A Genetic Algorithm for Community Detection in Attributed Graphs

Clara Pizzuti[0000−0001−7297−7126] and Annalisa Socievole[0000−0001−5420−9959]

Institute for High Performance Computing and Networking (ICAR), National Research Council of Italy (CNR), via P. Bucci 7/11C, 87036, Rende (CS), Italy, {clara.pizzuti,annalisa.socievole}@icar.cnr.it

Abstract. A genetic algorithm for detecting a community structure in attributed graphs is proposed. The method optimizes a fitness function that combines node similarity and structural connectivity. The communities obtained by the method are composed by nodes having both similar attributes and high link density. Experiments on synthetic networks and a comparison with five state-of-the-art methods show that the genetic approach is very competitive and obtains network divisions more accurate than those obtained by the considered methods.

Keywords: attributed networks, community detection, genetic algorithms, complex networks

1 Introduction

Complex networks constitute one of the main formalisms to model and study relationships of real-world systems. Networks have been mainly studied at the level of interactions among nodes, i.e. with respect to their structure. However, nodes are often endowed with a set of characteristics [1] such as work, gender, hobbies, age, and race. In online social networks, for example, people publish data regarding their personal profile, thus providing important information for analyzing relationships and properties of the social systems they participate. *Attributed graphs* extend network models by enriching nodes and/or edges with a set of features that measure the characteristics of the actors and/or the strength or type of links. As pointed out in [2], when attributes are related to nodes, attributed graphs are referred to as *node-attributed graphs*, while when related to edges, they are called *edge-attributed graph*. In this paper, we deal only with node-attributed graphs.

Because of the wealth of data available on social networks, the simultaneous analysis of the topological structure and the characteristics of the objects composing a network has received, in the last years, the interest of researchers for finding communities in complex networks. In fact, it has been observed that in real-world social systems there exists a correlation between attribute values and connectivity [3], and that the *homophily* effect, for which individuals are more likely to create relationships with others having similar attribute values, and the *social influence* effect, for which people tend to modify their behavior to be akin

to their friends, often co-occurr. Thus, the utilization of the information coming from both attributes and links can be beneficial to methods for community detection to obtain groups of nodes not only densely connected, but also having similar characteristics.

In the last year, several methods for detecting communities in attributed graphs have been proposed. A recent survey of Bothorel et al. [2] gives a detailed description of the most recent state-of-the-art algorithms. Approaches to find communities in attributed graphs, according to Bothorel et al. [2], can be classified into different categories, depending on the strategy adopted. One category uses a function to compute the similarity between couples of nodes, and then reduces the network to a weighted graph. At this point any community detection method for weighted graphs can be used. Neville et al. [4], for instance, define a similarity measure that computes the number of attribute values two nodes have in common. They compare three existing graph-partitioning techniques and show that a spectral clustering approach outperforms the others. The main drawback of these methods is that the number of groups to obtain must be given as input parameter. Cruz et al. [5] obtain the node similarity by grouping nodes with a self-organizing map [6] that takes into account the similarity between the features. The Louvain method [7] is then used to find the communities of the weighted graph. Another category of approaches combines structural and attribute similarity, and applies clustering methods to nodes with the combined similarity. Combe et al. [8] define a distance measure between two nodes as the sum of the attribute distance, computed for the features with any measure such as the Euclidean or the cosine distance, and a structural distance given by the shortest path between such nodes. A hierarchical agglomerative clustering is then applied on the computed distance matrix. The *unified distance measure* proposed by Papadopoulos et al. [9], extensively described in the next, follows the same principle. The authors formalize the problem as an optimization fuzzy clustering problem with an objective function that assigns different weights to edges and attributes, computed iteratively with the gradient descent technique during the clustering process. Dang and Viennet [10] extend the modularity concept [11] to include the similarity between node attributes. Zhou et al. [12] builds an attribute augmented graph by adding to the initial graph new vertices representing the attributes. Elhadi and Agam [13] propose an algorithm that uses either the structure data, or the attribute data depending on the type of graph, and then executes the *Louvain* method in the former case, and the *k-means* in the latter case.

In this paper, we propose a method for clustering attributed graphs, named @NetGA, based on *Genetic Algorithms (GAs)*, that optimizes a fitness function derived from the *unified distance measure* of Papadopoulos et al. [9], combining node similarity and structural connectivity. The communities obtained by the method are composed by nodes having both similar attributes and high link density. Experiments on synthetic networks and a comparison with five state-of-the-art methods show that the genetic approach is very competitive and obtains network divisions more accurate than those obtained by the considered methods.

The paper is organized as follows. In the next section we give preliminary definitions. In Section 3 the fitness function is introduced and the algorithm @NetGA is described in detail. Section 4 describes the synthetic networks used for evaluating the methods, the algorithms with which @NetGA has been compared, the evaluation measures adopted to perform the comparison, and the results obtained by all the methods. Section 5, finally, concludes the paper and discusses future developments.

2 Problem Definition

In this section we give the definition of attributed graph and the community detection problem for these kind of graphs.

Definition An *attributed graph* is a 4-tuple $G = (V, E, A, F)$ where $V = \{v_1, v_2, ..., v_N\}$ is a set of N vertices, $E \subseteq V \times V$ is a set of M edges, $A = \{\alpha_1, \alpha_2, ..., \alpha_{\mathcal{A}}\}$ is the set of numerical and categorical attributes (features), and $F = \{a_1, a_2, ..., a_{\mathcal{A}}\}$ is a set of functions. Each node $v \in V$ is characterized by a vector of feature values, obtained by the functions $a_\alpha : V \to D_\alpha$, $1 \leq \alpha \leq \mathcal{A}$, with D_α the domain of attribute α.

The objective of community detection in attributed graphs is to find a partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ of the nodes of V such that

- intra-cluster density is high and inter-cluster density is low, and
- nodes belonging to the same community are similar, while nodes of different communities are quite dissimilar.

3 @NetGA Description

The *GA* method we propose minimizes a fitness function based on the *unified distance measure*, introduced by Papadopoulos et al. [9], that takes into account both the graph structure and the attributes. We first recall the definition of this distance measure and then we define our fitness function. Given an attributed graph $G(V, E, A, F)$, the *similar connectivity* measures how dissimilar two vertices are with respect to all their outgoing edges as:

$$SC(i, j) = \frac{1}{N} \sum_{k=1}^{N} [w(i, k) - w(j, k)]^2 \tag{1}$$

where

$$w(i, j) = \begin{cases} 1 & if\ (i = j)\ or\ (i, j) \in E \\ 0 & otherwise \end{cases} \tag{2}$$

The *attribute distance* between two nodes measures their dissimilarity with respect to their attribute values. It is computed as:

$$AD(i, j) = \sum_{\alpha \in A} W_\alpha \cdot \delta_\alpha(i, j), \quad \sum_{\alpha \in A} W_\alpha = 1 \tag{3}$$

where W_α is a weight corresponding to the importance of attribute α, and $\delta_\alpha(i,j)$ is the attribute distance between nodes i and j for attribute α. For numerical attributes scaled in the interval [0, 1], $\delta_\alpha(i,j) = [a_\alpha(i) - a_\alpha(j)]^2$, while, for the categorical attributes

$$\delta_\alpha(i,j) = \begin{cases} 1 & if \quad a_\alpha(i) = a_\alpha(j) \\ 0 & otherwise \end{cases}. \tag{4}$$

The *unified distance measure (udm)* balances with appropriates weights the structural and attribute properties by combining *attribute distance* (AD) and *similar connectivity* (SC) between two nodes i and j as follows

$$d(i,j) = W_{attr} \cdot AD(i,j) + W_{links} \cdot SC(i,j) \tag{5}$$

where W_{attr} and W_{links} are weights representing the importance of attributes and edges, respectively.

Given a network division $\mathcal{C} = \{C_1, \ldots, C_k\}$, we define the *clustering unified distance measure* $cudm(\mathcal{C})$ of the solution $\mathcal{C}$ by computing for each $C_i \in \mathcal{C}$, $1 \leq i \leq k$, the *udm* of pairs of nodes belonging to C_i, and then averaging the results with respect to the number k of obtained communities:

$$cudm(\mathcal{C}) = \frac{1}{k} \sum_{C \in \mathcal{C}} \sum_{\{i,j\} \in C \ i \neq j} d(i,j) \tag{6}$$

where k is the number of communities of the solution $\mathcal{C}$, i and j are nodes of a community $C \in \mathcal{C}$ and $d(i,j)$ is the *unified distance measure* between nodes i and j.

The @NetGA method, thus, minimizes the *cumd* measure to obtain a community division that takes into account both the similarity of node features, as well the connections shared by pairs of nodes inside the network structure. Together with the *cumd* as fitness function, @NetGA uses the locus-based adjacency representation [14], uniform crossover and neighbor-based mutation. In the locus-based representation, an individual of the population is represented through a vector of n genes assuming values in the range $\{1, \ldots, n\}$. A value j assigned to the ith gene means that there is a link between the nodes i and j. A decoding step identifies the connected components of the graph corresponding to the network division in communities. Uniform crossover generates a random binary vector of length N, then an offspring is obtained by selecting from the first parent the genes where the value is 0, and from the second parent the genes where the value is 1. Finally, the neighbor-based mutation operator randomly changes the value j of a i-th gene with one of its neighbors.

@NetGA receives in input the graph $G = (V, E, A, F)$, the weighting factors W_{attr} and W_{links} to assign a score to attributes and links, respectively, an importance weight to each attribute W_α, and performs the following steps:

1. run the Genetic Algorithm on G for a number of iterations by using *cumd* as fitness function to minimize, uniform crossover and neighbor mutation as variation operators;

2. obtain the partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ corresponding to the solution with the lowest fitness value $cumd(\mathcal{C})$;
3. merge two communities if the number of inter-cluster connections is higher than the number of intra-cluster connections.

In the next section, we execute @NetGA on a number of synthetic networks and compare it with other state-of-the-art methods.

4 Experimental Evaluation

To validate the effectiveness of @NetGA, we performed several simulations on synthetic networks and compared the results with those obtained by other five methods. The algorithm has been implemented in Matlab 2015b by using the Global Optimization Toolbox. Since finding a balanced weight for attributes and links is not our aim, differently from Papadopoluos et al. [9], for each simulation, we fixed equal weight to attributes and links, thus setting $W_{attr} = W_{links} = 0.5$, and also $W_\alpha = 1/\mathcal{A}, \forall\alpha$. In the following, we describe the synthetic datasets used, the contestant algorithms, the evaluation measures employed to assess the quality of the methods, and the results obtained.

4.1 Datasets

We generated a set of synthetic datasets using the *LFR-EA* benchmark proposed by Elhadi and Agam [13], which is an extension of the *LFR* benchmark by Lancichinetti *et al.* [15].

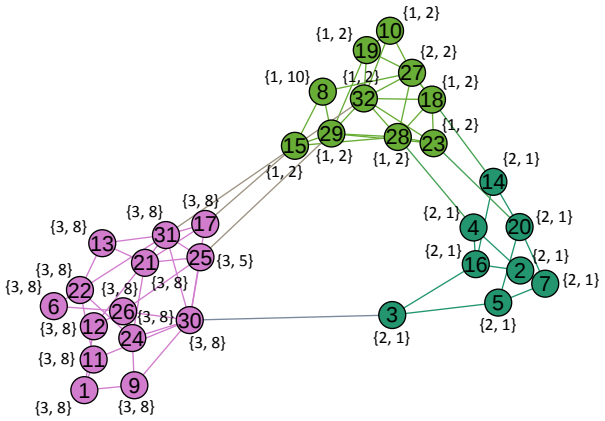

Fig. 1. Network structure and ground truth for the LFR-EA-32 dataset with $\mu = 0.1$ and $\nu = 0.1$: the 32 nodes are partitioned into 3 distinct communities.

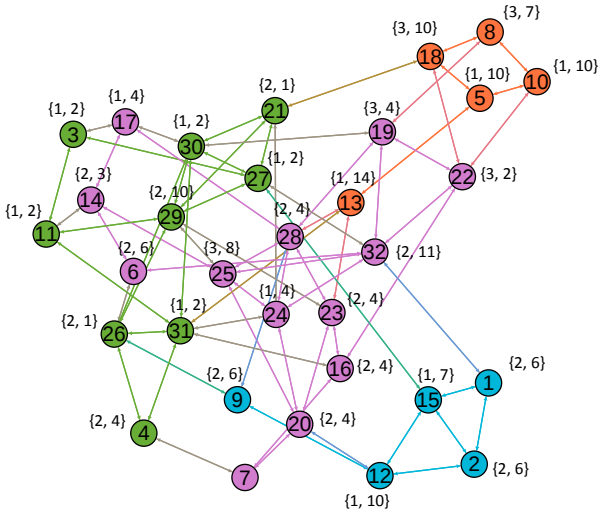

Fig. 2. Network structure and ground truth for the LFR-EA-32 dataset with $\mu = 0.3$ and $\nu = 0.5$: the 32 nodes are partitioned into 4 distinct communities.

The generator uses two parameters μ and ν, both ranging in the interval [0.1, 0.9], to control the structure and the attribute values, respectively. μ is called *mixing parameter* and determines the rate of intra- and inter-communities connections. Low values of μ give a clear community structure where intra-cluster links are much more than inter-cluster links. Analogously for ν, called *attribute noise*, low values generate similar features of nodes belonging to the same community. Besides ν, the number of attributes and the size of the domain D_α of each attribute α must be specified. The combination of μ and ν values produces graphs with a clear to ambiguous structure and/or attributes. To better understand the kind of networks that can be generated, Figures 1 and 2 show two examples of synthetic networks with 32 nodes. The network of Fig. 1 has both clear structure and attributes ($\mu = 0.1$ and $\nu = 0.1$), while the network of Fig. 2 has a less clear structure and attributes with medium similarity ($\mu = 0.3$ and $\nu = 0.5$).

The parameters used to generate the *LFR-EA* datasets are shown in the Table 1. We created networks with 1000 nodes (LFR-EA-1000) by setting 2 numerical attributes for each node. All the nodes in a community, in particular, share the same attribute domain values. The attribute's domain cluster assignment is set to random selection without replacing, in order to cover all the domain values across the different communities. We generated ten different instances of the combination of μ and ν parameters reported in Table 1, and executed @NetGA by fixing the population size to 300 individuals, the number of generations to 200, a mutation rate of 0.4, and a crossover fraction of 0.8. These genetic parameters have been selected with a trial-and-error procedure and choosing the values giving the best performance of the algorithm.

Table 1. LFR-EA-1000 parameters setting.

Parameter	Value
Number of nodes (N)	1000
Average degree (k)	25
Maximum degree ($maxk$)	40
Exponent for the degree distribution ($t1$)	2
Mixing parameter (μ)	[0.1; ...; 0.9]
Exponent for the community size distribution ($t2$)	1
Minimum for the community sizes ($minc$)	60
Maximum for the community sizes ($maxc$)	100
Number of overlapping nodes (on)	0
Number of memberships of the overlapping nodes (om)	0
Number of attributes (T)	2
Attribute's domain cluster assignment ($ainf$)	1
Attribute # 1 domain size	3
Attribute # 1 noise	[0.1; 0.5; 0.9]
Attribute # 2 domain size	15
Attribute # 2 noise	[0.1; 0.5; 0.9]

4.2 Algorithms in Comparison

We compared @NetGA to five types of algorithms: (1) structure-only, i.e. a classical community detection method that does not consider the attributes, (2) attribute-only, i.e. a method that uses only node similarity, (3) composite, i. e. that builds an attribute augmented graph, (4) ensemble, i.e. that combines different clustering results, and (5) selection, i.e. that decides which method to use depending on the graph. In the following, we briefly summarize these algorithms.

- **Louvain** [7] (structure-only) aims at optimizing the modularity [11] of a partition using a greedy technique. First, the method searches small communities locally optimizing modularity. Then, each community found is considered a node and modularity-based community detection is applied again until a hierarchy of high-modularity communities is obtained.
- **k-means** [16] (attribute-only) is considered one of the most famous clustering algorithms. Data points are randomly assigned to a number k of clusters. Then, the centroid of each cluster is computed and every data point is assigned to its closest centroid. These steps are repeated until there are not assignments of data points to clusters, and a stopping criterion is reached.
- **SA-Cluster** [12] (composite) builds an attribute augmented graph by adding to the initial graph new vertices representing the attributes. An edge between a graph vertex and an attribute vertex is present if the graph vertex has that attribute and the edge weight between them reflects the importance of that attribute. The method uses the neighborhood random walk model on the attributed augmented graph to compute a unified distance measure between vertices (i.e., combination of structural closeness and attribute similarity).

– **CSPA** [13] (ensemble) is a modified version of the *Cluster-based Similarity Partitioning Algorithm* of Strehl and Gosh [17] that combines *Louvain* and the *k-means* cluster labels through a cluster ensemble. A cluster ensemble solves the clustering problem in two steps. In the first step, a data set is taken as input and an ensemble of clustering solutions is generated as output. In the second step, the cluster ensemble is taken as input and these solutions are combined to produce a single clustering as the final output. $CSPA$ uses binary similarity matrices for representing the similarity between objects in the same cluster. Through these similarity matrices, $CSPA$ establishes a pairwise similarity measures and realizes a combined clustering.
– **Selection** [13] (selection), instead of combining the structure and the attribute data, this method makes the choice to use either the structure data, or the attribute data depending on the type of graph (clear or ambiguous structure). It detects the boundaries between clear and ambiguous graph structure content and applies the structure-only method of *Louvain* when the graph has a clear structure, while the *k-means* attribute-only method when the graph has an ambiguous structure.

4.3 Evaluation Measures

To assess the quality of the solutions, we use the following evaluation measures.

– **Normalized Mutual Information (NMI)**. The normalized mutual information $NMI(A, B)$ [18] of two divisions A and B of a network is defined as follows. Let C be the confusion matrix whose element C_{ij} is the number of nodes of community i of the partition A that are also in the community j of the partition B.

$$NMI(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} C_{ij} log(C_{ij} n / C_{i.} C_{.j})}{\sum_{i=1}^{c_A} C_{i.} log(C_{i.}/n) + \sum_{j=1}^{c_B} C_{.j} log(C_{.j}/n)} \tag{7}$$

where c_A (c_B) is the number of groups in the partition A (B), $C_{i.}$ ($C_{.j}$) is the sum of the elements of C in row i (column j), and n is the number of nodes. If $A = B$, $NMI(A, B) = 1$. If A and B are completely different, $NMI(A, B) = 0$.
– **Cumulative NMI (CNMI)**. $CNMI$ [13] is a modified NMI measure allowing the integration of NMI values over different settings of structure mixing parameter μ and attribute noise ν:

$$CNMI = \frac{\sum^{\mu}\sum^{\nu} NMI}{S} \tag{8}$$

where S is the number of samples of the network graphs considered.

4.4 Results

Fig. 3 shows the results obtained by @NetGA and the methods described in the previous section for the experiments conducted on the LFR-EA-1000 datasets.

Each subplot refers to a value of the mixing parameter μ ranging from 0.1 to 0.9, with three degrees of attribute noise (0.1: low, 0.5: medium, 0.9: high) reported on the x-axis and the corresponding NMI values on the y-axis.

The *Louvain* algorithm, being a structure-only method, obtains rather stable values of the NMI, independently from the ν values. For $0.1 \leq \mu \leq 0.4$, the network graph has a clear structure and the method is able to correctly identify the underlying communities. As the mixing parameter increases, the NMI value sensibly decreases, especially for $0.7 \leq \mu \leq 0.9$. For these mixing parameter values, the normalized mutual information is below 0.2.

The NMI values returned by the *k-means* method, since using only the attributes, are not influenced from the network structure. It is able to find communities with an NMI value medium-high only when the graph attributes are clear. Differently from Louvain, the *k-means* method is not able to match the ground-truth with good NMI values. The highest value of 0.75 is reached only when the attribute noise is 0.1.

SA-Cluster performs the worse in our settings. Even if it uses both structure and attributes, it is not able to correctly identify the communities. The NMI values it obtains are between 0.5 and 0.6 for $\mu = 0.1$ and all the three ν values. It reduces below 0.2 for $\mu \geq 0.5$.

$CSPA$, combining *Louvain* and *k-means* through cluster ensemble, performs better than SA-Cluster. However, the low NMI values of *k-means* in medium-high attribute noise situations influence the high NMI *Louvain* values in situations of low mixing parameter. Thus, the resulting NMI value of $CSPA$ sensibly decreases with the ensemble.

The *Selection* method, being driven by both attributes and structure, performs better than the previous methods. It obtains an NMI value equal to 1 for $0.1 \leq \mu \leq 0.5$ for all the attribute noise settings. When the structure of the graph becomes less clear ($0.7 \leq \mu \leq 0.9$), it is able to properly find the boundary between clear and ambiguous graph structure content. By exploiting the attribute-based clustering through *k-means*, *Selection* obtains an NMI value around 0.75 when $\nu = 0.1$. However, for $\nu = 0.5$ and $\nu = 0.9$ these values drastically decrease below 0.3 and 0.2, respectively.

@NetGA is able to achieve very high NMI values for all the mixing parameter values. In particular, for high μ values, @NetGA outperforms all the other algorithms for all the attribute noise values considered. Moreover, as Table 2 shows, @NetGA achieves the highest $CNMI$ value, obtained by averaging the NMI values for all the attributes and structure settings, compared to the other methods considered. @NetGA, for this cumulative metric, reaches 0.98, while the *Selection* method, which is the best performing method when compared to the other attribute and structure based contestant methods, achieves only 0.77. As such, @NetGA is able to better exploit both the attributes and the structure of the graph on all the settings considered.

Fig. 3. NMI results of the evaluated methods on the LFR-EA-1000 datasets.

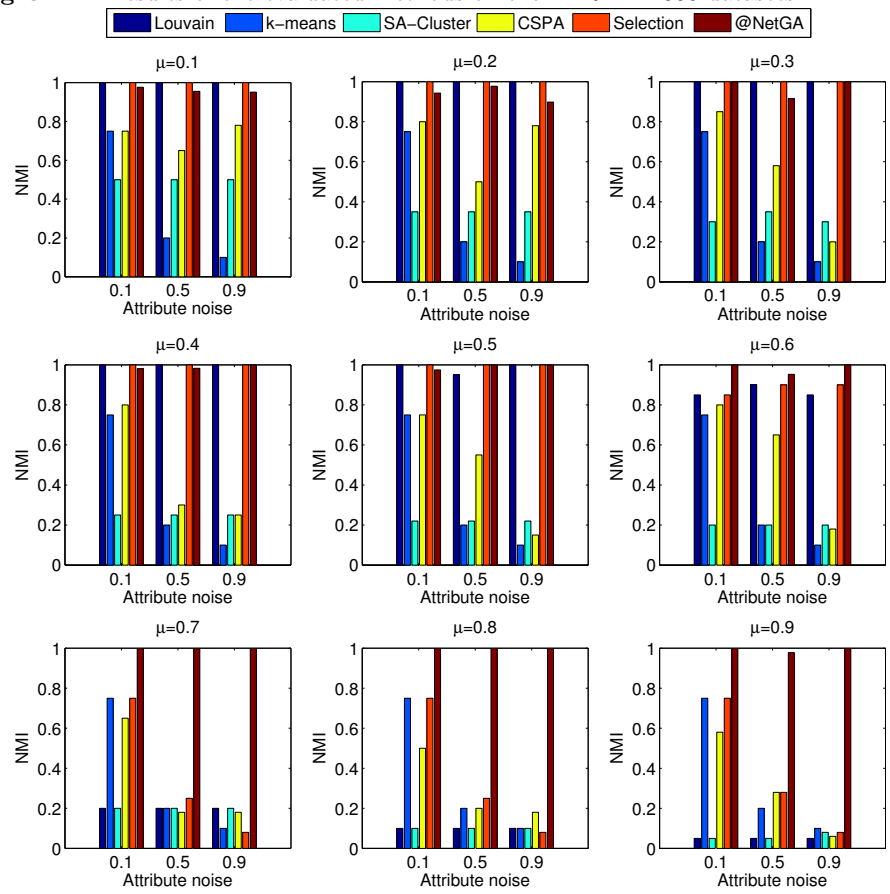

Table 2. Cumulative NMI of the methods on the LFR-EA-1000 datasets.

Method	**CNMI**
Louvain	0.69
k-means	0.35
SA-Cluster	0.24
CSPA	0.49
Selection	0.77
@NetGA	0.98

5 Conclusion

Genetic Algorithms, in the last years, showed to be a valid approach for the detection of community structure in complex networks. The method we proposed for attributed graphs confirms their ability also in this kind of networks, where nodes are characterized by a set of features. This additional information can be very important, combined with the topological structure, to detect relevant communities that share common characteristics. A comparison with other five methods, finding communities with different strategies, on synthetic networks has highlighted the capability of the genetic approach to obtain more accurate divisions. Future work aims at experimenting the method on real-world attributed networks.

References

1. Wasserman, S., Faust, K.: Social network analysis: Methods and applications, vol. 8. Cambridge university press (1994)
2. Bothorel, C., Cruz, J.D., Magnani, M., Micenkova, B.: Clustering attributed graphs: models, measures and methods. Network Science 3(03), 408–444 (2015)
3. La Fond, T., Neville, J.: Randomization tests for distinguishing social influence and homophily effects. In: Proceedings of the 19th International Conference on World Wide Web. pp. 601–610. WWW '10 (2010)
4. Neville, J., Adler, M., Jensen, D.: Clustering relational data using attribute and link information. In: Proceedings of the text mining and link analysis workshop, 18th international joint conference on artificial intelligence. pp. 9–15 (2003)
5. Cruz, J.D., Bothorel, C., Poulet, F.: Semantic clustering of social networks using points of view. In: CORIA. pp. 175–182 (2011)
6. Kohonen, T., Schroeder, M.R., Huang, T.S. (eds.): Self-Organizing Maps. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edn. (2001)
7. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment 2008(10), P10008 (2008)
8. Combe, D., Largeron, C., Egyed-Zsigmond, E., Géry, M.: Combining relations and text in scientific network clustering. In: Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on. pp. 1248–1253. IEEE (2012)
9. Papadopoulos, A., Pallis, G., Dikaiakos, M.D.: Weighted clustering of attributed multi-graphs. Computing 99(9), 813–840 (2017)
10. Dang, T., Viennet, E.: Community detection based on structural and attribute similarities. In: International Conference on Digital Society (ICDS). pp. 7–12 (2012)
11. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E 69(2), 026113 (2004)
12. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proceedings of the VLDB Endowment 2(1), 718–729 (2009)
13. Elhadi, H., Agam, G.: Structure and attributes community detection: comparative analysis of composite, ensemble and selection methods. In: Proceedings of the 7th Workshop on Social Network Mining and Analysis. p. 10. ACM (2013)
14. Park, Y., Song, M.: A genetic algorithm for clustering problems. In: Proceedings of the third annual conference on genetic programming. vol. 1998, pp. 568–575 (1998)

15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical review E 78(4), 046110 (2008)
16. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28(1), 100–108 (1979)
17. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of machine learning research 3(Dec), 583–617 (2002)
18. Danon, L., Diaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment 2005(09), P09008 (2005)