International Journal of Neural Systems, Vol. 0, No. 0 (April, 2000) 00–00
© World Scientific Publishing Company

ADAPTIVE K-MEANS ALGORITHM FOR OVERLAPPED GRAPH CLUSTERING

Gema Bello-Orgaz†, Héctor D. Menéndez‡and David Camacho*
Computer Science Department
Escuela Politecnica Superior, Universidad Autónoma de Madrid
28049, Madrid, Spain
†*gema.bello@uam.es*
‡*hector.menendez@uam.es*
* *david.camacho@uam.es*

Received (to be inserted
Revised by Publisher)

The graph clustering problem has become highly relevant due to the growing interest of several research communities in social networks and their possible applications. Overlapped graph clustering algorithms try to find subsets of nodes that can belong to different clusters. In social-based applications it is quite usual for a node of the network to belong to different groups, or communities, in the graph. Therefore, algorithms trying to discover, or analyse, the behaviour of these networks need to handle this feature, detecting and identifying the overlapped nodes. This paper shows a soft clustering approach based on a genetic algorithm where a new encoding is designed to achieve two main goals. First, the automatic adaptation of the number of communities that can be detected. Second, the definition of several fitness functions that guide the searching process using some measures extracted from graph theory. Finally, our approach has been experimentally tested using the Eurovision contest dataset, a well-known social-based data network, to show how overlapped communities can be found using our method.

Keywords: graph clustering, overlapped clustering, genetic algorithms, clustering coefficient, community finding, social networks.

1. Introduction

The clustering problem can be described as a blind search on a collection of unlabelled data, where elements with similar features are grouped together in sets. There are three main techniques to deal with the clustering problem [32]: overlapping [12] (or non-exclusive), partitional [42] and hierarchical [37]. Overlapping clustering allows each element to belong to multiple clusters, partitional clustering consists in a disjoint division of the data where each element belongs only to a single cluster, and hierarchical clustering nests the clusters formed through a partitional clustering method creating bigger partitions, grouping the clusters by hierarchical levels. In this work, the approach is focused in the overlapping clustering techniques trying to "relax" a well-known classical partitional technique named K-means using a genetic algorithm approach. K-means is a clustering algorithm that uses a fixed number (K) of clusters and looks for the best division of the dataset (through a predefined metric or distance) in this number of groups.

Several clustering algorithms, such as K-means, have been improved using genetic algorithms [32]. A genetic algorithm is inspired by biological evolution [38]: the possible problem solutions are represented as individuals belonging to a population. The individuals are encoded using a set of chromosomes (called the genotype of the genome). Later these individuals are evolved, during a number of genera-

*Corresponding author.

tions, following a survival/selection model where a fitness function is used to select the best individuals from each generation. Once the fittest individuals have been selected, the algorithm reproduces, crosses and mutates them trying to obtain new individuals (chromosomes) with better features than their parents. The new offspring and, depending on the algorithm definition, their parents, will pass to the following generation. This kind of algorithms have been usually employed in optimization problems [24,3], where the fitness function tries to find the best solution among a population of possible solutions which are evolving. In other approaches, such as clustering, the encoding and optimization algorithm are used to look for the best set of groups that optimize a particular feature of the data. In our approach each chromosome is used to define a set of K clusters which represents a solution to the clustering problem.

Clustering techniques can also be applied to different kinds of representations of the data collection like strings, numbers, records, text, images and semantic or categorical data [39,59,61]. In our approach, we apply the new clustering technique to data that can be represented as a graph, trying to find groups whose nodes share similar graph-based features.

The proposed technique is based on genetic algorithm methods for clustering and graph-based clustering techniques that are described in the next section. We are trying to combine these approximations to improve the results of graph clustering through classical optimization methods. The main contribution of this work can be summarized as follows: our approach tunes up the centroid positions and the number of clusters (K), maximizing the distance between them, and minimizing the distance between the elements found in each cluster. [11].

We also based our algorithm on network analysis techniques [18]. These techniques are usually based on graph theory methods, because a graph is the most common and straightforward representation for a network. The main measures used to analyse networks are the average distance between nodes, and the clustering coefficient (CC). The CC can be seen as the number of triangles formed by the edges of the network over the total possible number of triangles. Both these measures are usually employed to define the nature of the network [18]. We have used a modified clustering coefficient that can be applied in directed and weighted graphs [10,68] to experimentally study how it could be employed in a genetic-based clustering process.

Distance between nodes, clustering coefficient and the weighted clustering coefficient measures can be used to guide a genetic clustering algorithm with the goal of finding groups in a graph (or weighted graph) which minimize or maximize these measures. Although each of the measures can be used separately, our genetic algorithm approach combines them using a hybrid function which gives different weights to each measure. This combination generates some problems specially when it is necessary to decide which measure is more relevant than the others. That is the reason why some experimental tests have been carried out to obtain the final weight for each measure that will be used in the hybrid fitness function.

Finally, once a particular encoding and several fitness functions were designed, we applied the new algorithm to the Eurovision Contest Song dataset. This well-known contest provides interesting data which has been deeply studied and analysed from different perspectives (social, political, economical and historical, among others) over the last decades [27,49]. This data has been preprocessed and represented as a social network.

The rest of the paper is structured as follows. Section 2 describes the related work concerning clustering, genetic algorithm and community-finding algorithms. Section 3 presents some basic definitions referred to graph concepts that are later used to design our genetic algorithm. Section 4 shows the two genetic algorithms (with a fixed value of K, and the K-adaptive version) and the encoding designed, the fitness functions and other characteristics of the algorithms, like crossover and mutation operators. Section 5 provides a description of the dataset used, the experimental setup of the algorithms and a complete experimental evaluation of them. Finally, in Section 6 the conclusions and some future research lines of work are presented.

2. Related Work

This section starts with a general introduction to clustering techniques. After this brief introduction, we focus our attention on how genetic algorithms have been applied to clustering techniques. Later, we present an overview of graph clustering methods based on spectral clustering techniques, and some current applications to social networks that uses the clustering coefficient. Finally, we show some community finding algorithm methods paying close attention to social network analysis.

2.1. *Clustering*

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 [42] based on a set of common features given by distances or metrics that are used to determine how the cluster should be defined. Other approximation, such as Expectation-Maximization (EM) [19], uses a variable number of clusters. 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 [46].

Other research lines are trying 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 [9]. Some other improvements of K-means algorithm are related to deal the different kind of data representation, for example, mixed numerical data [5] and categorical data [57] . And there are also some studies comparing methods with different datasets, for example, Wang et al. [67] 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 2.2.

Machine learning techniques have also been improved through the k-means algorithm, for example, reinforcement learning algorithms[8,26]; or using topological features of the data set [25,26] which can also be helpful for data visualization.

As we mentioned before, in our approach we are working with overlapping clustering instead of partitional clustering (which is the case of the original K-means). In overlapping clustering there are two main approaches[32]: soft (each object fully belongs to zero or more clusters) and fuzzy (each object belongs to zero or more clusters with a membership probability). Fuzzy instances are important when there is not a complete deterministic separation in the data set, a good example is human activity recognition [34]. One of the first approximations was fuzzy K-means [51], which can also benefit from combining with a genetic approach [7,41]. In our problem (overlapped clustering in social data) soft computing allows each node in the graph to belong to one or more subgraphs, and no membership probability is considered.

2.2. *Genetic Algorithms for Clustering*

Genetic algorithms have been traditionally used in a large number of different domains, mainly related to optimization problems [13,17,58]. 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 [16,62]. These algorithms have also been used for general data and information extraction [24]. The operators of the genetic algorithms can also be modified. Some examples of these modifications can be found in (Poli and Langdon, 2006)[54] 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, [38] software systems [21], file clustering [22] and task optimization [53], 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)[15], 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)[32] which provides a complete updated review in evolutionary algorithms for clustering.

There are several methods using evolutionary approaches from different perspectives, for example: (Aguilar, 2007)[4] modifies the fitness considering cluster asymmetry, coverage and specific information of the studied case; (Tseng and Yang, 2001)[63] use a compact spherical cluster structure and a heuristic strategy to find the optimal number of clusters; (Maulik and Bandyopadhyay, 2000)[43] use the clustering algorithm for metric optimization trying to improve the cluster centre positions; (Shi et al., 2011)[60] 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)[17] 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)[36] replace the crossover of the algorithm using K-means as a search operator, and (Wojciech and Kwedlo, 2011)[69] 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)[2]. There are also other complete studies for multi-objective clustering in (Handl et al., 2004)[30] and for Nearest Neighbour Networks in (Huttenhower et al., 2007)[33].

2.3. *Graph Clustering*

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 [29] 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 [18,23,45,68].

A complete roadmap to graph clustering can be found in (Schaeffer, 2007)[59] 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, 2000)[31]), 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 the spectral clustering. Next subsection, describes briefly the basic concepts of this technique.

2.3.1. *Spectral Clustering*

Spectral clustering methods are based on a straightforward interpretation of weighted undirected graphs as can be seen in [65,6,48,44]. The Spectral clustering approach is based on a similarity graph which can be formulated in three different types (equivalents[65]) of graphs: the ϵ-neighbourhood graph (all the components whose pairwise distance is smaller than ϵ are connected), 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) and 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. Some works focus on this problem: (von Luxburg, 2007)[65] presents the problem, (Ng et al.,2001)[48] applies an approximation to a specific case, and (Nadler et al.), [44] applies Fokker-Planck operators to get better results.

The classical algorithms can be found in (von Luxburg, 2007)[65], and a particular modification of them which obtains good clustering results, similar to human selection, can be found in (Ng et al.,2001)[48].

The theoretical analysis of this observed good behaviour is justified using the perturbation theory [65,44], random walks and graph cut [65]. The perturbation theory is also the main approximation for the proofs about convergence of the modification of Fokker-Planck operators and 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 typical 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)[66].

Part of the present work is inspired by spectral clustering because we use clustering techniques which analyse similarity graphs. Nevertheless, in our case we are using different methods such as the clustering coefficient measures to find the subgraphs, even to use the Laplacian matrix extracted from the similarity graph.

2.3.2. *Clustering Coefficient (CC)*

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 [18,45,68].

The present work is close to the network approach and has been developed over different kinds of graphs. In our case, we are working with undirected, directed and weighted graphs, and we apply the graph structure to the clustering coefficient using this new value to find clusters in the network [45].

2.4. *Community Finding Approach*

The main application of the communities approach are social networks. The clustering problem is more complex when applied to find communities in networks (subgraph identifications). A community can be considered as a subset of individuals with relatively strong, direct, and intensive connections [23] between them. Some algorithms such as Edge Betweenness Centrality (EBC) [28] or Clique Percolation Method (CPM) [20] have been designed to solve this problem following a deterministic process. Both algorithms have been applied to community classification and detection in (Bello et al., 2011)[50]. EBC algorithm [28] is based on finding the edges of the network which connect communities and removing them to determine a good definition of these communities. CPM [20] finds communities using k-cliques (where k is a fixed value of connections in a graph) which are defined as complete (fully connected) subgraphs of k vertices. It defines a community as the highest union of k-cliques. CPM has two variants: directed graphs and weighted graphs [52].

In the initial study of the problem [11], we adopt an evolutionary approach based on the K-means algorithm applied to community finding approach. However, in the process of community finding problems, K-means algorithm cannot be directly applied because it does not allow overlapping. But our representation for communities in form of overlapped subgraphs does not need membership probability for a node. And our fitness design allows to extend the algorithm to consider overlapped groups.

Other approximations related to the finding-community problem can be found in (Reichardt and Bornholdt, 2006)[56] where different statistical mechanics for community detection are used. (Pons and Latapy, 2005)[55] uses 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)[47], and metrics that can be used to find the structure of a community in very large networks in (Clauset et al., 2004)[14]. Genetic algorithms have also been applied to find communities or clusters through agglomerative genetic algorithms [40] and multi-objective evolutionary algorithms [35] amongst others.

3. Basic Definitions from Graph Theory

Defining and selecting an appropriate fitness function is one of the most critical issues in any evolutionary algorithms. 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 1 (Graph) *A graph $G = (V, E)$ is a set of vertices or nodes V denoted by $\{v_1, \dots, v_n\}$ 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 (Adjacency Matrix) *An adjacency matrix of G, A_G, is a square $n \times n$ matrix where each coefficient satisfies:*

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

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

Definition 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 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 $\Gamma_{v_i} = \{v_j \mid e_{ij} \in E \text{ and } \mathrm{e_{ji}} \in \mathrm{E}\}$. 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 and the weighted clustering coefficient.

Definition 5 (Local CC [18]) *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 6 (Local Weighted CC [10]) *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 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: $k_i = \sum_{j=1}^{|V|} a_{ij} a_{ji} = |\Gamma_{v_i}| = |\{v_j \mid e_{ij} \in E \text{ and } e_{ji} \in E\}|$ And finally:

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

Which proves theorem 1 $\Box$.

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

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

Definition 7 (Global CC [18,10]) *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 [64].

4. Genetic-based Community Finding Algorithm

The Genetic-based Community Finding (GCF) Algorithm uses a genetic algorithm to find the best K communities in a dataset that has been represented as a graph, and where any particular neighbour could belong to different clusters. In our initial work [11] we develop a simple version of the algorithm with a binary encoding using a fixed value for K, we will name this algorithm *K-fixed GCF*, or simply K-fixed algorithm. The experiments carried out show us some important improvements that could be made to obtain better solutions for the communities or subgraphs detected, and to increase the performance of the clustering process (taken as of accuracy between the obtained communities and the correlations with the weighting values of the subgraph). To achieve these stated goals, a more complex encoding has been designed to include the value of K in the evolutionary process, we will name this new version of the algorithm K-adaptive GCF.

This section describes the previous and the new algorithm including the encoding, the genetic algorithm (crossover and mutation operators) and the fitness functions designed.

4.1. *K-fixed GCF Algorithm*

Our initial version of the algorithm was based on a standard genetic algorithm with a binary codification to represent a community, the number of possible K communities was fixed to a predefined value. The goal of this algorithm was to find overlapping communities in a dataset represented as an undirected graph.

4.1.1. *Encoding*

In this version of the algorithm the genotypes are represented as a set of binary values. Each allele represents the membership of a node of the graph and each chromosome is used to represent a community. The chromosome length will be equal to the graph size.

This encoding defines a direct relationship between each node in the graph and the allele of the chromosome. In this binary representation the value "1" means that the node belongs to a community and the value "0" the opposite (see Fig. 1).

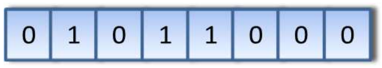

Fig. 1. A chromosome representing a community. Each allele represents a node of the graph and its belonging, or not, to the current community. In this example, a community built by three nodes of the graph is shown.

4.1.2. *The Algorithm*

The basic GCF Algorithm with a fixed K works as follows:

1. A random population of communities is generated.
2. The population evolves using a standard GA. Therefore, the following steps are repeated until a fixed number of iterations, or a convergence value, are reached:
   (a) Evaluate the fitness function of each chromosome in the population.
   (b) Copy the n-best chromosomes to the new population (Elitism Selection). It prevents losing the n-best found solutions.
   (c) Generate the rest of the new population by repeating the following steps:
      i. **Selection**: select two parent chromosomes from the population.
      ii. **Crossover**: crossover the parents to form a new offspring.
      iii. **Mutation**: using a given mutation probability, the value for each bit in the allele is changed.
   (d) Replace the old population with the new population.
3. The chromosomes which are the K-best solution of the algorithms are selected. Our selection process **subsumes** the communities which have better fitness and belong to a bigger community. An individual subsumes another when the subgpraph that represents its community, contains at least all the nodes and connections of the other one. This **subsumption** process has the following steps:
   (a) An empty list of K elements is created.
   (b) The chromosomes are sorted by their fitness value.
   (c) While the list is not full, a new chromosome is selected. If the new individual represents a new community, it is included in the list. However, if this individual represents a community that currently is contained by some other individual in the list (the nodes encoding this chromosome are a subset of a currently stored chromosome), the more general chromosome is selected.
   (d) The process stops when the K best individuals are found, or when there are no more individuals to select.

Finally, the rest of the basic characteristics of the GA algorithm; selection, crossover and mutation operators, are briefly described:

- **Selection**. The parent selection can be done in different ways, but the main idea is to select the better parents to produce better offspring in each generation. When creating the new population by crossover and mutation, there is a big probability of loosing the best community (chromosome). So we have used the elitism selection method which first copies the n-best communities to the new population. The rest of the population is generated in a classical way, as we have described in the previous steps of the algorithm.

- **Crossover**. To do the crossover, the algorithm chooses two crossover points at random. Then everything between these two points is copied from the first parent to the second and vice versa, see Fig. 2.

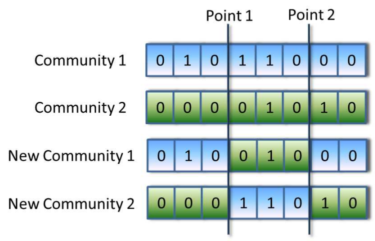

Fig. 2. Crossover of two communities.

- **Mutation**. Once the crossover process has finished the mutation is executed. This operator is applied to prevent the falling of all solutions into a local optimum of the problem. In our approach, for a binary encoding, we have chosen a few alleles (nodes of a community) at random and changed their values from 1 to 0 or viceversa using a mutation probability. The mutation operator will work as we can see in Fig. 3.

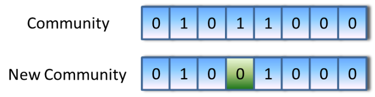

Fig. 3. Mutation of a community. The fourth allele has been selected, and the bit has been changed using the mutation probability, so this node in now excluded from the community.

4.1.3. *Fitness Functions*

In our approach we have implemented four fitness functions, each of them with a different goal. The first one tries to find nodes with a similar rating behaviour (minimal distance fitness), the second one tries to find clusters using the clustering coefficient (maximum clustering coefficient fitness), the third one is similar to the previous but using the weighted clustering coefficient and, finally, the last fitness function (hybrid fitness) combines both strategies (minimal distance and maximum CC) to find communities with a similar rating behaviour and whose members are connected between them. These fitness functions can be described as follows:

- **Minimal Distance Fitness (MDF)**. The objective of this fitness function is to find similar node communities. The evaluation of this fitness function is done using the following criteria:

    1. Each node belonging to a community is represented as a vector of attributes. The definition of these attributes depends on the problem being solved.
    2. The average euclidean distance between vectors of attributes within a community is calculated. The fitness calculates distances to be taken into account from peer to peer, between all vectors.
    3. The fitness value for the community is the average distance of the values calculated in the previous step (we are trying to minimize the fitness). It is a measure of similarity for those rows, hence it checks if they follow the same similarity pattern. We call this average distance d_{in} (see Fig. 7).
    4. Fitness penalizes those cases where the community has a single node, giving it a value of zero.

- **Maximum Clustering Coefficient Fitness (MCCF)**. The goal of this fitness is to discover communities whose members are connected between them. It is measured through the clustering coefficient, defined as follows: the fitness takes the sub-graph defined by the community and calculates its clustering coefficient. It returns the inverse value, because the genetic algorithm tries to minimize the fitness function.

- **Maximum Weighted Clustering Coefficient Fitness (MWCCF).** This fitness is similar to the Clustering Coefficient fitness. The main difference between them is the way they consider the community definition. As we show in section 3, Weighted Clustering Coefficient considers to be stronger those communities whose sub-graphs have higher weight values. It also returns the inverse value, because the genetic algorithm tries to minimize the fitness function.

- **Hybrid Fitness(HF).** This last fitness function combines both Clustering Coefficient and Distance fitness ideas: it tries to find a set of communities satisfying both conditions previously defined. With this method we try to find strong and similar communities (members are highly connected between them and they have similar behaviour). The function defined is a simple weighted function: suppose that $F(x, y)$ is the fitness function, CC the clustering coefficient and d_{in} the value of HF fitness is:

$$F_i(CC, d_{in}) = w_1 * \frac{CC_i}{Max(\{CC_i\}_{i=1}^{K})} + w_2 * \frac{d_{in_i}}{Max(\{d_{in_i}\}_{i=1}^{K})}$$

  Where w_i are the weights given to each fitness: $w_i \in (0, 1)$. The values were set experimentally to $w_1 = 0.1$ and $w_2 = 0.9$.

4.2. *K-adaptive GCF Algorithm*

In the previous algorithm, one of the possible improvements that can be performed is allowing the parameter K (the number of communities found) to change its value through the execution of the clustering process. To achieve this, the encoding and the fitness function have been modified to obtain a new algorithm.

4.2.1. *Encoding*

In this new approach, the possible solutions can contain groups of communities, and not just an individual community. For this reason, the genotypes (chromosomes) are represented as a set of vectors of binary values. Each allele represents a community that is composed by a set of binary values, one for each node in the graph. This binary vectors are similar to the chromosomes of the previous encoding, the value 1 meaning that the node belongs to the community and value 0 the opposite. The number of binary vectors (communities) that the chromosome (group of communities) has, corresponds to the value of K in the solution, see Fig. 4.

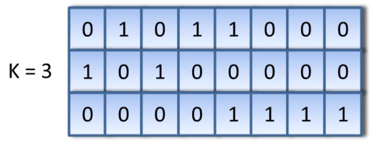

Fig. 4. A chromosome representing a group of communities of the graph. Each allele is a individual community where its binary vector represents the nodes of the graph and their belonging or not to the current community. In this example the solution contains 3 vectors representing three different communities, hence the K is equal to 3.

In this new codification the length of a particular chromosome could vary from the total length of the graph (with K = 1) to K times this size (where K is equal to the maximum number of subgpraphs allowed). This variable length of the chromosomes will be adequately managed by the crossover operator to generate correct individuals for the evolutionary process.

4.2.2. *The Algorithm*

The GCF Algorithm with adaptive K works as follows:

1. A random population of community groups is generated.

2. The population evolves using a standard GA. The steps of the process are the same as was previously described in the GCF algorithm with fixed K.

3. The chromosome that has the best fitness function value is selected as a final solution.

Although the genetic algorithm has not been changed, the new codification has modified how the genetic operators (crossover and mutation) are applied. The new operators works as follows:

- **Crossover**. To apply the crossover operator, the algorithm chooses a random crossover point. Then every community preceding this point is copied from both parents to create a first new child, and every community succeeding this point is copied to create a second new child, as Fig. 5 shows.

- **Mutation**. Once the crossover operator has finished, the mutation is executed. The algorithm chooses some values of the vectors that represent the communities at random, and change their values (with a predefined probability) from 1 to 0 or viceversa, see Fig. 6.

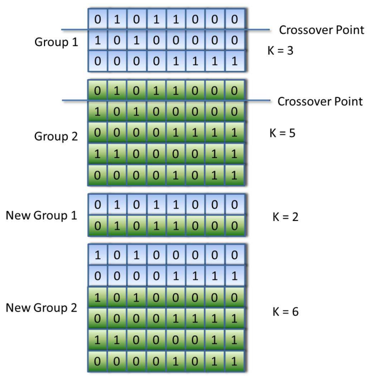

Fig. 5. Crossover of two groups of communities with different K. The new generated offspring maintains the maximum length of a community, but it uses an adaptive K.

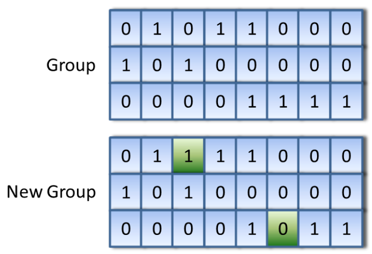

Fig. 6. Mutation of a group of communities with K equal to 3. In this example, two nodes from two different communities have been modified.

4.2.3. *The Clustering Centroid Fitness Function*

Our initial encoding only allows to use metrics related to measures of a member belonging to their own community. Therefore, metrics such as the clustering coefficient or the minimal distance between nodes were used. However, the new encoding makes possible to include measures between groups of different communities.

We have designed a new fitness function, called Clustering Centroid Fitness, that calculates the distance between the community centres belonging to a particular chromosome. This new measure is called d_{out} and it has been represented in Fig. 7. With this new measure, large distances between centres could be desirable because it represents a bigger gap between classes or communities.

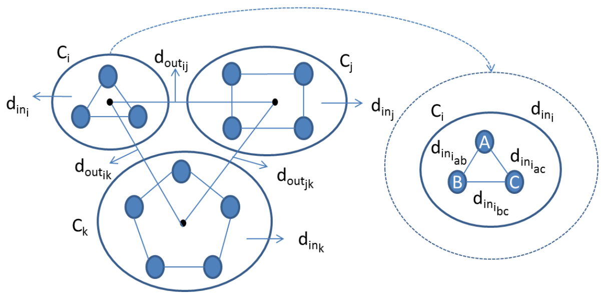

Fig. 7. Sample network graph illustrating three communities and the distances that are calculated in the fitness function of the algorithm. The distance d_{in} represents the average distance calculated between the nodes which belong to a community. The distance d_{out} represents the distance between community centres.

As a result of this new measure, that can be calculated for each individual, a new fitness function which combines the Clustering Coefficient, the distance between nodes (d_{in}) and finally the distance between centres (d_{out}) can be designed. The idea of this new fitness is to find a set of communities that could satisfy all of the previously defined conditions. This new method tries to find groups of communities where each community is strong and similar, but also whose communities are the most different as possible between themselves.

The function defined is a simple weighted function: let $F(x, y)$ be the fitness function, CC the clustering coefficient, d_{in} the distance between nodes, and d_{out} the distance between centres, the value of the new fitness is calculated as follows:

$$F_i(CC, d_{in}, d_{out}) = w_1 * \frac{CC_i}{Max(\{CC_i\}_{i=1}^{K})}$$
$$+w_2 * \frac{d_{in_i}}{Max(\{d_{in_i}\}_{i=1}^{K})}$$
$$+w_3 * \frac{d_{out_i}}{Max(\{d_{out_i}\}_{i=1}^{K})}$$

Where w_i are the weights given to each fitness: $w_i \in (0,1)$. The values were set experimentally to $w_1 = 0.05$, $w_2 = 0.05$ and $w_3 = 0.9$.

5. Experiments

5.1. *Dataset Description*

The Eurovision Song Contest has been studied using different clustering methods since the nineties [27,49]. The main interest was to study and analyse alliances between countries, which had been reflected in form of communities or country clusters found. For this reason we have selected the dataset of this contest to carry out the experimental phase of our algorithm. The data used in this work has been extracted from Eurovision's official website [1].

Since 1975, the scoring system in the Eurovision Contest consists of the following rules. Each country distributes among other participants the following set of points: 1, 2, 3, 4, 5, 6, 7, 8, 10, 12. Countries give the highest score to the best song and the lowest to the less popular or less preferred. Once all the votes are added up, the final ranking is obtained. The country with the highest score wins the contest.

The contest has undergone a series of changes throughout the years in its voting system. From 1956 to 1996, votes where casted by a *jury* of representatives sent from each of participating country. In 1997 *televoting* was introduced in five countries, to gradually displace the jury-based system until 2004 when *televoting* was made mandatory for all participants. *Televote* technology allows viewers to cast their votes via phone, SMS or the internet for a set window of time–normally within the live broadcast. Finally, 2009 saw the implementation of the current voting system, a *hybrid* system of *televoting* and a *jury* was implemented, whereby each part contributes half of the total vote cast for each country.

This data can be easily represented using a graph for each year of the contest. In this graph, the nodes will be countries and the points emitted can be used to weight the edges. The graph could be *directed* (the edges represent votes), or *undirected* (the edges only connect countries which have exchanged points in any direction). If we consider the latter, it is similar to setting edge weights uniformly to 1. According to this, the dataset of the votes emitted in a particular year could be represented as a graph, as is showed in Fig. 8.

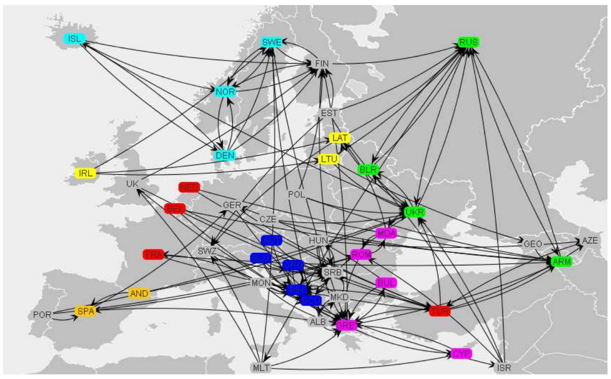

Fig. 8. Eurovision graph example illustrating the votes emitted between the countries in 2009.

5.2. *Preliminary analysis of fitness functions*

In our previous work [11] a data analysis using the clustering coefficient was performed. This analysis confirms the existence of clusters or communities in the Eurovision graph representation. The 2009 year dataset has the greatest difference in clustering coefficient, meaning this year contains a large set of different communities. Hence, we have selected this year to perform an initial study for all the fitness functions designed.

This preliminary study has been divided in two parts, one for each version of the algorithm that has been described in section 4. To compare these algorithms, the following measures (which have been previously defined in the fitness functions description) are considered:

- d_{in}: It gives information about the node similarity within clusters.

- CC: It gives information about the inner connections of the clusters or the k-cliques.

- d_{out}: It gives information about the distances between centroids.

Table 1 shows the experimental set up. In this table, we can see the parameters of the K-fixed and K-adaptive versions of the algorithm that have been experimentally obtained. $\mu + \lambda$ is the selection criteria used in both genetic algorithms, where λ is the number of offspring (population size), and μ is the number of the best parents that survive from the current generation to the next.

5.2.1. *Fitness Function Analysis for the K-fixed Algorithm*

The clustering coefficient and the distances d_{in} and d_{out} have been calculated to compare the results obtained by each fitness function. Firstly, using the K-fixed algorithm, and analysing the results obtained for the regular clustering coefficient and the weighted clustering coefficient. The obtained values of these measures for each fitness functions are shown in Table 2.

In terms of distance measures, both have been greatly improved using the hybrid fitness (HF and WHF). The distance between centres (d_{out}) increases dramatically from the MC^{2}F and MWC^{2}F functions to the hybrid ones. Therefore, the communities found are far from each other and they can be better differentiated. The intra-cluster distance (d_{in}) obtains lower values, meaning that the found communities have more similar members. Finally, the clustering coefficient takes similar and very high values in all the cases.

On the other hand, the addition of the weights to the functions improves the distance (d_{out}) in the clustering coefficient fitness functions, but it worsens the distance (d_{out}) in the hybrid fitness. Based on these experimental results we can conclude that globally the hybrid approaches perform better, and the weights in the clustering coefficient do not greatly affect the outcome.

5.2.2. *Comparison of Fitness Functions for K-fixed and K-adaptive Algorithms*

Using the previous experimental conclusions, the next experiments will be executed using the K-adaptive algorithm and the clustering coefficient (without weights). Fig. 9 shows experimental results, comparing both versions of the algorithm. Fitness functions labelled with an asterisk represent the results for the K-adaptive algorithm.

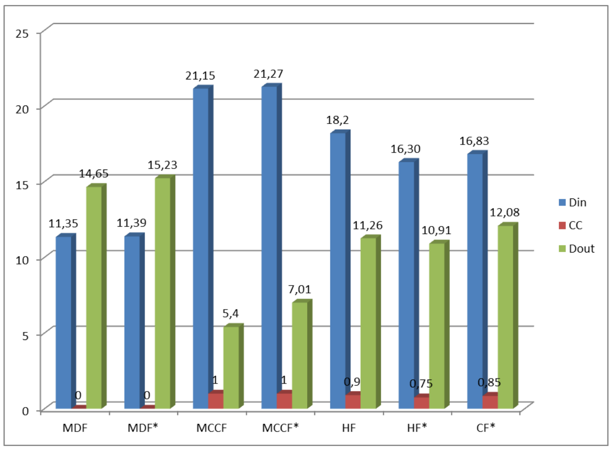

Fig. 9. Values of the clustering coefficient and the distances d_{in} and d_{out} obtained using the designed fitness functions with both versions of the algorithm. The fitness functions labelled with an asterisk show the values for the K-adaptive algorithm.

As we can seen in the previous figure, the first two fitness functions, (MDF and MDF^*), take the minimum (d_{in}) distance and the maximal d_{out} distance, but the value of the CC is 0 in both cases. It means that the members of the communities are not connected between them.

In the next two functions (MC^{2}F, MC^{2}F*) the opposite situation is encountered. The maximum possible value of CC is reached, but the distance measures get dramatically worse.

Both approaches have been combined in new hybrid fitness functions (HF, HF^*) that try to find new communities with better values for all the considered measures. Fig. 9 shows the distance between centres (d_{out}) and the intra-cluster distance (d_{in}), take values lying between the first and second functions. Finally, the clustering coefficients (0,9 and 0,75 respectively) are closer to the values obtained

Algorithm	K-fixed	K-Adaptive
Mutation probability	0.2	0.03
Generations	2500	500
Population size	3000	1000
Selection criteria $(\mu + \lambda)$	$3000 + 300$	$1000 + 100$
K value	6	-

Table 1: Genetic Parameters of GCF Algorithm

	MC^{2}F	MWC^{2}F	HF	WHF
d_{in}	21,15	21,56	18,2	19,02
d_{out}	5,4	6,39	11,26	11,99
CC	1	1	0,9	1

Table 2: Values of clustering coefficient and distances d_{in} and d_{out} obtained using weighted and unweighted fitness functions for K-fixed algorithm (2009 data set).

by the second fitness functions, that obtain the maximum possible value (1).

The last fitness function considered, the centroid fitness function (CF), obtains similar results for CC and d_{in} values and improves the d_{out} distance. This expected result came from the own definition of this function, that uses the distance between centroids to determine how to build the community.

Finally, all the experimental results from these fitnesses are compared for both versions of the algorithm. It can be noticed that the K-adaptive algorithm obtains similar or better results than the K-fixed algorithm in all the cases. Therefore, the CF function has been selected to experimentally test our community finding approach against other community finding algorithms.

5.3. ***Experimental Results***

5.3.1. *Comparative between algorithms*

In this section, we will compare the different algorithm results that we have gotten in a previous work [50] with the new algorithm results. In this previous study, the periods which we had been considered most representative were:

- 1992-1996: Jury-based voting system was used exclusively.
- 2004-2008: Televoting was used exclusively, as well as having a semi-finals round.

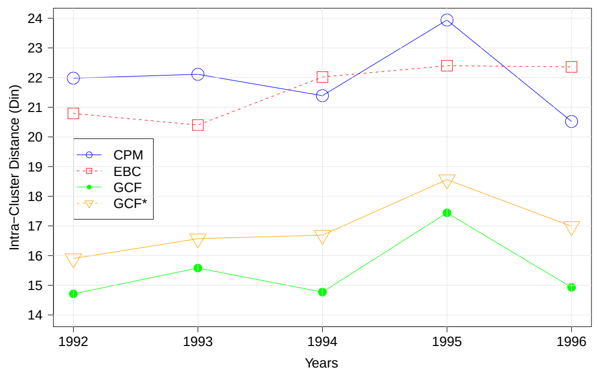

Fig. 10. d_{in} comparison of the Eurovision Contest Song between 1992 and 1996.

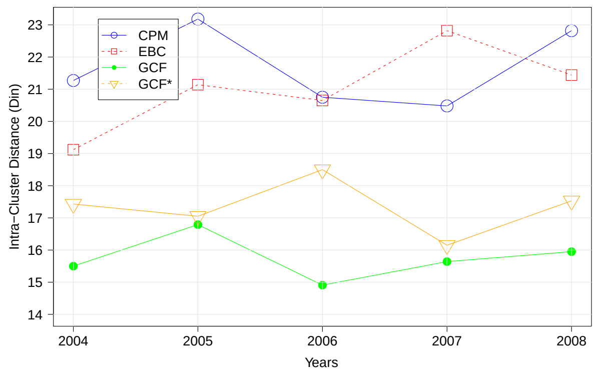

Fig. 11. d_{in} comparison of the Eurovision Contest Song between 2004 and 2008.

As we can see in these results, the d_{in} measure is minimized by both genetic algorithms, however the first version of the algorithm obtains better results. The new approach has close results and there is a big gap between the genetic algorithms and EBC or CPM. It means that the nodes of the GCF algorithms have more similar nodes than EBC and CPM.

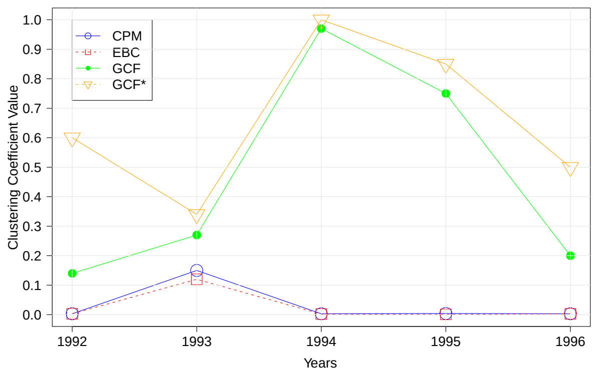

Fig. 12. CC comparison of the Eurovision Contest Song between 1992 and 1996.

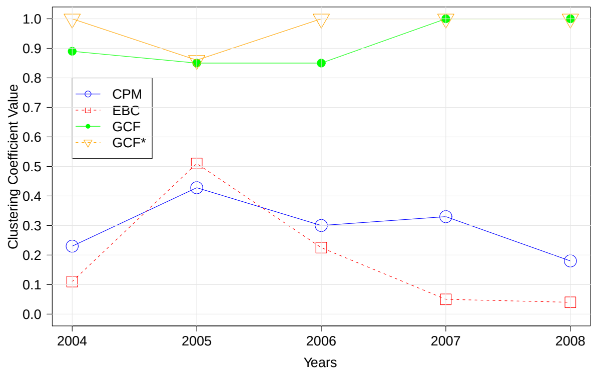

Fig. 13. CC comparison of the Eurovision Contest Song between 2004 and 2008.

The above figure shows the CC measure results, and as we can observe, its value is maximized by both genetic algorithms. In this case the new genetic algorithm approximation using adaptive K obtains the best results, followed by the first version of the algorithm with fixed K. The EBC and CPM algorithms obtain the worst CC results, meaning there are fewer connections between nodes within communities.

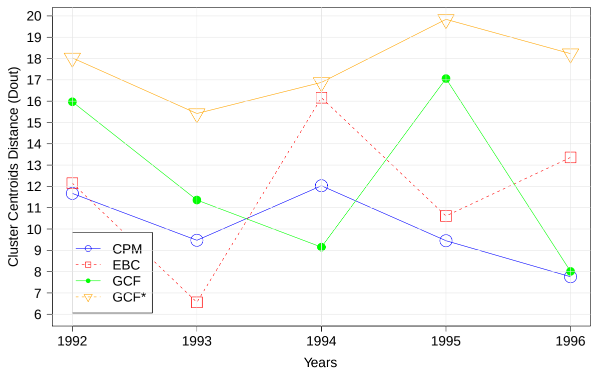

Fig. 14. d_{out} comparison of the Eurovision Contest Song between 1992 and 1996.

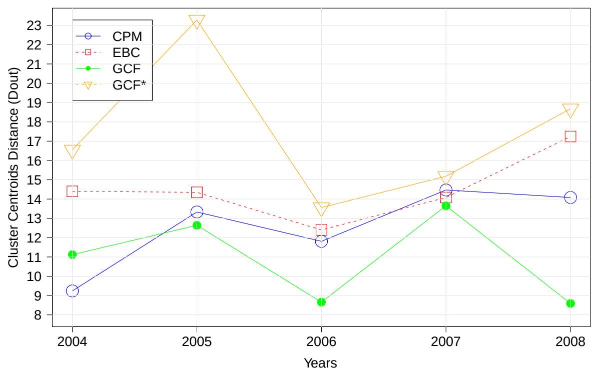

Fig. 15. d_{out} comparison of the Eurovision Contest Song between 2004 and 2008.

Regarding the d_{out} measure, we can see that it is maximized by both genetic algorithms. As in the previous case, the new genetic algorithm approximation obtains the best results. It was one of the original goals of the algorithm modification. The difference observed in cluster centroid distance between the results obtained by the first genetic approach and those through by EBC and CPM algorithms is not far too noticeable. Nonetheless, the adaptive GCF version always improves that value.

5.3.2. *Community Interpretation*

In this subsection we compare the results of the communities created by the algorithm, giving them a human interpretation. We have chosen the results of one particular year to make a more detailed analysis. The year selected is 2006.

The next four figures plot the communities found in a geographical context, where a high correlation between neighbouring countries and their membership to like communities can already be appreciated. A example of the neighbour effect is the subset conformed by Norway, Sweden and Finland, that we can see in all the maps except for the first version of GCF.

The first map is the CPM map, as Fig. 16 shows there are big communities with great overlapping, where overlapped countries are in bold. However, d_{in} and CC results show these countries do not share common features.

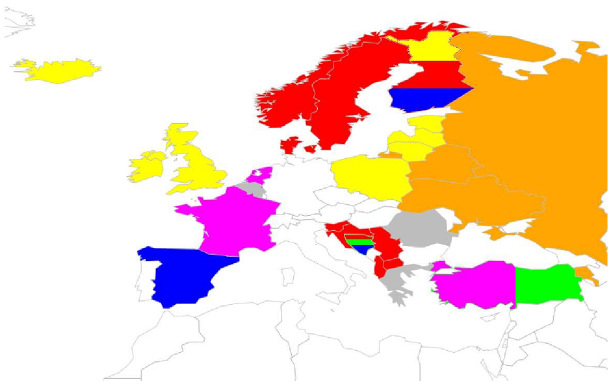
Fig. 16. CPM Cluster Results of 2006. The communities are: [Spain, **Bosnia and Herzegovina** and **Finland**], [France, Netherlands and **Turkey**], [Iceland, Ireland, United Kingdom, Poland, Lithuania, Latvia, Estonia and **Finland**], [Norway, Sweden, **Finland**, Macedonia, Albania, Serbia, **Bosnia and Herzegovina**, Croatia and Slovenia and Denmark], [Belgium, Romania and Greece], [**Turkey**, **Bosnia and Herzegovina**] and [Russia, Belarus, Ukraine, Armenia]

The EBC algorithm does not allow overlapping, but it also generates big communities whose d_{in} and CC measures take the worst values.

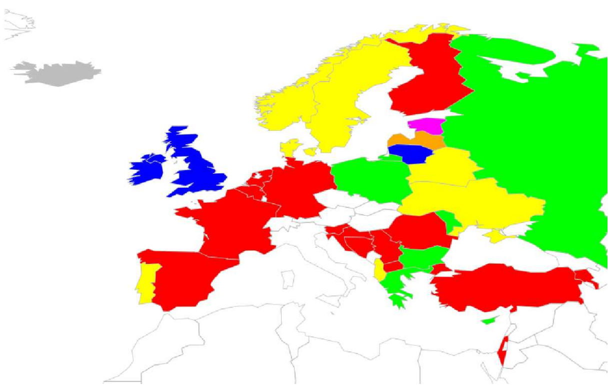
Fig. 17. EBC Cluster Results of 2006. The communities are: [Ireland, United Kingdom, Lithuania], [Estonia], [Sweden, Norway, Portugal, Ukraine, Belarus, Denmark, Albania], [Spain, France, Belgium, Germany, Netherlands, Slovenia, Croatia, Bosnia and Herzegovina, Serbia, Macedonia,Romania,Turkey], [Iceland] and [Poland, Russia, Greece, Moldavia, Bulgaria] and [Latvia]

On the other hand, the communities resulting from the genetic algorithms are smaller. This is expected if we consider that our algorithm tries to find communities whose members are highly connected between them, and also have similar characteristics.

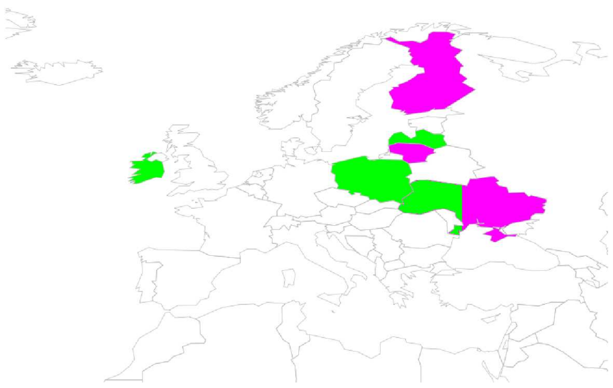
Fig. 18. GCF Cluster Results of 2006. The communities are: [Ireland,Poland, **Turkey**, Latvia] and [Finland, Lithuania, **Turkey**]

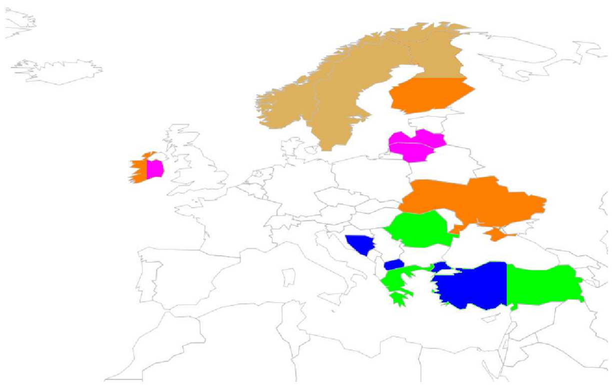
Fig. 19. New GCF (GCF*) Cluster Results of 2006. The communities are: [**Ireland**, **Finland**, Ukraine], [**Turkey**, Macedonia, Bosnia and Herzegovina], [Lithuania, Latvia and **Ireland**], [Sweden, Norway and **Finland**], [Greece, **Turkey** and Romania]

Once the found members of the new GCF communities are analysed, an important issue appears immediately. For each GCF* community most of their countries, or a subset of them, are contained

GCF* community	**CPM related community**
Ireland, **Finland**, Ukraine	Iceland, **Ireland**, United Kingdom, Poland, Lithuania Latvia, Estonia, **Finland**
Turkey, **Macedonia**, **Bosnia and Herzegovina**	Norway, Sweden, Finland, **Macedonia**, Albania, Serbia **Bosnia and Herzegovina**, Croatia, Slovenia, Denmark
Lithuania, **Latvia**, **Ireland**	Iceland, **Ireland**, United Kingdom, Poland,**Lithuania** **Latvia**, Estonia, Finland
Sweden, **Norway**, **Finland**	**Norway**, **Sweden**, **Finland**, Macedonia, Albania, Serbia Bosnia and Herzegovina, Croatia, Slovenia, Denmark
Greece, Turkey, **Romania**	Belgium, **Romania**, **Greece**

Table 3: Comparative analysis between new GCF (GCF*) communities and their related CPM communities of 2006. The GCF* countries that are contained in a CPM community appears in bold.

in other community found by the CPM algorithm. Table 3 shows for each community obtained using our genetic algorithm, the related CPM community which contains it. For example, this is the case of the community formed by Sweden, Norway and Finland or the formed by Lithuania, Latvia and Ireland shown in this table.

5.4. *Experimental Conclusions*

To improve the results of the GCF algorithm we have developed the k-adaptive GCF which also maximizes the centroid distances of the clusters. Comparing both approximations with CPM and EBC we discover that the communities defined by the genetic algorithms are smaller than the communities defined by CPM and EBC. It is important to observe that each community generated by the genetic algorithm is contained, or partially contained, in a community generated by the CPM algorithm. It means that the genetic algorithm has tuned up the original community definition of this classical algorithm.

6. Conclusions

To create an overlapped graph clustering algorithm we have focused our research in genetic algorithms. We have developed an algorithm where the number of clusters is adaptive instead of predefined. To guide the algorithm to its goals, we defined the fitness functions. In our solution the fitness functions have been inspired by complex network analysis specially focused in the clustering coefficient measure. The fitness functions also consider the quality of the clusters minimizing the distance between the elements which belong to a cluster and maximizing the cluster centroid distance.

Our experimental findings show that, using this new approach, it is able to reach better results than the other classical approaches studied. The GCF algorithm finds communities that have an appropriate size, reduced overlapping and closer distances between clusters.

Finally some improvements can be made in the the algorithm. Our future work is focused on network evolution. We are interested in dynamical network behaviour. Also, for the Eurovision dataset, other features such as geographical distances or historical behaviours could be included in future fitness functions to study the analysis of the GCF algorithm.

7. Acknowledgements

This work has been partly supported by: Spanish Ministry of Science and Education under project TIN2010-19872 and the grant BES-2011-049875 from the same Ministry.

References

1. Eurovision song contest, 2011. http://www.eurovision.tv.
2. K. Adamska. Cluster analysis of genetic algorithms results. *Inteligencia Artificial, Revista Iberoamericana de IA*, 9(28):25–32, 2005.
3. H. Adeli and K. Sarma. *Cost Optimization of Structures: Fuzzy Logic, Genetic Algorithms, and Parallel Computing.* John Wiley & Sons, 2006.

4. Jose Aguilar. Resolution of the clustering problem using genetic algorithms. *International Journal of Computers*, 1(4):237 – 244, 2007.
5. 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.
6. Francis Bach and Michael Jordan. Learning Spectral Clustering, With Application To Speech Separation. *Journal of Machine Learning Research*, 7:1963 – 2001, October 2006.
7. Sanghamitra Bandyopadhyay. Genetic algorithms for clustering and fuzzy clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(6):524–531, 2011.
8. Wesam Barbakh and Colin Fyfe. Clustering with reinforcement learning. In Hujun Yin, Peter Tino, Emilio Corchado, Will Byrne, and Xin 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.
9. Wesam Barbakh and Colin Fyfe. Online clustering algorithms. *International Journal of Neural Systems*, 18(3):185–194, 2008.
10. 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.
11. Gema Bello, Héctor Menéndez, and David Camacho. Using the clustering coefficient to guide a genetic-based communities finding algorithm. In Hujun Yin, Wenjia Wang, and Victor 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.
12. James C. Bezdek, James Keller, Raghu Krisnapuram, and Nikhil Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing (The Handbooks of Fuzzy Sets)*. Springer, 1 edition, March 2005.
13. Yi-Yuan Chen and Kuu-Young Young. An som-based algorithm for optimization with dynamicc weight updating. *International Journal of Neural Systems*, 17(3):171 – 181, 2007.
14. Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111+, December 2004.
15. Rowena M. Cole. Clustering with Genetic Algorithms. Master's thesis, Nedlands 6907, Australia, 1998.
16. Coley. *An Introduction to Genetic Algorithms for scientists and engineers.* World Scientific Publishing, 1999.
17. 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.
18. M. Dehmer, editor. *Structural Analysis of Complex Networks*. Birkhäuser Publishing, 2010. in press.
19. 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.
20. Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique Percolation in Random Networks. *Physical Review Letters*, 94(16):160202–1 – 160202–4, Apr 2005.
21. 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.
22. 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.
23. Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(5):056104, 2004.
24. 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.
25. Colin Fyfe. Topographic maps for clustering and data visualization. In John Fulcher and L. Jain, editors, *Computational Intelligence: A Compendium*, volume 115 of *Studies in Computational Intelligence*, pages 111–153. Springer Berlin - Heidelberg, 2008.
26. Colin Fyfe and Wesam Barbakh. Immediate reward reinforcement learning for clustering and topology preserving mappings. In Michael Biehl, Barbara Hammer, Michel Verleysen, and Thomas Villmann, editors, *Similarity-Based Clustering*, volume 5400 of *Lecture Notes in Computer Science*, pages 35–51. Springer Berlin - Heidelberg, 2009.
27. Derek Gatherer. Comparison of eurovision song contest simulation with actual results reveals shifting patterns of collusive voting alliances. *Journal of Artificial Societies and Social Simulation*, 9(2):1, 2006.
28. M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, June 2002.
29. 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.
30. 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.
31. Erez Hartuv and Ron Shamir. A clustering algorithm

based on graph connectivity. *Information Processing Letters*, 76(4–6):175–181, 2000.

32. 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.
33. 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.
34. Jose Antonio Iglesias, Plamen Angelov, Agapito Ledezma, and Araceli Sanchis. Human activity recognition based on evolving fuzzy systems. *International Journal of Neural Systems*, 20(5):355 – 364, 2010.
35. 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.
36. 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.
37. G. N. Lance and W. T. Williams. A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. *The Computer Journal*, 9(4):373–380, February 1967.
38. 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.
39. Daniel T. Larose. *Discovering Knowledge in Data*. John Wiley and Sons, 2005.
40. 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.
41. Jianzhuang Liu and Weixin Xie. A genetics-based approach to fuzzy clustering. In *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE International Conference on*, volume 4, pages 2233–2240 vol.4, 1995.
42. 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.
43. U Maulik. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.
44. Boaz Nadler, Stéphane Lafon, Ronald Coifman, and Ioannis G. Kevrekidis. Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators. pages 955 – 962, 2005.
45. 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.
46. 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.
47. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.
48. 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.
49. Alberto Ochoa Ortíz, Angel E. Muñoz Zavala, and Arturo Hernández Aguirre. A hybrid system using pso and data mining for determining the ranking of a new participant in eurovision. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1713–1714, New York, NY, USA, 2008. ACM.
50. Gema Bello Orgaz, Raul Cajias, and David Camacho. A study on the impact of crowd-based voting schemes in the 'eurovision' european contest. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS '11, pages 25:1–25:9, New York, NY, USA, 2011. ACM.
51. M. Oussalah and Samia Nefti. On the use of divergence distance in fuzzy clustering. *Fuzzy Optimization and Decision Making*, 7:147–167, June 2008.
52. Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.
53. 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.
54. Riccardo Poli and William B. Langdon. Backward-chaining evolutionary algorithms. *Artificial Intelligence*, 170(11):953 – 982, 2006.
55. P. Pons and M. Latapy. Computing communities in large networks using random walks (long version). *ArXiv Physics e-prints*, December 2005.
56. Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006.
57. 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.

58. K. Sarma and H. Adeli. Fuzzy genetic algorithm for optimization of steel structures. *Journal of Structural Engineering*, 126(5):596–604, 2000.
59. Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
60. 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.
61. Liangdong Shi, Yinghuan Shi, Yang Gao, Lin Shang, and Yubin Yang. Xcsc: a novel approach to clustering with extended classifier system. *Int. J. Neural Syst.*, 21(1):79–93, 2011.
62. 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.
63. Lin Yu Tseng and Shiueng Bien Yang. A genetic approach to the automatic clustering problem. *Pattern Recognition*, 34(2):415 – 424, 2001.
64. 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.
65. Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.
66. Ulrike von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586, April 2008.
67. 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.
68. D. J. Watts. *Small worlds : the dynamics of networks between order and randomness.* 1999.
69. Wojciech and Kwedlo. A clustering method combining differential evolution with the k-means algorithm. *Pattern Recognition Letters*, 32(12):1613 – 1621, 2011.