. It does not have missing values.
- **Wine** (Wn): This dataset has 178 instances. Each instance has 13 attributes and can belongs to 1 of the three different classes. Each class refers to 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** (HW): This dataset is based on digits handwriting. It has 60000 train instances and 10000 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. It does not have missing values. In this work only 6000 instances of this dataset has been analysed because the Similarity Graph generated by the Spectral Clustering algorithm is bigger than the memory available [‡]

5.2.2. *Preprocessing and Normalizing the Data*

The preprocessing process is divided in two steps:

- The first step has been the study of the available variables through histograms and correlation diagrams which were used for dimension 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.
- The second preprocessing phase consists of the normalization of the variables. First, the attributes with outliers are recentralized. After, the same range is applied for all. We combine Z-score[11] to recentralized the distribution and avoid outliers and MinMax[35] to fixed the range of all the values between 0 and 1.

The Iris and Wine datasets contain a few number of instances and attributes, it implies that the dimentionality 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, for instance those pixels which have always the same value. There are also features which have a high correlation between them. The Handwriting attributes have been reduced in the first step leaving 195 of 784 attributes for the analysis. All the attributes of the datasets have been normalized applying the techniques of the second step.

[‡]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.

5.2.3. *Experimental Results*

The experiments have followed the same procedure that was used with the synthetic datasets experiments. Table 6 reports the parameters selection. The value of σ of the RBF kernel (used to generate the Similarity Graph) has been approximated to 100. Table 7 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 one (92%). The results for the Wine dataset show that all the algorithms obtained high accuracy values (bigger than the 95 %), and the GGC algorithm obtained a perfect classification with the maximum fitness value (see Table 6). Finally, the results of the Handwriting show that SC 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 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 instances which belongs to different clusters but are closed to each other. In the case of the Wine dataset, the classes are clearly 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 fitness value shows that there are some instances in the cluster boundaries and they are difficult to assign. The standard deviation shows that the algorithms stability corresponds with the stability of the synthetic dataset tests.

Data	**Pop.**	**Gen.**	**Cross.**	**Mut.**	**Eli.**	**Fit.**
Ir	1000	2000	0.1	0.8-10^{-4}	50	0.99
Wn	100	20000	0.4	0.01-10^{-4}	50	1
Hw	20	20000	0.9	0.2-10^{-4}	5	0.90

Table 6. Best parameter selection (Population, Generations, Crossover probability, Mutation probability and Elitism size) used in GGC algorithm for the different real datasets and the best fitness value obtained. The K value of the KNN-Minimal Cut fitness is always set to 2. The tournament size is also 2.

Data	**K-M (%)**	**EM (%)**	**SC (%)**	**GGC (%)**
Ir	89 ± 8.8	**97 ± 10.1**	89 ± 8.1%	92 ± 2.1
Wn	96 ± 3.1	**97 ± 4.1**	96 ± 2.9%	**100 ± 1.9**
Hw	51 ± 7.1	35 ± 8.1 %	74 ± 5.1	**99 ± 3.1**

Table 7. Best accuracy values (and the standard deviation) obtained by each algorithm during the experimental results applied to the UCI datasets.

6. Conclusions and Future Work

This work presents a new clustering method inspired in the Spectral Clustering algorithm and based on Genetic Algorithms. The Genetic Graph-based Clustering (GGC) algorithm has been defined comparing different encodings and fitness functions. The main contributions of the algorithm are its simple design according to the encoding and the fitness functions based on Graph Theory measures. The kernel of the GGC algorithm is the fitness function which combines the KNN and Minimum Cut measures. This heuristic is applied to the Similarity Graph which is generated in the first step of the Spectral Clustering method. Several advantages of this approach over SC can be summarized as follows:

- The combination of these measures improves the robustness of the algorithm giving higher independence of the parameters of the Similarity Function metric.
- The memory usage is similar to SC because both work with the same Similarity Graph.
- The experimental results show that the new algorithm obtains good results for both, synthetic and real datasets.

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

Acknowledgments

This work has been partly supported by Spanish Ministry of Science and Education under project TIN2010-19872.

References

1. K. Adamska. Cluster analysis of genetic algorithms results. *Inteligencia Artificial, Revista Iberoamericana de IA*, 9(28):25–32, 2005.
2. H. Adeli and N. Cheng. Concurrent genetic algorithms for optimization of large structures. *Journal of Aerospace Engineering*, 7(3):276–296, 1994.
3. Jose Aguilar. Resolution of the clustering problem using genetic algorithms. *International Journal of Computers*, 1(4):237 – 244, 2007.
4. Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering*, 63(2):503 – 527, 2007.
5. Mehran Ahmadlou and Hojjat Adeli. Enhanced probabilistic neural network with local decision circles: A robust classifier. *Integr. Comput.-Aided Eng.*, 17(3):197–210, August 2010.
6. Mehran Ahmadlou and Hojjat Adeli. Visibility graph similarity: A new measure of generalized synchronization in coupled dynamic systems. *Physica D: Nonlinear Phenomena*, 241(4):326 – 332, 2012.
7. Francis Bach and Michael Jordan. Learning Spectral Clustering, With Application To Speech Separation. *Journal of Machine Learning Research*, 7:1963 – 2001, October 2006.
8. Wesam Barbakh and Colin Fyfe. Online clustering algorithms. *International Journal of Neural Systems*, 18(3):185–194, 2008.
9. 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.
10. Gema Bello-Orgaz, Héctor D. Menéndez, and David Camacho. Adaptive k-means algorithm for overlapped graph clustering. *International Journal of Neural Systems*, 22(05):1250018, 2012.
11. Susan Rovezzi Carroll and David J. Carroll. *Statistics Made Simple for School Leaders*. Rowman & Littlefield, 2002.
12. Timur Chabuk, James A. Reggia, Jason D. Lohn, and Derek S. Linden. Causally-guided evolutionary optimization and its application to antenna array design. *Integrated Computer-Aided Engineering*, 19(2):111–124, 2012.
13. Hong Chang and Dit-Yan Yeung. Robust path-based spectral clustering. *Pattern Recogn.*, 41(1):191–203, January 2008.
14. Ling Chen, Ling-Jun Zou, and Li Tu. A clustering algorithm for multiple data streams based on spectral component similarity. *Information Sciences*, 183(1):35–47, 2012.
15. Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586, 2011.
16. Krzysztof J. Cios, Roman W. Swiniarski, Witold Pedrycz, Lukasz A. Kurgan, Krzysztof J. Cios, Roman W. Swiniarski, Witold Pedrycz, and Lukasz A. Kurgan. Unsupervised learning: Clustering. In *Data Mining*, pages 257–288. Springer US, 2007.
17. Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111–1 – 066111–6, December 2004.
18. Rowena M. Cole. Clustering with Genetic Algorithms. Master's thesis, Nedlands 6907, Australia, 1998.
19. Coley. *An Introduction to Genetic Algorithms for scientists and engineers*. World Scientific Publishing, 1999.
20. T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21 –27, january 1967.
21. 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.
22. Luke M. Davis, Barry-John Theobald, Jason Lines, Andoni Toms, and Anthony Bagnall. On the segmentation and classification of hand radiographs. *International Journal of Neural Systems*, 22(05):1250020, 2012.
23. M. Dehmer, editor. *Structural Analysis of Complex Networks*. Birkhäuser Publishing, 2010.
24. 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.
25. 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.
26. A E Eiben and Márk Jelasity. A critical note on experimental research methodology in EC. In *In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002*, pages 582–587. IEEE, 2002.
27. 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.
28. Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community struc-

tures based on information centrality. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(5):056104–1–056104–13, 2004.
29. A. Frank and A. Asuncion. UCI machine learning repository, 2010.
30. Alex A. Freitas. A review of evolutionary algorithms for data mining. In *In: Soft Computing for Knowledge Discovery and Data Mining*, pages 61–93, 2007.
31. Limin Fu and Enzo Medico. Flame, a novel fuzzy clustering method for the analysis of dna microarray data. *BMC Bioinformatics*, 8:1–15, 2007.
32. S. Ghosh-Dastidar, H. Adeli, and N. Dadmehr. Principal component analysis-enhanced cosine radial basis function neural network for robust epilepsy and seizure detection. *Biomedical Engineering, IEEE Transactions on*, 55(2):512 – 518, feb. 2008.
33. Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1):1–30, March 2007.
34. Alexander N. Gorban and Andrei Zinovyev. Principal manifolds and graphs in practice: From molecular biology to dynamical systems. *International Journal of Neural Systems*, 20(3):219 – 232, 2010.
35. Jiawei Han and Micheline Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.
36. Julia Handl, Julia H, and Joshua Knowles. Evolutionary multiobjective clustering. In *In Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pages 1081–1091. Springer, 2004.
37. Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4–6):175–181, 2000.
38. E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, and A.C.P.L.F. 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. Wei-Yen Hsu. Application of competitive hopfield neural network to brain-computer interface systems. *International Journal of Neural Systems*, 22(01):51–62, 2012. PMID: 22262524.
40. S.L. Hung and H. Adeli. A parallel genetic/neural network learning algorithm for mimd shared memory machines. *Neural Networks, IEEE Transactions on*, 5(6):900–909, 1994.
41. Curtis Huttenhower, Avi Flamholz, Jessica Landis, Sauhard Sahi, Chad Myers, Kellen Olszewski, Matthew Hibbs, Nathan Siemers, Olga Troyanskaya, and Hilary Coller. Nearest Neighbor Networks: clustering expression data based on gene neighborhoods. *BMC Bioinformatics*, 8(1):250, 2007.
42. Anil Jain and Martin Law. Data clustering: A user's dilemma. In Sankar Pal, Sanghamitra Bandyopadhyay, and Sambhunath Biswas, editors, *Pattern Recognition and Machine Intelligence*, volume 3776 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin / Heidelberg, 2005.
43. Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.
44. Keehyung Kim, RI (Bob) McKay, and Byung-Ro 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.
45. Vassilis S. Kodogiannis, Mahdi Amina, and Ilias Petrounias. A clustering-based fuzzy wavelet neural network model for short-term load forecasting. *International Journal of Neural Systems*, 0(0):1350024, 2013.
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. W.B. 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.
48. Daniel T. Larose. *Discovering Knowledge in Data*. John Wiley & Sons, 2005.
49. Marek Lipczak and Evangelos 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.
50. Sean Luke. Is the perfect the enemy of the good. In *In Genetic and Evolutionary Computation Conference*, pages 820–828. Morgan Kaufmann, 2002.
51. 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.
52. U Maulik. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.
53. Héctor D. Menéndez and David Camacho. A ge-

netic graph-based clustering algorithm. In Hujun Yin, José Costa, and Guilherme Barreto, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, volume 7435 of *Lecture Notes in Computer Science*, pages 216–225. Springer Berlin / Heidelberg, 2012.

54. Boaz Nadler, Stéphane Lafon, Ronald Coifman, and Ioannis G. Kevrekidis. Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators. *Advance in Neural Information Processing Systems*, 18:955 – 962, 2006.
55. Mariá Cristina Vasconcelos Nascimento and André 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.
56. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.
57. 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.
58. 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.
59. Riccardo Poli and William B. Langdon. Backward-chaining evolutionary algorithms. *Artificial Intelligence*, 170(11):953 – 982, 2006.
60. Rahul Putha, Luca Quadrifoglio, and Emily M. Zechman. Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions. *Comp.-Aided Civil and Infrastruct. Engineering*, 27(1):14–28, 2012.
61. Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006.
62. Franz Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer-Verlag, Heidelberg, New York, 2nd editio edition, 2006.
63. Dharmendra K Roy and Lokesh 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.
64. Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
65. Liang-Dong Shi, Ying-Huan Shi, Yang Gao, Lin Shang, and Yu-BinN Yang. Xcsc:: A novel approach to clustering with extended classifier system. *International Journal of Neural Systems*, 21(1):79 – 93, 2011.
66. M. Srinivas and L.M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656 –667, apr 1994.
67. Lin Yu Tseng and Shiueng Bien Yang. A genetic approach to the automatic clustering problem. *Pattern Recognition*, 34(2):415 – 424, 2001.
68. C.J. Veenman, M.J.T. Reinders, and E. Backer. A maximum variance cluster algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(9):1273 – 1280, sep 2002.
69. Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.
70. Ulrike von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586, April 2008.
71. Michael D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, USA, 1998.
72. Huiqing Wang, Junjie Chen, and Kai Guo. A genetic spectral clustering algorithm. *Journal of Computational Information Systems*, 7(9):3245–3252, 2011.
73. Lin Wang, Minchu Jiang, Yinghua Lu, Minfu Sun, and Frank Noe. A comparative study of clustering methods for molecular data. *International Journal of Neural Systems*, 17(6):447 – 458, 2007.
74. Duncan J Watts. *Small worlds: The dynamics of networks between order and randomness*. Princeton University Press, Princeton, NJ, 1999.
75. Wojciech and Kwedlo. A clustering method combining differential evolution with the k-means algorithm. *Pattern Recognition Letters*, 32(12):1613 – 1621, 2011.
76. C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, C-20(1):68 – 86, jan. 1971.