A Multiobjective and Evolutionary Clustering Method for Dynamic Networks

Francesco Folino and Clara Pizzuti
Institute for High Performance Computing and Networking (ICAR)
Italian National Research Council
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
Email: {f.folino,pizzuti}@icar.cnr.it

***Abstract*—The discovery of evolving communities in dynamic networks is an important research topic that poses challenging tasks. Previous evolutionary based clustering methods try to maximize cluster accuracy, with respect to incoming data of the current time step, and minimize clustering drift from one time step to the successive one. In order to optimize both these two competing objectives, an input parameter that controls the preference degree of a user towards either the snapshot quality or the temporal quality is needed. In this paper the detection of communities with temporal smoothness is formulated as a multiobjective problem and a method based on genetic algorithms is proposed. The main advantage of the algorithm is that it automatically provides a solution representing the best trade-off between the accuracy of the clustering obtained, and the deviation from one time step to the successive. Experiments on synthetic data sets show the very good performance of the method compared to state-of-the-art approaches.**

I. Introduction

The adaptability of networks to represent many real world complex systems, including those undergoing dynamic shifts of their structure, is generating a growing interest in the study of their topological features. Networks are modeled as graphs, where nodes represent the individual objects and edges represent the interactions among these objects. Individuals in a network interact each other and exchange information by forming communities. The detection of *community structure*, i.e. the organization of nodes into groups having many connections inside the same cluster and relatively sparse connections between vertices of different communities, is a fundamental research topic in the study of complex networks. An important standpoint to analyze in networks is their dynamic behavior, i.e. the evolutions they go through over time.

Dynamic networks, in fact, capture the modifications of interconnections over time, allowing to trace the changes of network structure at different time steps. Many approaches have been proposed for the analysis and temporal evolution of dynamic networks [1], [3], [12], [13], [15], [14], [17], [21], [23], [24], [26], [25]. Some of these methods [3], [15], [24], [12] employ the concept of *evolutionary clustering*, introduced by Chakrabarti et al. in [2], to catch the evolution of clusters in temporal data.

Evolutionary clustering groups data coming at different time steps to produce a sequence of clusterings by introducing a framework called *temporal smoothness*. This framework assumes that abrupt changes of clustering in a short time period are not desirable, thus it *smooths* each community over time. Smoothness is realized by trading-off between two different criteria. The first, called *snapshot quality*, is that the clustering should reflect as accurately as possible the data coming during the current time step. The second, called *temporal cost*, is that each clustering should not shift dramatically from one time step to the successive one.

In particular, Lin et al. [15] define the snapshot cost by using the *KL-divergence* between the observed node similarity matrix at time t and an approximate community structure computed by using a mixture model. Their algorithm, named $FacetNet$, at each iteration, updates the values of the approximate structure in order to decrease the cost function. Convergence to the optimal solution is guaranteed by the monotonic decrease of the cost function. $FacetNet$ discovers communities that maximize the fit to the observed data and the temporal evolution. FacetNet, however, as observed in [12], assumes only a fixed number of communities over time.

Kim and Han [12] thus proposed an evolutionary *particle-and-density* based clustering method able to deal with a variable number of communities between different timesteps. The method introduces the concept of *nano-community* and *l-clique-by-clique (l-KK)* to discover a variable number of communities that can evolve, form, and dissolve. A nano-community captures the evolution of a dynamic network over time at particle level. A community is modeled as a dense subset of nano-communities and *l-KK*. A biclique is a complete bipartite graph such that two nodes are connected if and only if they are in different partites. Being complete, each node in a partite is connected with all the nodes in the other partite. An *l-clique-by-clique* is an extension of a biclique to a number l of bicliques. A cost embedding technique to allow temporal smoothing, and a density-based clustering method to find local clusters by optimizing the clustering modularity are proposed.

Both methods, in order to apply temporal smoothness, need an input parameter that controls the preference degree of a user with respect to either the snapshot quality or the temporal quality. The two quality functions, however, are competing. In fact, optimizing one produces a degradation of the other.

In this paper we propose a multiobjective approach, named

DYN-MOGA (*DYNamic MultiObjective Genetic Algorithms*), to discover communities in dynamic networks by employing genetic algorithms [7]. The detection of community structure with temporal smoothness, in fact, can be formulated as a *multiobjective optimization problem*. The first objective is the maximization of the snapshot quality, that measures how well the clustering found represents the data at the current time. The second objective is the minimization of the temporal cost, that measures the distance between two clusterings at consecutive timesteps. In order to maximize the snapshot quality to measure the goodness of the division in communities of a network, the concept of *community score*, introduced in [19], and proved very effective in detecting communities, is used. The higher the community score, the more dense the clustering obtained.

To minimize the temporal cost we compute the *Normalized Mutual Information* (NMI for short), a well known entropy measure in information theory that measures the similarity of two clusterings, between the community structure obtained at the current time step with that obtained at the previous one.

DYN-MOGA exploits the benefits of these two functions and discovers the communities in the network by selectively exploring the search space, without the need to know in advance the exact number of groups. This number is automatically determined by simultaneously optimizing the objectives.

Experiments on synthetic and real life networks show the capability of the multiobjective genetic approach to correctly detect communities with results competitive w.r.t. the state-of-the-art approaches.

It is worth to note that, though multiobjective evolutionary algorithms have been proposed for partitioning static graphs [5], [8], [20], their use for dynamic networks is new.

The paper is organized as follows. In the Section II the concept of dynamic network is defined and the evolutionary clustering problem is formalized. Section III formulates the community detection problem in dynamic networks as a multiobjective optimization problem and describes the method, the genetic representation adopted and the variation operators used. In section IV, finally, the results of the method on synthetic and real life networks are presented, and a comparison with the approaches of [15] and [12] is reported.

II. Problem Formulation

A. Notation

Let $\{1,\dots,T\}$ be a finite set of time steps and $V = \{1,\dots,n\}$ be a set of individuals or objects. A static network $\mathcal{N}^t$ at time t can be modeled as a graph $G^t = (V^t, E^t)$ where V^t is a set of objects, called nodes or vertices, and E^t is a set of links, called edges, that connect two elements of V^t at time t. Thus G^t is the graph representing a snapshot of the network $\mathcal{N}^t$ at time t. $V^t \subseteq V$ is a subset of individuals V observed at time t. An edge $(u^t, v^t) \in E^t$ if individuals u and v have interacted at time t.

A community (also called cluster or module) in a static network $\mathcal{N}^t$ is a group of vertices $V_i^t \subseteq V^t$ having a high density of edges inside the group, and a lower density of edges with the remaining nodes V^t/V_i^t. Let C^t denote the sub-graph representing a community.

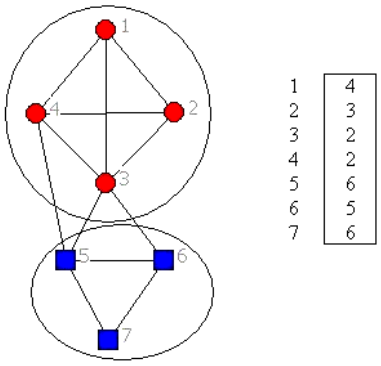


Fig. 1. A network of 7 nodes partitioned in two communities $\{1,2,3,4\}$ and $\{5,6,7\}$, and the corresponding locus-based representation.

A clustering, or community structure, $\mathcal{CR}^t = \{C_1^t, \dots C_k^t\}$ of a network $\mathcal{N}^t$ at time t is a partitioning of G^t in groups of nodes such that for each couple of communities C_i^t and $C_j^t \in \mathcal{CR}^t$, $V_i^t \cap V_j^t = \emptyset$.

A dynamic network is a sequence $\mathcal{N} = \{\mathcal{N}^1, \dots, \mathcal{N}^T\}$ of static networks, where each $\mathcal{N}^t$ is a snapshot of individuals and connections among these individuals at time t.

B. Evolutionary Clustering

Evolutionary clustering was introduced by Chakrabarti et al. in [2] as the problem of clustering data coming at different time steps to produce a sequence of clusterings. At each time step a new clustering must be produced by simultaneously optimizing two conflicting criteria. The first is that the clustering should reflect as accurately as possible the data coming during the current time step. The second is that each clustering should not shift dramatically from one time step to the successive. To satisfy this last property a framework called *temporal smoothness* is defined. This framework assumes that the abrupt change of clustering in a short time period is not desirable, thus it $smooths$ each community over time. For smoothing, a cost function composed by two sub-costs, the *snapshot cost* ($\mathcal{SC}$) and the *temporal cost* ($\mathcal{TC}$), is defined. The snapshot cost $\mathcal{SC}$ measures how well a community structure $\mathcal{CR}^t$ represents the data at time t. The temporal cost $\mathcal{TC}$ measures how similar the community structure $\mathcal{CR}^t$ is with the previous clustering $\mathcal{CR}^{t-1}$. As pointed out by the authors, the clustering algorithm must trade off the benefit of maintaining a consistent clustering over time (temporal cost) with the cost of deviating from an accurate representation of the current data (snapshot cost) [2]. The framework of evolutionary clustering is apt in those situations in which the clustering result is frequently and regularly consumed by a user. Thus mild changes are preferred over dramatic shifts because in such a way the user is not required to learn a new data segmentation. Evolutionary clustering will provide a smooth view of the transition for successive time steps. In this setting it is possible

to associate clusters within the historical context and thus trace their evolution. Chakrabarti et al. defined the cost function for generic data objects. A specialized version of this function in the context of dynamic networks has been introduced in [15], and adopted also by Kim and Han in [12]. The cost function in this case is defined as follows:

$$cost = \alpha \cdot \mathcal{SC} + (1 - \alpha) \cdot \mathcal{TC}$$

where α is an input parameter used by the user to emphasize one of the two objectives. When α = 1 the approach returns the clustering without temporal smoothing. When α = 0, however, the same clustering of the previous time step is produced, i.e. $\mathcal{CR}^t = \mathcal{CR}^{t-1}$. Thus a value between 0 and 1 is used to control the preference degree of each sub-cost.

In the next section we propose a multiobjective evolutionary community detection approach that tries to optimize both the snapshot cost and the temporal cost without the need to fix the control parameter α. The solutions contained on the Pareto front of the multiobjective optimization problem will represent the better compromise satisfying both the snapshot and temporal costs.

It is worth to note that the word $evolutionary$ has a different meaning with respect to the context in which it is used. For Chakrabarti et al. in [2] the term evolutionary is intended as temporal evolution. In the context of multiobjective optimization it means evolutionary algorithms implementing the concept of Darwinian biological evolution [11].

III. Multiobjective Evolutionary Clustering

A multiobjective evolutionary clustering problem $(\Omega, \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h)$ for a static network $\mathcal{N}^t$ can be defined as

$$\textit{min } \mathcal{F}_i(\mathcal{CR}^t), \quad i = 1, \ldots, h \qquad \textit{subject to} \quad \mathcal{CR}^t \in \Omega$$

where $\Omega = \{\mathcal{CR}_1^t, \ldots, \mathcal{CR}_k^t\}$ is the set of feasible clusterings of $\mathcal{N}^t$ at time stamp t, and $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h\}$ is a set of h single criterion functions. Each $\mathcal{F}_i : \Omega \rightarrow \mathcal{R}$ is a different objective function that determines the feasibility of the clustering obtained. Since $\mathcal{F}$ is a vector of competing objectives that must be simultaneously optimized, there is not one unique solution to the problem, but a set of solutions are found through the use of Pareto optimality theory [9]. Given two solutions $\mathcal{CR}_1$ and $\mathcal{CR}_2 \in \Omega$, solution $\mathcal{CR}_1$ is said to *dominate* solution $\mathcal{CR}_2$, denoted as $\mathcal{CR}_1 \prec \mathcal{CR}_2$, if and only if

$$\forall i : \mathcal{F}_i(\mathcal{CR}_1) \leq \mathcal{F}_i(\mathcal{CR}_2) \ \wedge \ \exists \ i \ s.t. \ \mathcal{F}_i(\mathcal{CR}_1) < \mathcal{F}_i(\mathcal{CR}_2)$$

Instead, a *nondominated* solution is one for which an improvement in one objective requires a degradation of another. These solutions are called *Pareto-optimal*. The goal is therefore to construct the Pareto optima. More formally, the set of Pareto-optimal solutions Π is defined as

$$\Pi = \{\mathcal{CR} \in \Omega : \quad \nexists \mathcal{CR}' \in \Omega \ with \ \mathcal{CR}' \prec \mathcal{CR}\}$$

The vector $\mathcal{F}$ maps the solution space into the objective function space. When the nondominated solutions are plotted in the objective space, they are called the *Pareto front*. Thus the *Pareto front* represents the better compromise solutions satisfying all the objectives as best as possible.

In the last few years many efforts have been devoted to the application of evolutionary computation to the development of multiobjective optimization algorithms. Evolutionary algorithms, in fact, proved very successful to solve multiobjective optimization problems because of the population-based nature of the approach that allows the generation of several elements of the Pareto set in a single run [4], [7].

The MultiObjective Genetic Algorithm (MOGA) we used is the *Nondominated Sorting Genetic Algorithm (NSGA-II)* proposed by Srinivas and Deb in [22] and implemented in the *Genetic Algorithm and Direct Search Toolbox* of MATLAB. NSGA-II builds a population of competing individuals and ranks them on the basis of nondominance (for a detailed description of the approach see [7]). In order to employ NSGA-II, *DYN-MOGA* has been adapted with a customized population type that suitably represents a partitioning of a network and endowed with two complementary objectives. In the following the objective functions selected, the genetic encoding adopted and the modified variation operators used to work with this encoding are described.

Objective Functions: As described in the previous section, we are interested in optimizing the cost function $cost = \alpha \cdot \mathcal{SC} + (1 - \alpha) \cdot \mathcal{TC}$ composed by the two competing objectives, the snapshot cost $\mathcal{SC}$ and the temporal cost $\mathcal{TC}$. Since $\mathcal{SC}$ measures how well a community structure C^t represents the data at time t, we need an objective function that maximizes the number of connections inside each community and minimizes the number of links between the communities. To this end we employ the *community score* introduced in [19] that proved very effective in detecting communities.

Let $\mathcal{CR}^t = \{C_1^t, \ldots C_k^t\}$ be a clustering of a network G^t at time t. The *community score* of $\mathcal{CR}^t$ is defined as follows

$$\mathcal{CS}(\mathcal{CR}^t) = \sum_{i=1}^{k} score(C_i^t) \tag{1}$$

where

$$score(C_i^t) = \frac{\sum_{i \in C^t} (\mu_i)^2}{|C^t|} \times \sum_{i,j \in C^t} A_{ij}^t \tag{2}$$

The second term of (2) is the number of edges connecting vertices inside C^t, i.e the number of 1 entries in the adjacency sub-matrix A^t corresponding to C^t. The first term computes the square mean of

$$\mu_i = \frac{1}{\mid C^t \mid} \sum_{j \in C^t} A_{ij}^t$$

where μ_i denotes the fraction of edges connecting each node i of C^t to the nodes in the same community C^t. Thus the score takes into account both the fraction of interconnections

among the nodes (through the first term of (2)), and the number of interconnections contained in the module C^t (through the second term of (2)).

The second objective must minimize the temporal cost $\mathcal{TC}$, thus we need a metric to measure how similar the community structure $\mathcal{CR}^t$ is with the previous clustering $\mathcal{CR}^{t-1}$. To this end we employ the *Normalized Mutual Information*, a well known entropy measure in information theory. Given two partitionings $A = \{A_1, \ldots, A_a\}$ and $B = \{B_1, \ldots, B_b\}$ of a network in communities, let C be the confusion matrix whose element C_{ij} is the number of nodes of the community $A_i \in A$ that are also in the community $B_j \in B$. The normalized mutual information $NMI(A, B)$ is defined as:

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

where c_A (c_B) is the number of groups in the partitioning 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, $NMI(A, B) = 0$. Thus our second objective at a generic time step t is to maximize $NMI(\mathcal{CR}^t, \mathcal{CR}^{t-1})$.

Genetic representation: Our clustering algorithm uses the locus-based adjacency representation proposed in [18]. In this graph-based representation an individual of the population consists of N genes $g_1, \ldots, g_N$, where N is the number of nodes. Each gene can assume allele value j in the range $\{1, \ldots, N\}$. Genes and alleles represent nodes of the graph $G = (V, E)$ modeling a network $\mathcal{N}$, and a value j assigned to the i-th gene is interpreted as a link between the nodes i and j of V. This means that in the clustering solution found i and j will be in the same cluster. A decoding step, however, is necessary to identify all the components of the corresponding graph. The nodes participating to the same component are assigned to one cluster. A main advantage of this representation is that the number k of clusters is automatically determined by the number of components contained in an individual and determined by the decoding step. Figure 1 shows a network partition and the corresponding encoded genotype.

Initialization: A random individual is generated such that if in the i-th position there is an allele value j, than j must be one of the neighbors of i, i.e. the edge (i, j) must exist.

Uniform Crossover: We used uniform crossover because it guarantees the maintenance of the effective connections of the nodes in the network in the child individual. In fact, because of the biased initialization, each individual in the population has the property that, if a gene i contains a value j, then the edge (i, j) exists. Thus, given two parents, a random binary mask is created. Uniform crossover (see Table I) then selects the genes where the mask is a 0 from the first parent, and the genes where the mask is a 1 from the second parent, and combines the genes to form the child. The child at each position i contains a value j coming from one of the two parents. Thus the edge (i, j) exists.

Input: Given a dynamic network $\mathcal{N} = \{\mathcal{N}^1, \ldots, \mathcal{N}^T\}$, the sequence of graphs $\mathcal{G} = \{G^1, \ldots, G^T\}$ modeling it, and the number T of timestamps.
Output: A clustering for each network $\mathcal{N}^i$ of $\mathcal{N}$.
Method: Perform the following steps:

1 **Generate** an initial clustering $\mathcal{CR}^1 = \{C_1^1, \ldots C_k^1\}$ of the network $\mathcal{N}^1$ without smoothing by optimizing only the first objective (i.e. the community score);
2 **for** $t = 2$ to T
3 **Create** a population of random individuals whose length equals the number $N =\mid V^t \mid$ of nodes of G^t;
4 **while** termination condition is not satisfied **do**
5 **Decode** each individual $I = \{g_1, \ldots, g_N\}$ of the population to generate the partitioning $\mathcal{CR}^1 = \{C_1^t, \ldots, C_k^t\}$ of the graph G^t in k connected components;
6 **Evaluate** the two fitness values of the translated individuals;
7 **Assign** a rank to each individual and **sort** them according to nondomination rank;
8 **Create** a new population of offspring by applying the variation operators;
9 **Combine** the parents and offspring into a new pool and partition it into fronts;
10 **Select** points on the lower front (with lower rank) and apply the variation operators on them to create the next population;
11 **end while**
12 **return** the solution $\mathcal{CR}^t = \{C_1^t, \ldots C_k^t\}$ of the Pareto front having the maximum modularity value;
13 **end for**

Fig. 2. The pseudo-code of the *DYN-MOGA* algorithm.

Parent1 :	4 3 2 2 6 5 6
Parent2 :	3 3 1 5 4 7 6
Mask :	0 1 1 0 0 1 1
Offspring	4 3 1 2 6 7 6

TABLE I
EXAMPLE OF UNIFORM CROSSOVER.

Mutation: The mutation operator that randomly changes the value j of a i-th gene causes a useless exploration of the search space, because of the same above observations on node connections. Thus the possible values an allele can assume are restricted to the neighbors of gene i. This mutation guarantees the generation of a mutated child in which each node is linked only with one of its neighbors.

The pseudo-code of *DYN-MOGA* is reported in Figure 2. Given a dynamic network $\mathcal{N} = \{\mathcal{N}^1, \ldots, \mathcal{N}^T\}$ and the sequence of graphs $\mathcal{G} = \{G^1, \ldots, G^T\}$ modeling it, *DYN-MOGA* finds a partitioning of the network $\mathcal{N}^1$ by running the genetic algorithm that optimizes only the first objective, i.e. the community score. For a given number of timestamps, the multiobjective genetic algorithm creates a population of random individuals whose length is the number of nodes of the current graph G^t. Then, for a fixed number of generations, it decodes the individuals to generate the partitioning at time step t, evaluates the objective values, assigns a rank to each individual according to Pareto dominance and sorts them. A new population is generated by applying the specialized variation operators described above. Parents and offspring are then combined, and the new pool is partitioned into fronts. The individuals with the lower rank are selected and variation

operators are applied on them to create the new population. At the end of each timestamp *DYN-MOGA* returns a set of solutions, i.e. all those contained in the *Pareto front*. Each of these solutions corresponds to a different trade-off between the two objectives and thus to diverse partitioning of the network consisting of various number of clusters. A criterion should be established to automatically select one solution with respect to another. To this end, we use the *modularity*, introduced by Girvan and Newman [16] to select, among the solutions found, that having the highest value of modularity. The modularity is a well known quality function to evaluate the goodness of a partitioning. Let k be the number of modules found inside a network, the modularity Q is defined as

$$Q = \sum_{s=1}^{k} [\frac{l_s}{m} - (\frac{d_s}{2m})^2]$$

where m is the number of edges of the network, l_s is the total number of edges joining vertices inside the module s, and d_s is the sum of the degrees of the nodes of s. The first term of each summand of the modularity Q is the fraction of edges inside a community, the second one is the expected value of the fraction of edges that would be in the network if edges fall at random without regard to the community structure. Values approaching 1 indicate strong community structure.

In the next section we show that *DYN-MOGA* is able to find meaningful network structure for both synthetic and real life data sets.

IV. Experimental Results

In this section we study the effectiveness of our approach and compare the results obtained by *DYN-MOGA* w.r.t. the algorithms of Lin et al. [15] and Kim and Han [12] on synthetic networks for which the partitioning in communities is known. Then, we also evaluate our method on a real-world network. In both cases we show that our multiobjective genetic algorithm successfully detects the network structure and is very competitive vs. the other approaches.

The *DYN-MOGA* algorithm has been written in MATLAB[1], using both the *Genetic Algorithms* and *Direct Search 2* toolboxes. The experiments have been performed on a Pentium 4 machine with 1800MHz and 1GB RAM. We used standard parameters for the genetic algorithm: *crossover rate* = 0.8, *mutation rate* = 0.2, *elite reproduction* = 10% of the population size, and roulette selection function. The population size was 200, the number of generations 30.

Synthetic data set. In order to check the ability of our approach to successfully detect the community structure of a dynamic network, we used the same benchmark adopted by Lin et al. [15] and Kim and Han [12]. It consists of two kinds of data sets. The first is a dynamic network of a fixed number of communities (named SYN-FIX). The second is a dynamic network of a variable number of communities (named SYN-VAR).

[1]http://www.mathworks.com

SYN-FIX is generated analogously to the classical benchmark proposed by Girvan and Newman in [10]. The network consists of 128 nodes divided into four communities of 32 nodes each. Every node has an average degree of 16 and shares a number z_{in} of links with the nodes of its community, and z_{out} with the other nodes of the network. Increasing z_{out} augments the noise level of the network. In order to introduce dynamics in $\mathcal{G}$, 3 nodes are randomly selected from each community in G^{t-1} and randomly assigned to the other three communities. Edges are placed with higher probability between a pair of nodes of the same community, and with lower probability between nodes of different communities. These probabilities depend on the value of z_{out}.

SYN-VAR is obtained by modifying the generation method of SYN-FIX to introduce the forming and dissolving of communities and the attaching and detaching of nodes. The initial networks contains 256 nodes, divided in 4 communities of 64 nodes each. 10 consecutive networks are generated by choosing 8 nodes from each community and generating a new community with these 32 nodes. This is done for 5 timestamps, then the nodes return to the original communities. Thus, the number of communities for the 10 timestamps is 4, 5, 6, 7, 8, 8, 7, 6, 5, 4. The average degree of each node in a cluster is set to the half of the size of this cluster. Furthermore, at each time step 16 nodes are randomly deleted and 16 new nodes are added to the network.

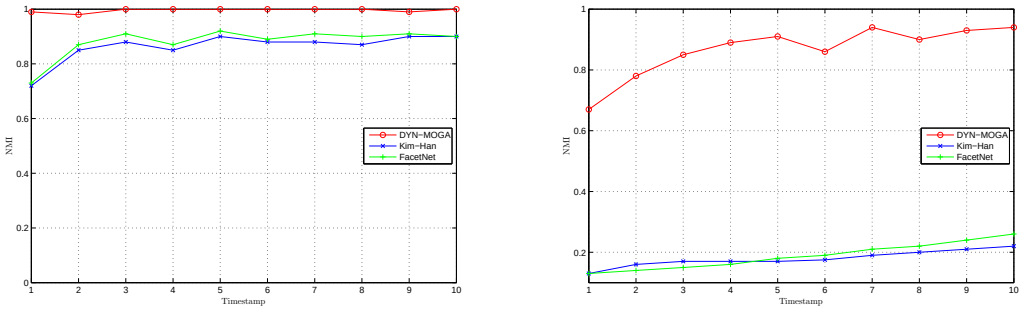
We generated 10 different networks for 10 timestamps and run *DYN-MOGA* on them. Since the network structure is known, we computed the *Normalized Mutual Information* to measure the similarity between the true partitions and the detected ones.

Figure 3 shows the average normalized mutual information, over the 10 networks for the 10 timestamps for SYN-FIX when the value of $z_{out} = 3$ (Figure 3(a)) and $z_{out} = 5$ (Figure 3(b)).

Figure 4 shows the average normalized mutual information, over the 10 networks for the 10 timestamps for SYN-VAR when the value of $z_{out} = 3$ (Figure 4(a)) and $z_{out} = 5$ (Figure 4(b)).

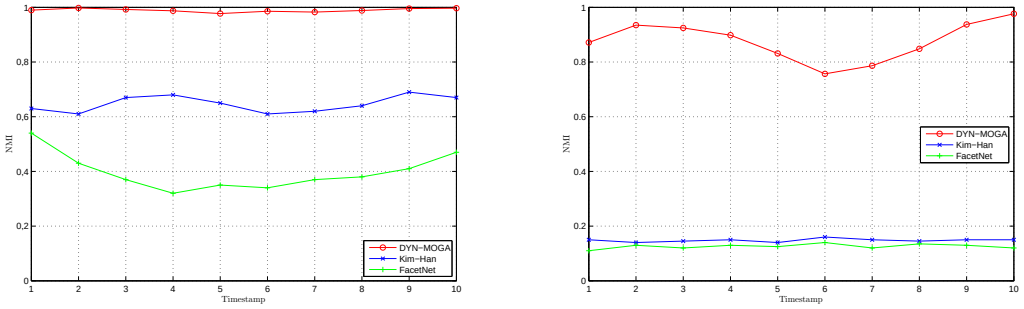
Both figures show the significantly better results obtained by *DYN-MOGA* with respect to both FacetNet and Kim-Han algorithms. In fact, for SYN-FIX and SYN-VAR, when $z_{out} = 3$, *DYN-MOGA* obtains a value which is almost always 1, while FacetNet and Kim-Han are around 0.9 for SYN-FIX and between 0.3 and 0.7 for SYN-VAR. The differences, however, are much more remarkable when $z_{out} = 5$. In this case *DYN-MOGA* obtains values above 0.8 for all the timestamps, except the first one, while both FacetNet and Kim-Han methods fail to uncover the community structure. The normalized mutual information obtained, in fact, is between 0.1 and 0.2. Same considerations apply for SYN-VAR, when $z_{out} = 5$. Also in this case FacetNet and Kim-Han algorithms are not able to find the true community structure, getting values of normalized mutual information between 0.1 and 0.2, while the values obtained by *DYN-MOGA* are above 0.75.

It is worth to note that the results reported for FacetNet and Kim-Han algorithms have been obtained by the authors



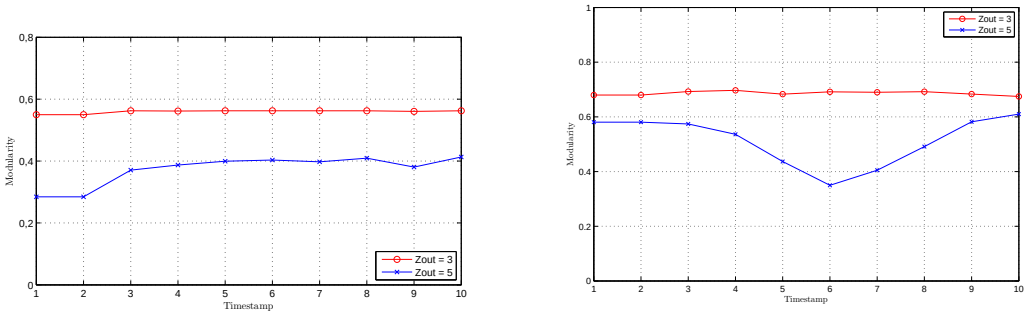
(a) (b)

Fig. 3. Normalized mutual information of clustering results for SYN-FIX when $z_{out} = 3$ (a) and $z_{out} = 5$ (b).



(a) (b)

Fig. 4. Normalized mutual information of clustering results for SYN-VAR when $z_{out} = 3$ (a) and $z_{out} = 5$ (b).



(a) (b)

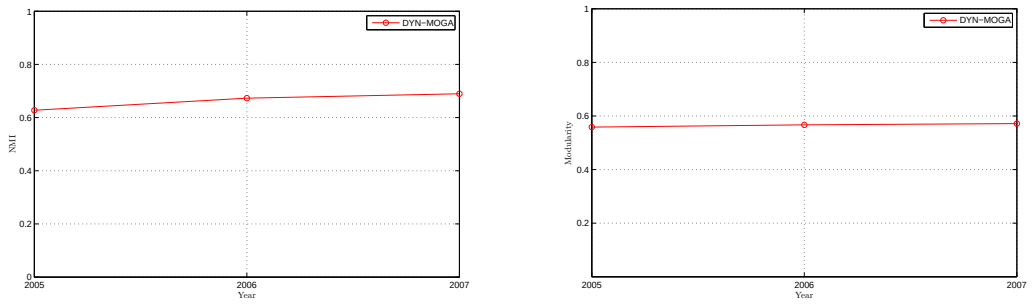
Fig. 5. Modularity values obtained by *DYN-MOGA* for SYN-FIX ($z_{out} = 3, 5$) (a) and for SYN-VAR ($z_{out} = 3, 5$) (b).

for $\alpha = 0.8$, i.e., they gave higher preference to the snapshot quality. However, in spite of the higher preference degree in searching for the true data clustering as better as possible, they could not detect the community structure.

Finally, Figure 5(a) and Figure 5(b) report the modularity values obtained by *DYN-MOGA* for the two synthetic networks. The values corroborate the good performance of *DYN-MOGA* in discovering dense interconnections in networks.

Real-life data set. We now show the application of *DYN-MOGA* on the *Football data*[2]. The Football network comes from the United States college football. The football data is the NCAA Football Division 1-A games. Nodes in the graph represent teams and edges represent the regular season games between the two teams they connect. The teams are divided in conferences and they tend to play between members of the same conference, thus the team cluster is assumed to be the conference. This data set, restricted to year 2000, has been used by Girvan and Newman in [10]. We selected years 2005, 2006, and 2007. The number of conferences is 12 for all the three years and the number of teams is 120. Figure 6(a) shows the NMI over the three years. The values obtained are between 0.6 and 0.7, which is a quite good result. This is also confirmed by the modularity values reported in figure 6(b),

[2]http://www.jhowell.net/cf/scores/scoresindex.htm



(a) (b)

Fig. 6. Results for *Football* network: Normalized Mutual Information (a) and Modularity (b).

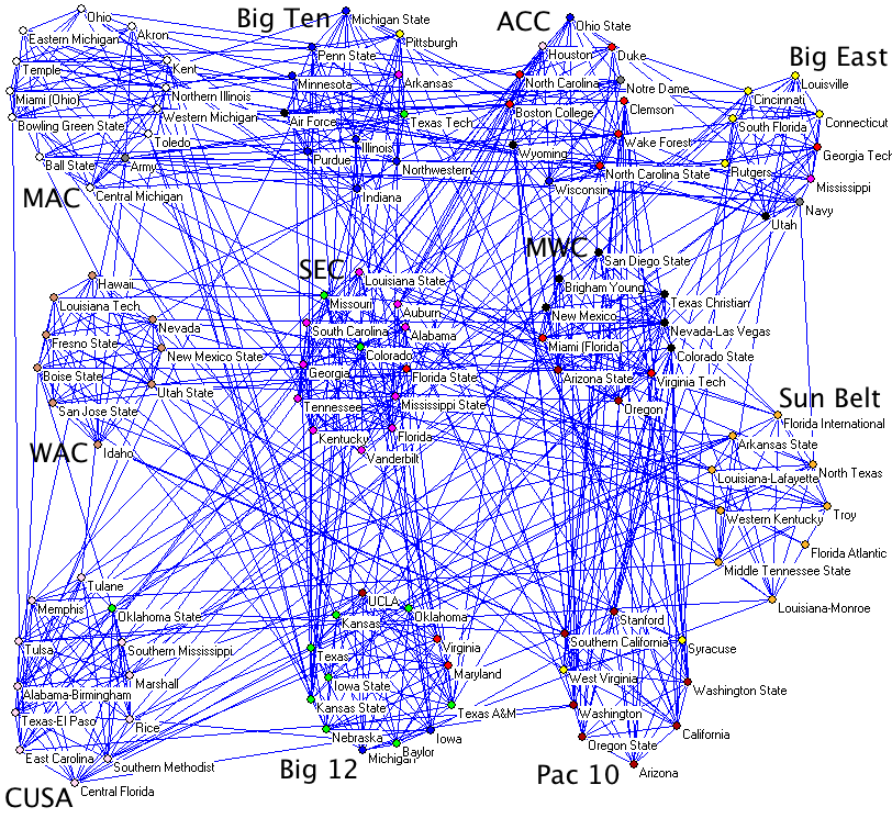


Fig. 7. Communities found by *DYN-MOGA* on the Football data for the year 2007.

that are almost 0.6.

To conclude, Figure 7 displays the communities recognized by *DYN-MOGA* on the Football data for the year 2007. The figure has been obtained by using the $Pajek$ software [6]. In particular, we associated 12 distinct RGB colors (see Table II for more details) with the 12 true classes – i.e. the conferences the teams really belong to – and used them to paint the nodes. Then, we grouped the nodes on the base of the clustering provided by the *DYN-MOGA* algorithm.

It is worth notice that many communities exhibit a quite homogeneous coloring so proving the capability of *DYN-MOGA* to effectively deal with the community identification in networks and confirming the quantitative results shown in Figure 6 . For instance, the conferences CUSA, MAC, Pac 10, SEC, Sun Belt, WAC are almost completely identified.

However, some misplaced assignments of nodes to erroneous groups are easily identifiable. These errors are especially due to the inherent complexity of the network at hand. As matter of fact, *DYN-MOGA* was able to recognize 11 (over 12) different communities and, then, all nodes belonging to the missed cluster were redistributed. Clearly, the redistribution causes a reduction in terms of accuracy for the remaining

Conference	Color
ACC	Red
Big 12	Green
Big East	Yellow
Big Ten	Blue
CUSA	Pink
MAC	White
MWC	Black
Pac 10	Maroon
SEC	Magenta
Sun Belt	Dandelion
WAC	Tan
Independent	Gray

TABLE II
ASSOCIATION CONFERENCES-COLORS.

conferences. More in detail, the percentage (on average) of correclty clustered teams is about 62% for the conferences Big 12, Big Ten and MWC, whereas it is about 55% for the conferences ACC and Big East.

V. CONCLUSIONS

A multiobjective genetic algorithm for detecting communities in dynamic networks has been presented. The algorithm at each time step provides the solution representing the best trade-off between the accuracy of the clustering obtained with respect to the data of the current time step, and the drift from one time step to the successive. Experimental results on two kinds of synthetic data sets and a real life network showed the better performance of our approach compared to state-of-the-art methods. Future work aims at evaluating the method on large-scale networks to analyze the scalability and applicability of the approach in real-life domains. It is known, in fact, that evolutionary techniques can be very computing demanding and require high memory capability to store populations of individuals. On the other hand they are naturally parallelizable. Thus the implementation of *DYNMOGA* on a parallel architecture would provide a considerable improvement in terms of both performance and scalability.

VI. ACKOWLEDGMENTS

We wish to thank Min-Soo Kim for providing us the synthetic data set generator, the Football data set, and the results obtained by FacetNet and his method on the synthetic networks.

REFERENCES

[1] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data*, 3(4):Paper 16, 2009.

[2] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proc. of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 554–560, 2006.

[3] Y. Chi, X. Song, D.Zhou, K.Hino, and B.L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 153–162, 2007.

[4] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.

[5] D. Datta, J.R Figuera, C.M. Fonseca, and F. Tavares-Pereira. Graph partitioning through a multi-objective evolutionary algorithm: A preliminary study. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'08)*, pages 625–632, 2008.

[6] W. de Nooy, A. Mrvar, and V.Batagelj. *Exploratory social network analysis with pajek*. Cambridge University Press, New York, 2005.

[7] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Ltd, Chichester, England, 2001.

[8] G. N. Demir, A. S. Uyar, and S. Oguducu. Graph-based sequence clustering through multiobjective evolutionary algorithm for web recommender systems. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'07)*, pages 1943–1950, 2007.

[9] M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin, 2nd edition, 2005.

[10] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proc. National. Academy of Science. USA 99*, pages 7821–7826, 2002.

[11] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[12] M. Kim and J. Han. A particle-and-density based evolutionary clustering method for dynamic networks. In *Proc. of the International Conference on Very Large Data Bases (VLDB'09)*, pages –, 2009.

[13] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 611–717, 2006.

[14] L. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws,shrinking diameters and possible explanations. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD'05)*, pages 177–187, 2005.

[15] Yu-Ru Lin, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Facetnet: A framework for analyzing communities and their evolutions in dynamic networks. In *Proc. of the International World Wide Web Conference (WWW'08)*, pages 685–694, 2008.

[16] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[17] G. Palla, A.L. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, (466), 2007.

[18] Y.J. Park and M.S. Song. A genetic algorithm for clustering problems. In *Proc. of 3rd Annual Conference on Genetic Algorithms*, pages 2–9, 1989.

[19] Clara Pizzuti. Ga-net: a genetic algorithm for community detection in social networks. In *Proc. of the 10th Intenational Conference on Parallel Problem Solving from Nature (PPSN 2008)*, pages 1081–1090, 2008.

[20] Clara Pizzuti. A multi-objective genetic algorithm for community detection in networks. In *Proc. of the 121st IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2009)*, pages 379–386, 2009.

[21] Myra Spiliopoulou, Irene Ntoutsi, Yannis Theodoridis, and Rene Schult. Monic - modeling and monitoring cluster transitions. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 706–711, 2006.

[22] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[23] J. Sun, C. Faloutsos, S. Papadimitriou, and P.S. Yu. Graphscope: parameter-free of large time evolving-graphs. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD'05)*, pages 687–696, 2005.

[24] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 677–685, 2007.

[25] T. Xu, Z. Zhang, P. S. Yu, and Bo Long. Dirichlet process based evolutionary clustering. In *Proc. of the 8th IEEE International Conference on Data Mining (ICDM'08)*, pages 648–657, 2008.

[26] T. Xu, Z. Zhang, P. S. Yu, and Bo Long. Evolutionary clustering by hierarchical dirichlet process with hidden markov state. In *Proc. of the 8th IEEE International Conference on Data Mining (ICDM'08)*, pages 658–667, 2008.