An Evolutionary Multiobjective Approach for Community Discovery in Dynamic Networks

Francesco Folino, Clara Pizzuti

Abstract—The discovery of evolving communities in dynamic networks is an important research topic that poses challenging tasks. Evolutionary clustering is a recent framework for clustering dynamic networks that introduces the concept of temporal smoothness inside the community structure detection method. Evolutionary-based clustering approaches 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 when compared with state-of-the-art approaches.

Index Terms—Evolutionary clustering, complex networks, dynamic networks, community discovery.

1 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 individual objects, and edges represent interactions among these objects. Individuals in a network interact with 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], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. Some of these methods [2], [5], [10], [3] employ the concept of *evolutionary clustering*, introduced by Chakrabarti et al. in [13], 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 dramatically shift from one time step to the next one.

In this paper a multiobjective approach, named *DYNMOGA* (*DYNamic MultiObjective Genetic Algorithms*), to discover communities in dynamic networks by employing genetic algorithms [14], is proposed. 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 time steps. In order to maximize the snapshot quality to measure the goodness of the division in communities of a network, different quality measures, used in the literature to capture the intuition of network community [15], are considered. In particular, we fix the attention on the concept of *modularity*, introduced by Newman and Girvan in [16].

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 [17].

DYNMOGA 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

- *F. Folino is with National Research Council of Italy (CNR), Institute for High Performance Computing and networking (ICAR), Via Pietro Bucci, 41C, 87036 Rende (CS), Italy, Email: {ffolino}@icar.cnr.it*
- *Clara Pizzuti is with National Research Council of Italy (CNR), Institute for High Performance Computing and networking (ICAR), Via Pietro Bucci, 41C, 87036 Rende (CS), Italy, Email: {pizzuti}@icar.cnr.it*

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 [18], [19], [20], and for data clustering [21], their use for dynamic networks, however, has not been explored very much [22].

The main contributions of the paper can be summarized as follows:

- The detection of community structure in dynamic networks is formalized as a multiobjective optimization problem. The first objective searches for highly modular structures at the current time step, the second objective tries to minimize the differences between the community structure at the current time step and that obtained at the previous time step.
- The approach proposed can be considered as a general framework for evolutionary clustering. In fact, it is sufficient to change one of the two objective functions (or both) to implement and test different quality functions for analyzing dynamic networks.
- The method is parameter free. It does not need neither the parameter introduced by Chi et al. [2] to give different weight to the snapshot and temporal costs, nor the number of clusters to find. The former parameter is automatically determined by the multiobjective method, while the number of communities to find at each time step is determined by the genetic representation employed.

The paper is organized as follows. In Section 2 the concept of dynamic network is defined and the evolutionary clustering problem is formalized. Section 3 gives a review of the evolutionary clustering approaches. Section 4 formulates the community detection problem in dynamic networks as a multiobjective optimization problem. Section 5 describes the method, the genetic representation adopted and the variation operators used. In section 6 the results of the method on synthetic and real life networks are presented, and a comparison with the approaches of [5] and [3] is reported. Section 7, finally, concludes the paper.

2 Problem Formulation

Let $\{1, \ldots, T\}$ be a finite set of time steps and $V = \{1, \ldots, n\}$ 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 low density of edges with the remaining nodes V^t / V_i^t. Let C^t denote the sub-graph representing a community.

A clustering, or community structure, $\mathcal{CR}^t = \{C_1^t, \ldots C_k^t\}$ 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 $\mathcal{N} = \{\mathcal{N}^1, \ldots, \mathcal{N}^T\}$ of static networks, where each $\mathcal{N}^t$ is a snapshot of individuals and connections among these individuals at time t.

3 Related work

Analyzing networks and their evolution is recently receiving increasing interest from researchers [7], [23], [24]. In fact, the representation of many complex systems through a static graph, even when the temporal dimension describing the varying interconnections among nodes is available, does not allow to study the network dynamics and the changes it incurs over time. Kumar et al. [4] studied the evolution of network properties of two large blogosphere networks, and tried to classify members of each network into groups and their changes. Sun et al. [9] introduced $GraphScope$, an efficient parameter-free method, based on the Minimum Description Length principle, to discover communities in evolving graphs. Asur et al. [1] have characterized the evolution of communities by defining critical events that occur in dynamic graphs. Tang et al. [25] studied multimode networks and introduced a spectral clustering framework to discover communities and find out how they evolve. There are several other methods for detecting communities in dynamic networks, not mentioned here. For a recent survey see [26]. Many of the proposed approaches divide the community detection step from the temporal analysis of the network, i.e. first the communities are extracted, and then the structural differences over time are analyzed in order to determine the correspondences among communities in two consecutive time steps.

A different methodology, known as *evolutionary clustering*, has been proposed by Chakrabarti et al. in [13]. Since our method has been inspired by the evolutionary clustering framework, in the following lines a more detailed description of evolutionary clustering approaches is reported.

Chakrabarti et al. in [13] observe that, often, changes of connections in short time periods could be caused by noise. Thus, though clustering mainly depends on the current object connections, in many dynamic application domains, abrupt drifts from the most recent history should not be expected. 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 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 to 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) [13]. 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 to trace their evolution. Though evolutionary clustering is robust to noise, it does not allow that the number of communities varies over time, neither that new communities may appear, nor that existing communities may dissolve.

Chakrabarti et al. proposed the framework for generic data clustering, and applied it to two well known clustering methods, k-means and agglomerative hierarchical clustering, to deal with evolving data. Thus they introduced the cost function for generic data objects. A specialized version of this function in the context of dynamic networks has been first introduced by Chi et al. [2] and adopted also by Lin et al. [5], and Kim and Han [3]. Chi et al. [2] define the cost function as follows:

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

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. The authors proposed two evolutionary spectral clustering algorithms by using the *normalized cut* concept of Shi and Malik [27]. The two methods differ in how the temporal cost $\mathcal{TC}$ is measured.

An evolutionary-based framework for analyzing communities and evolutions in dynamic networks, named $FacetNet$, has been proposed by Lin et al. [5]. The framework employs a stochastic block model for generating communities, and a probabilistic model based on the Dirichlet distribution to catch community evolutions. The authors 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. $FacetNet$ thus, at each iteration, updates the values of the approximate structure in order to decrease the cost function. Convergence to a local 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.

Kim and Han [3] proposed a *particle-and-density* based clustering method based on the evolutionary approach of Chakrabarti et al. [13], extended to deal with a variable number of communities between different time steps. 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. Furthermore, to map communities between consecutive time periods, the mutual information concept among the clusterings obtained at time $t-1$ and t is employed, by purifying the link distributions between consecutive networks in order to maximize the mutual information. The mapping can determine the stage of each community as evolving, forming, splitting. It is worth to note that the approach does not take into account neither the possibility of a community to split into multiple communities, nor that multiple communities merge into only one.

The described methods present two main limitations. The first regards the number of clusters to find, the second is relative to the value α to choose in order to apply temporal smoothness. The approach of Chi et al. allows for a varying number k of communities between successive time steps, however the value of k must be given as input parameter. Some heuristics are argued by the authors in order to determine its optimal value. $FacetNet$ assumes a fixed number k of communities over time, though the authors suggest that, to detect the best community number at time t, the algorithms could be executed for a range of different k values, and that giving the best value of modularity could be chosen. Kim and Han's approach does not need the number of communities, however, since their approach is density-based, they must set the density parameters. Another problem is the value to use for the temporal smoothness. All the approaches must give it as input parameter in order to control 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 the following we propose a parameter-free method that automatically determines both the number k of communities, and the optimal α value.

4 Multiobjective Clustering

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

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

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* [28]. 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 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 [29], [14].

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 best 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 [13] 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 [30].

5 THE DYNMOGA ALGORITHM

The MultiObjective Genetic Algorithm (MOGA) we used is the *Nondominated Sorting Genetic Algorithm (NSGA-II)* proposed by Srinivas and Deb in [31] 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 [14]). In order to employ NSGA-II, *DYNMOGA* 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, we first give a brief introduction to Genetic Algorithms, and than the objective functions selected, the genetic encoding adopted, and the modified variation operators used to work with this encoding are described.

Genetic Algorithms (GAs) are a class of adaptive general-purpose search techniques inspired by natural evolution proposed by Holland [32] in the early 1970s as computer programs that simulate the evolution process in nature. A standard Genetic Algorithm evolves a constant-size population of individuals (called $chromosomes$) by using the genetic operators of *reproduction*, *crossover* and *mutation*. Each chromosome is composed by a number of genes and represents a candidate solution to a given problem. An individual is associated with a *fitness value* that reflects how good it is, with respect to the other solutions in the population. Reproduction operator copies elements of the current population into the next generation with a selected strategy. Crossover generates two new chromosomes by crossing two elements of the population. Mutation randomly alters genes of individuals.

Objective Functions: As described in the previous section, we are interested in optimizing the cost function (formula (1)) 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 while minimizing the number of links between the communities. To this end we employ four quality measures, well known in the community detection field [15], that formalize the intuitive concept of community.

Let $\mathcal{CR}^t = \{C_1^t, \dots C_k^t\}$ be a clustering of a network $G^t = (V^t, E^t)$ at time t with n nodes and m edges, C^t a cluster having n_s nodes and m_s edges, $m_s(u) = \{v \mid v \in C^t\}$ be the number of nodes in C^t connected with u, $c_s = \{(u, v) \mid u \in C^t, v \notin C^t\}$ be the number of edges on the boundary of C^t, l_s the total number of edges joining vertices inside the module C_s^t, and d_s the sum of the degrees of the nodes of C_s^t.

The scores we consider are $modularity$ [16], $conductance$ [33], *normalized cut* [27], and *community score* [22]. Their definition is as follows:

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

In modularity Q [16] the first term of each summand of is the fraction of edges inside a community, while 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.

$$CO = \sum_{s=1}^{k} \frac{c_s}{2m_s + c_s} \tag{3}$$

Conductance CO [33] measures the fraction of edges pointing outside the clustering.

$$NC = \sum_{s=1}^{k} \frac{c_s}{2m_s + c_s} + \frac{c_s}{2(m - m_s) + c_s} \tag{4}$$

Normalized Cut NC [27] measures the fraction of total edge connections to all the nodes in the graph.

$$CS = \sum_{s=1}^{k} (\sum_{u \in C^t} (\frac{m_s(v)}{n_s})^2) \times \frac{2m_s}{n_s} \tag{5}$$

Community Score CS [22] measures the fraction of internal edges of each cluster per nodes.

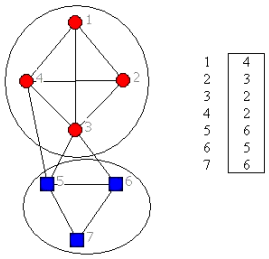


Fig. 1. A network of 7 nodes partitioned in two communities $\{1,2,3,4\}$ and $\{5,6,7\}$, and the corresponding locus-based representation.

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 to the previous clustering $\mathcal{CR}^{t-1}$. To this end we employ the *Normalized Mutual Information*, a well known entropy measure in information theory [17]. 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:

$$NMI(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} C_{ij} log(C_{ij}N/C_{i.}C_{.j})}{\sum_{i=1}^{c_A} C_{i.} log(C_{i.}/N) + \sum_{j=1}^{c_B} C_{.j} log(C_{.j}/N)} \quad (6)$$

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 $NMI(\mathcal{CR}^t, \mathcal{CR}^{t-1})$.

```
DECODE(I)
 1) for each vertex i ∈ V
 2)    MAKESET(i)
 3) for each edge (i, g_i)
 4)    r1=FINDSET(i); r2=FINDSET(g_i);
 5) if r1 ≠ r2
 6)    UNION(r1, r2)
MAKESET(i)
 1) p[i]=i;
 2) rank[i]=0;
UNION(r1, r2)
 1) if rank(r1) > rank(r2)
 2)    p(r2)=r1;
 3) else
 4)    p(r1)=r2;
 5) end if
 6) if (rank(r1)==rank(r2))
 7)    rank(r2)=rank(r1)+1;
 8) end
FINDSET(i)
 1) while (i ≠ p(i))
 2)    i=p(i);
 3) end while
 4) return p(i);
```

Fig. 2. The pseudo-code of the $Decode$ procedure.

Genetic representation: Our clustering algorithm uses the locus-based adjacency representation proposed in [34], adopted in [21] for data clustering and in [20] for community detection in static networks. 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 a value j in the range $\{1, \ldots, n\}$. Genes 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 in 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. Figure 5 reports the decoding step pseudo-code to compute the connected components, i.e. the clustering represented by an individual. The procedure DECODE receives in input an individual $I = \{g_1, \ldots, g_n\}$ of n genes. Each couple (i, g_i) is an edge belonging to one of the components of the graph G. Components can be efficiently obtained by using a *disjoint-set* data structure [35] that maintains a collection of disjoint dynamic sets, where each set is identified by one of its members, called *representative*. The detection of connected components involves three main operations: MAKESET(i) creates a new set whose only member is i, UNION(i,j) merges the dynamic sets containing i and j, FINDSET(i) returns a pointer to the representative of the unique set containing i. In order to improve the implementation, each disjoint set is represented by a rooted tree, where the root contains the representative and each node i of the tree points only to its parent $p(i)$. The collection of rooted trees is represented by a *disjoint-set forest*. The UNION operation is critical for obtaining efficiency, thus the heuristics *union by rank* can be adopted. The idea underlying this heuristic is to maintain an upper bound, named *rank*, on the height of each node so that the the root of the tree having less nodes points to the root of the tree having more nodes. To implement a disjoint forest with union-by-rank heuristic, each node i thus maintains an integer value $rank(i)$.

At the beginning, DECODE associates each vertex with its own set (steps 1-2 in Figure 5) by executing the MAKESET operation. MAKESET(i) (steps 1-2 of the MAKESET procedure) sets the value of the parent of i with i itself, and assigns a zero value to $rank(i)$. Then, for each edge (i, g_i) of a individual I, it merges the sets r_1=FINDSET(i) and r_2=FINDSET(g_i) containing i and g_i (steps 3-6), if they do not coincide. FINDSET(i) visits the tree back until it finds the root of the tree i belongs to (steps 1-4 of the FINDSET procedure). The union of two trees must consider two different cases. If the rank of the two roots is different, the root having higher rank becomes the parent of the other one (steps 1-4 of the UNION procedure). If the ranks are the same, one of the two is chosen as parent and its rank is augmented by one (steps 6-7).

Initialization: A population of random individuals is generated such that for each node i, the value of g_i is randomly

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 time steps.
Output: A clustering for each network $\mathcal{N}^i$ of $\mathcal{N}$.
Method: Perform the following steps:

```
1   Generate an initial clustering CR^1 = {C_1^1, ... C_k^1} of the
    network N^1 without smoothing by optimizing only the first
    objective;
2   for t = 2 to T
3     Create a population of random individuals whose
      length equals the number n =| V^t | of nodes of G^t;
4     while termination condition is not satisfied do
5       Decode each individual I = {g_1, ..., g_n} of the
        population to generate the partitioning
        CR^t = {C_1^t, ..., 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 CR^t = {C_1^t, ... C_k^t} of the
    Pareto front having the maximum modularity value;
13  end for
```

Fig. 3. The pseudo-code of the *DYNMOGA* algorithm.

chosen among one of its neighboring nodes j. This means that the edge (i, j) exists.

Uniform Crossover: Given two parents, a random binary mask is created. Uniform crossover (see Table 1) 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 uniform crossover maintains node connections in the child individual since the edge (i, j) exists.

TABLE 1
Example of uniform crossover.

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

Mutation: The mutation operator, similarly to initialization, for each node i randomly changes the value of g_i with one of the neighbors of 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 *DYNMOGA* is reported in Figure 3. 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, *DYNMOGA* finds a partitioning of the network $\mathcal{N}^1$ by running the genetic algorithm that optimizes only the first objective. For a given number of time steps, the multiobjective genetic algorithm creates a population of random individuals whose length is

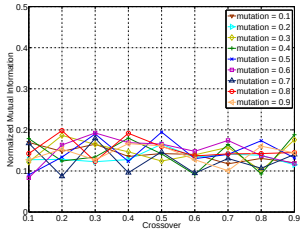


Fig. 4. Normalized mutual information for different combinations of crossover and mutation rates for sysnthetic dataset #1.

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 time step *DYNMOGA* 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 choose the partitioning having the highest value of modularity. This choice is motivated by the fact that, since the Pareto front already selected the nondominated solutions that satisfy the snapshot and the temporal cost at the best, a solution presenting a better community structure is preferable.

Computational Complexity: *DYNMOGA* uses the *Nondominated Sorting Genetic Algorithm (NSGA-II)* proposed by Deb et al. in [36]. NSGA-II builds a population of competing individuals and ranks them on the basis of nondominance. In [37] Jensen proposed an efficient algorithm for nondominated sorting that can be applied for NSGA-II. Jensen showed that the run-time complexity of the NSGA-II algorithm is $O(gp\ log^{h-1}\ p)$, where g is the number of generations, p is the population size, and h is the number of objective functions. Since *DYNMOGA* optimizes two objectives, its complexity will be $O(gp\ log\ p)$. At each generation, however, genetic operators must be executed. In particular, crossover needs $O(n)$ time, mutation $O(1)$ time, while fitness computation is composed of three terms: decoding of an individual, modularity and NMI computation. Decoding can be efficiently performed by using the disjoint set algorithm described in Figure , that represents sets by rooted trees, where each set corresponds to a community. With this data structure the decoding step requires $O(n\ log\ n)$ time [35].

To compute modularity we need to consider, for each node i its d_i neighbors, thus the time complexity is $O(m)$, where m is the number of edges. As regards normalized mutual information, it has been shown [38] that it can be efficiently

computed in $O(n)$ time. Fitness computation can thus be realized in $O(n\ log\ n)+O(m)+O(n)$ time. Therefore, the overall complexity of *DYNMOGA* is $O((gp\ log\ p) \times (n\ log\ n+m))$.

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

6 Experimental Results

In this section we study the effectiveness of our approach by first employing modularity as objective function that optimizes the snapshot cost, and compare the results obtained by *DYNMOGA* w.r.t. the algorithm of Lin et al. [5] and Kim and Han [3] on synthetic networks for which the partitioning in communities is known. In such a case we show that our multiobjective genetic algorithm successfully detects the network structure and it is very competitive with respect to the other approaches. In Section 6.7 we then show the behavior of the method when conductance, normalized cut and community score are applied, and report the results obtained by each objective. Finally, we apply our method on two real-world networks.

The *DYNMOGA* algorithm has been written in MATLAB, using both the *Genetic Algorithms* and *Direct Search 2* toolboxes. Parameter setting is a challenging research problem in evolutionary algorithms. In [39] the problem has been deeply investigated and the authors proved that, though it is possible to find good parameter values for a set of problems, general tuning that allows for good performance on a wide range of problems is difficult. As regards *DYNMOGA*, we employed a trial-and-error procedure on one of the synthetic dataset described below (dataset #1) by computing the normalized mutual information for different combinations of crossover and mutation rates. Figure 4 shows the obtained values. It can be observed that they do not present high variation, thus we set *crossover rate* = 0.8 and *mutation rate* = 0.2, since, in general, high crossover rate and low mutation rate are suggested in the literature. Furthermore, we fixed *elite reproduction* = 10% of the population size, roulette selection function, population size = 100, and number of generations 100.

6.1 Evaluation measures

In order to compare *DYNMOGA* and the other approaches on the synthetic data sets, two validation measures, the *normalized mutual information (NMI)*, already described in the previous section, and the *error rate* are employed. The error rate, as reported in [5], is computed by considering an $n \times k$ indicator matrix Z storing, for each of the n nodes, the community membership to one of the k communities obtained by the algorithm, and a similar indicator matrix G built for the ground truth. The error rate is then defined as the norm $|| ZZ^T - GG^T ||$, which measures the distance between the community structures represented by Z and G.

6.2 Synthetic dataset #1

The first dataset we consider is the benchmark adopted by Lin et al. in [5]. This data set is generated by the authors analogously to the classical benchmark proposed by Girvan and Newman in [40]. The network consists of 128 nodes divided into four communities of 32 nodes each. Every node has a fixed average degree $avgDegree$, and shares a number z of links with the nodes not belonging to its community. Increasing z augments the noise level of the network. Two different values of $avgDegree$ and z have been considered. When $avgDegree = 16$ a less dense network is generated, while with $avgDegree = 20$ we have a denser network. As regards z, a value $z = 5$ generates a more distinct community structure, while fixing $z = 6$, the community structure becomes less clear. Edges are placed with higher probability between a pair of nodes in the same community, and with lower probability between nodes in different communities. These probabilities depend on the value of z. In order to introduce dynamics in $\mathcal{G}$, $nC\%$ of nodes are moved among communities. To this purpose, two different cases have been considered. In the former, the community structure presents soft changes, thus the 10% of nodes are randomly selected from each community, and randomly assigned to the other three communities. In the second case the changes are more substantial, thus the 30% of nodes change their community at each time step. We considered 20 time steps and the values averaged over 50 experiments are reported.

Figures 5 and 6 show the error rate and the normalized mutual information obtained by *DYNMOGA* and $FacetNet$ for the different networks built with the parameter values described. The results for $FacetNet$ have been obtained for $\alpha = 0.8$, value chosen by the authors in their paper [5]. In particular, Figure 5 reports the error rates when (a) z = 5 and percentage of node changes nC = 10%, (b) z = 5 and percentage of node changes nC = 30%, (c) z = 6 and nC = 10%, (d) z = 6 and nC = 30%. Figure 6 reports the normalized mutual information for the same configuration. These two figures point out that *DYNMOGA* obtains lower error rate for all the 4 synthetic networks, at each time step. As regards the NMI, when the fuzziness is low and the community structure changes are mild (Figure 6(a)), the NMI of the two methods are comparable. However, when the fuzziness increases, thus the networks present more dramatic changes, *DYNMOGA* is able to better detect the true community structure. This behavior is clear in Figures 6(c) and (d).

$FacetNet$ needs that the parameter α be fixed by the user. Lin et al. [5] point out that automatically finding the best value of α has not been investigated, and that setting a value pushes the algorithm to prefer a result more biased towards either snapshot quality or temporal smoothness. Figure 7 shows the error rate obtained by $FacetNet$ when the average degree has been fixed to 20, for increasing values of the parameter α, compared with that obtained by *DYNMOGA*. Figure 7(a) shows the results for z = 5 and nC = 30%, while Figure 7(b) for z = 6 and nC = 30%. In the former case, *DYNMOGA* outperforms $FacetNet$ when $\alpha \leq 0.7$. In the latter case, the errors obtained by *DYNMOGA* are always lower, independently of the value of α, except for slight differences at three time steps for $\alpha \geq 0.8$. The figure points out that the choice of α is not an easy task because it influences the quality of the results. As already emphasized, *DYNMOGA*

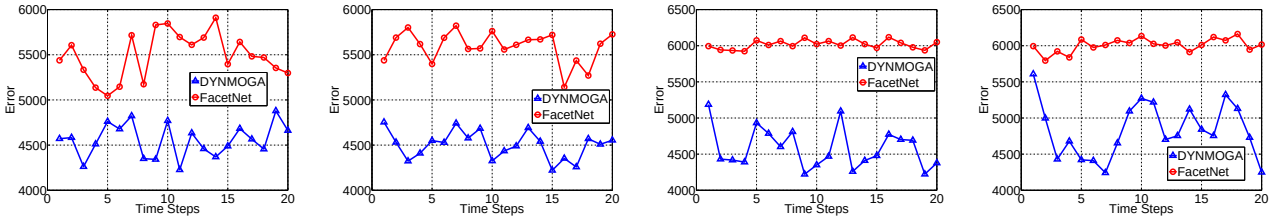


Fig. 5. Error rate on synthetic dataset #1 when avgDegree = 16: (a) z = 5 and nC = 10%, (b) z = 5 and nC = 30%, (c) z = 6 and nC = 10%, (d) z = 6 and nC = 30%.

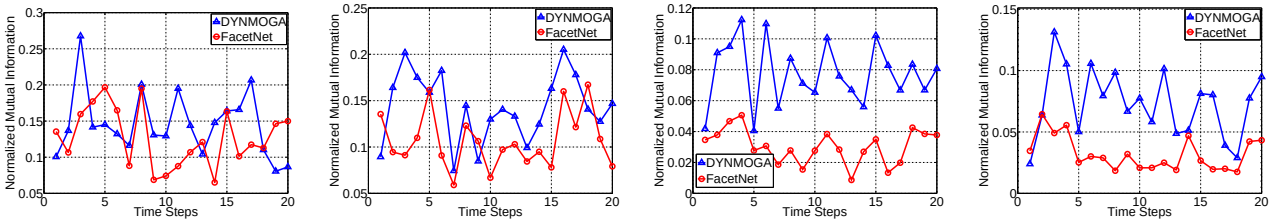


Fig. 6. NMI on synthetic dataset #1 when avgDegree = 16: (a) z = 5 and nC = 10%, (b) z = 5 and nC = 30%, (c) z = 6 and nC = 10%, (d) z = 6 and nC = 30%.

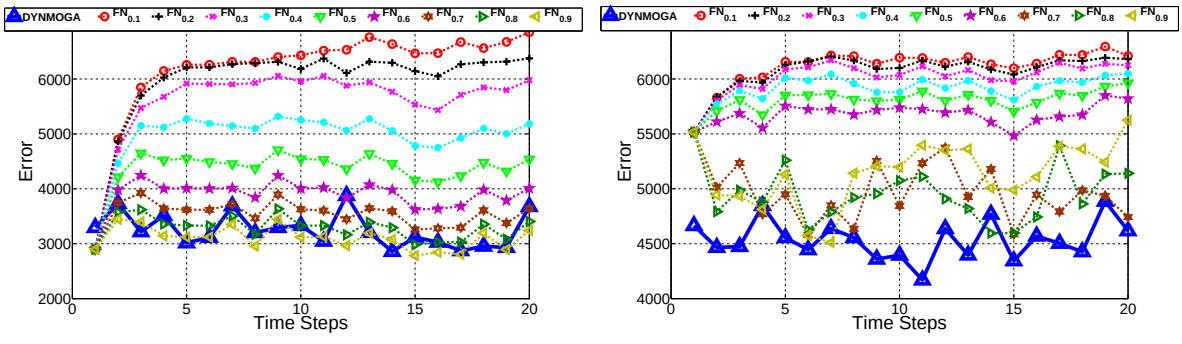


Fig. 7. Error rate on synthetic dataset #1 when avgDegree = 20: (a) z = 5 and nC = 30%, (b) z = 6 and nC = 30%, and α varying between 0.1 and 0.9. $FN_{0.i}$, i=1,...9, stands for FacetNet result with input $\alpha = 0.i$

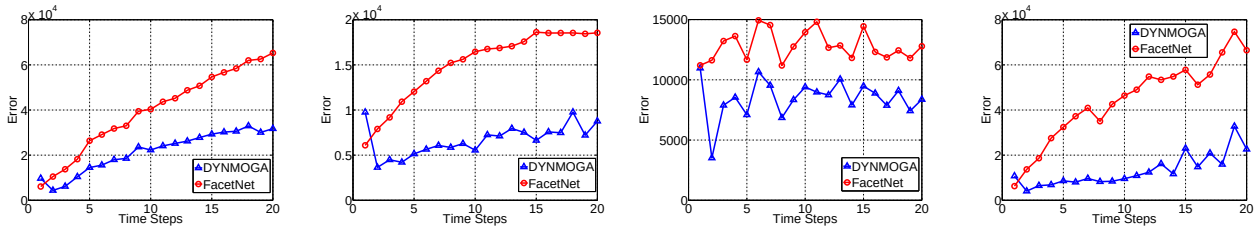
is able to automatically determine the best tradeoff between the two competing objectives.

6.3 Synthetic dataset #2

The second synthetic dataset has been generated by taking into account some main events that may characterize the evolution of dynamic networks [7], [41], [1], [42]. To this end we assumed four types of events, as introduced by Greene et al. in [42]. The events are the following:

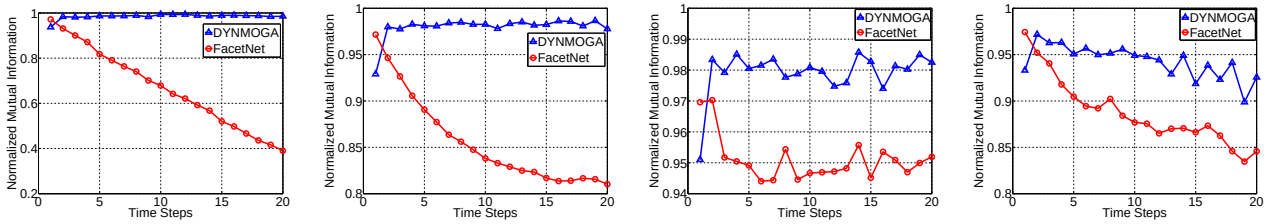
- *Birth and death*: from the second time step on, 10% of new communities are created by removing nodes from other existing communities, and randomly removing 10% of the existing communities.
- *Expansion and contraction*: 10% of communities are randomly selected and expanded or contracted by 25% of their size. When expanded, the new nodes are chosen at random from the other communities.
- *Intermittent communities*: 10% of communities from the first time step are hide.
- *Merging and splitting*: at each time step, 10% of communities are split, 10% of communities are chosen, and couples of communities are merged.

We generated four synthetic data sets for the four different types of events, for 20 time steps. The parameters to the generator have been set such that each network is constituted by 1000 nodes having mean degree of 15 and maximum degree 50, number of communities between 20 and 50, and mixing parameter (percentage of edges between communities) 0.2. Figures 8 and 9 depict the error rate and the normalized mutual information on the four different data sets. The figures clearly show that *DYNMOGA* outperforms $FacetNet$ on all these four types of networks. In particular, it is worth to note that the error rate of *DYNMOGA* is the same or higher than that obtained by $FacetNet$ at the first time step. However, from the second time step on, the error rate of *DYNMOGA* is



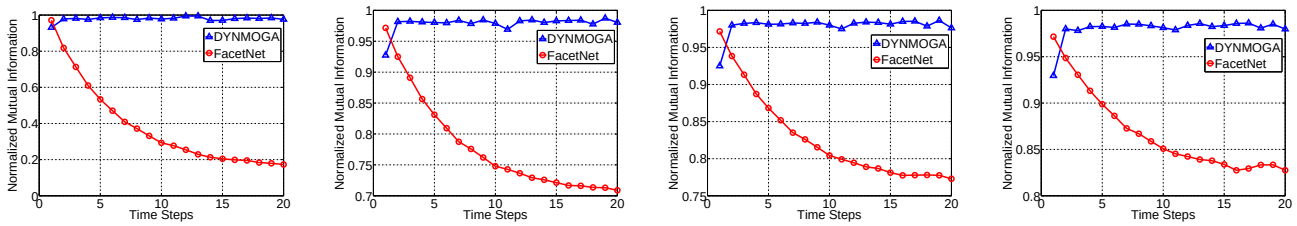
(a) (b) (c) (d)

Fig. 8. Error rate on synthetic dataset #2: (a) birth and death, (b) expansion and contraction, (c) intermittent communities, (d) merging and splitting.



(a) (b) (c) (d)

Fig. 9. Normalized mutual information on synthetic dataset #2: (a) birth and death, (b) expansion and contraction, (c) intermittent communities, (d) merging and splitting.

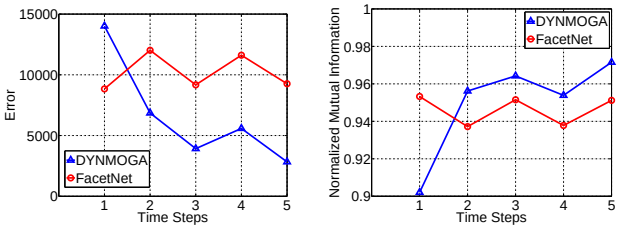


(a) $\alpha = 0$ (b) $\alpha = 0.4$ (c) $\alpha = 0.6$ (d) $\alpha = 1$

Fig. 10. NMIs on Derek-Green's *expansion and contraction* datasets while varying α.

sensibly lower, and the NMI value is notably higher. Note that that the behavior of $FacetNet$ is rather different with respect to the type of event that can occur. In particular, in the case of birth/death of communities (Figure 9(a)) its NMI drastically decreases from 0.99 to 0.4 over the 20 time steps, while *DYNMOGA* maintains an NMI value near to 1. As regards the other three events, the NMI obtained by $FacetNet$ diminishes in a more soft way, reaching a value not less than 0.80, while *DYNMOGA* does not obtain values lower than 0.92.

As already reported, the results for $FacetNet$ have been obtained by fixing $\alpha = 0.8$. In order to check the influence of different values of α also on this kind of data set, Figure 10 shows the NMI values obtained when α assumes the values 0, 0.4, 0.6, and 1, respectively, in case of expansion and contraction events. When $\alpha = 0$ the NMI obtained by $FacetNet$ drastically decreases, since the cost function biases the method to find solutions more similar to the previous time step, completely disregarding the current snapshot. The opposite behavior is obtained when $\alpha = 1$. Although in this case $FacetNet$ has to find the community structure that best fits the current snapshot, independently from the previous time step, it obtains NMI values between 0.97 and 0.83, which are lower than that obtained by *DYNMOGA* (above 0.98 from the second time step on).

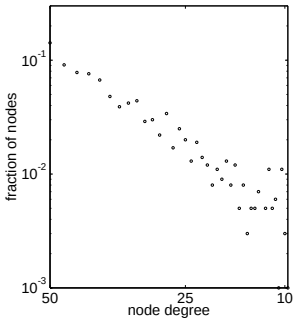


(a) (b)

Fig. 11. Error (a) and Normalized Mutual Information (b) on power law synthetic dataset (merging and splitting).

6.4 Power-law networks

Lancichinetti et al. [43] proposed a new class of benchmarks (LFR benchmark) that extend the Girvan and Newman benchmark by introducing power law degree distributions and different community size. In [44] it has been experimented that many community detection algorithms perform well on the



(a)

Fig. 12. Degree distribution in log-log scale on power law synthetic dataset (merging and splitting).

Girvan and Newman benchmark, but give poor results on the LFR benchmark. Although the network generator of Greene et al. in [42] is based on binary network generation tools proposed in [44], it does not allow to fix the exponent of the power law distribution from which assigning degrees to nodes. Thus, we generated an LFR benchmark constituted by 1000 nodes, average node degree 20, maximum node degree 50, exponent of degree distribution -2, community size distribution -1, and mixing parameter 0.3. The generated networks have only one node with maximum degree 50, while almost 70% of nodes have degree lower than the average degree 20. In order to introduce dynamics, for 5 time steps, 10% of communities were chosen in a random way, and repeatedly they were split (steps 2, 4) and merged (steps 3, 5).

Figure 11(a) and Figure 11(b) show error and normalized mutual information obtained by $FacetNet$ and *DYNMOGA*. We can notice that, after the first time step, *DYNMOGA* is able to better recover the true evolving community structure, and it works well also on power law networks. Figure 12 reports the degree distribution of networks. The figure clearly shows the typical long tail of power law distribution.

6.5 Synthetic dataset #3

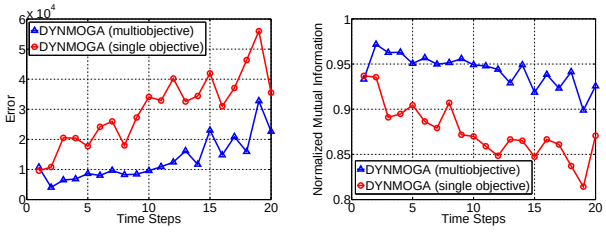
In this section we compare *DYNMOGA* with the method of Kim and Han [3]. It is worth to point out that the results reported for this latter approach are those appearing in [3], and provided by the authors. The comparison has been performed on two kinds of data sets. The first one is a dynamic network of a fixed number of communities (named SYN-FIX). The second one is a dynamic network with a variable number of communities (named SYN-VAR).

SYN-FIX is similar to the synthetic dataset #1. 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 of links with the other nodes of the network. In order to introduce dynamics, 3 nodes are randomly selected from each community in G^{t-1} and randomly assigned to the other three communities. 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, run *DYNMOGA* on them, and computed the *Normalized Mutual Information* to measure the similarity between the true partitions and the detected ones.

Figure 13 shows the average normalized mutual information over the 10 timestamps for SYN-FIX when $z = 3$ (Figure 13(a)) and $z = 5$ (Figure 13(b)), for SYN-VAR when $z = 3$ (Figure 13(c)) and $z = 5$ (Figure 13(d)).

The figure shows the significantly better results obtained by *DYNMOGA* with respect to Kim and Han's algorithm. In fact, for SYN-FIX and SYN-VAR, when $z = 3$, *DYNMOGA* obtains a value which is almost always 1, while the values obtained by Kim and Han's algorithm are around 0.9 for SYN-FIX and aroud 0.7 for SYN-VAR. The differences, however, are much more remarkable when $z = 5$. In this case *DYNMOGA* obtains values above 0.9 for all the timestamps, while Kim and Han method fails to uncover the community structure. In fact, it returns values of normalized mutual information between 0.1 and 0.2.



(a) (b)

Fig. 14. Error and Normalized Mutual Information on synthetic dataset #2 (merging and splitting), single objective versus multiobjective.

6.6 Pareto front solution selection

A characteristic of multiobjective optimization is the generation of a set of solutions. Thus, a single solution out of this set must be selected. There has been a lot of research in this decision making problem, and many different approaches have been proposed [29]. As already stated, *DYNMOGA* prefers, among the Pareto front solutions, that having community structure with the higher modularity value, since this concept has been recognized as the most suitable to interpret the intuitive idea of community. It is worth to note that, the optimization of the two objectives employed by *DYNMOGA*, i.e. modularity and normalized mutual information, does not produce the same results of optimizing the single objective of modularity. In fact, in the former case, the Pareto front contains those non-dominated solutions that try to optimize both the

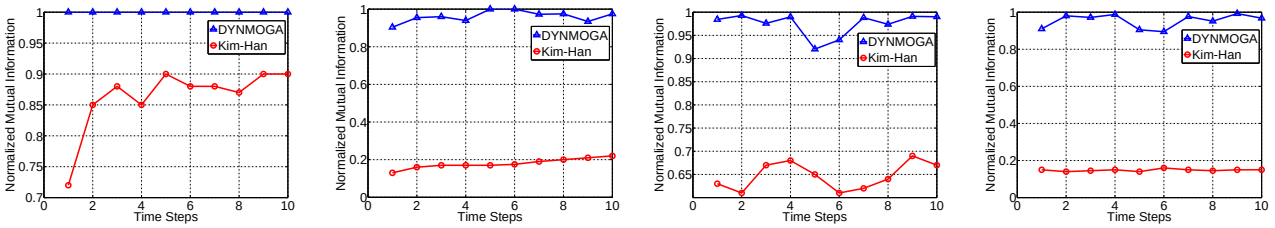


Fig. 13. Normalized mutual information on synthetic dataset #3: (a) SYN-FIX when $z = 3$, (b) SYN-FIX when $z = 5$, (c) SYN-VAR when $z = 3$, (d) SYN-VAR when $z = 5$.

snapshot cost (modularity) and the temporal cost (NMI), which is different than optimizing only the snapshot cost (modularity alone). In order to emphasize that optimizing two objectives gives better results than optimizing only modularity, Figure 14 compares Error and NMI values obtained by *DYNMOGA* for the synthetic dataset #2 (merging and splitting), with those returned when *DYNMOGA* is executed by fixing the temporal cost at zero for all the time steps, thus completely disregarding NMI. The figure clearly points out the better performance of the multiobjective approach with respect to the single objective one.

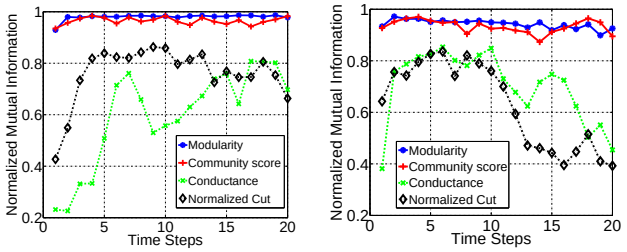


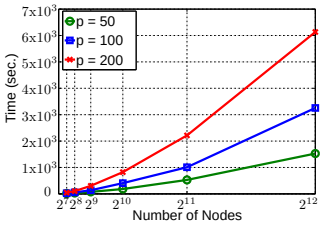
Fig. 15. Normalized mutual information on synthetic dataset #2 with different objective functions: (a) expansion and contraction, (b) merging and splitting.

6.7 Changing objective function to compute snapshot cost

Results presented so far have been obtained by employing modularity as snapshot cost function. As already pointed out, *DYNMOGA* can be considered as a general framework for evolutionary clustering. In fact, it is sufficient to change one of the two objective functions (or both), to implement and test different approaches for analyzing dynamic networks. In this section we consider the other three quality measures for community detection introduced in Section 5, and compare the normalized mutual information values computed by using these scores.

NMI has been computed for the synthetic dataset #2, in particular expansion and contraction, and merging and splitting networks. The results on the other networks are analogous, thus we do not report them for lack of space. Figure 15 shows that modularity overcomes all the other three objectives, though community score works quite well and is comparable with it.

It is worth to notice that it has been shown that the optimization of modularity has a resolution limit that depends on the total size of the network and the interconnections of the modules [45]. This limit implies the important drawback that, searching for partitioning of maximum modularity, may lead to solutions in which important structures at small scales could not be discovered. However, to overcome this problem, Granell et al. [46] introduced a resolution control parameter γ in the modularity formulation. The new formula being $Q_R = \sum_{s=1}^{k}[\frac{l_s}{m} - \gamma(\frac{d_s}{2m})^2]$. When $\gamma = 1$ the original formulation is obtained, while for increasing values of γ, smaller groups of nodes can be found. Thus, despite the criticisms regarding resolution limit problem, the use of modularity as snapshot cost seems the best choice.

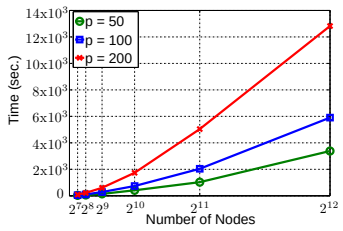


(a) g=50

Fig. 16. Running time of *DYNMOGA* in seconds against number of nodes varying as $\{128, 256, 512, 1024, 2048, 4096\}$ with corresponding number of edges $\{1938, 4018, 8184, 16158, 33026, 65256\}$, on the synthetic dataset #1, z=5, nC=10%, for different combinations of population size p and number of generations $g = 50$.

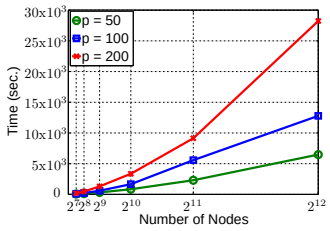
6.8 Scalability Analysis

One of the main criticisms in using Genetic Algorithms, compared with traditional optimization algorithms, is the high execution time required to generate a solution. The major limitation of evolutionary algorithms is, in fact, the repeated fitness function evaluation that, for complex problems could often be prohibitive. The problem is exacerbated when large populations of individuals are used, and, in particular for the multiobjective approach. In our method fitness evaluation is



(a) g=100

Fig. 17. Running time of *DYNMOGA* in seconds against number of nodes varying as $\{128, 256, 512, 1024, 2048, 4096\}$ with corresponding number of edges $\{1938, 4018, 8184, 16158, 33026, 65256\}$, on the synthetic dataset #1, z=5, nC=10%, for different combinations of population size p and number of generations $g = 100$.



(a) g=200

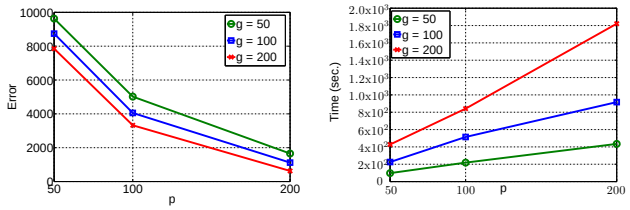
Fig. 18. Running time of *DYNMOGA* in seconds against number of nodes varying as $\{128, 256, 512, 1024, 2048, 4096\}$ with corresponding number of edges $\{1938, 4018, 8184, 16158, 33026, 65256\}$, on the synthetic dataset #1, z=5, nC=10%, for different combinations of population size p and number of generations $g = 200$.

rather efficient, thus the main problem comes from the network size.

To evaluate the scalability of the method we used the synthetic dataset #1, avgDegree=16, z=5, nC=10%, with number of nodes increasing as $\{128, 256, 512, 1024, 2048, 4096\}$, and corresponding number of edges $\{1938, 4018, 8184, 16158, 33026, 65256\}$, population size p, and number of generations g varying in the interval [50,100, 200].

Figures 16-18 report the time requirements for one time step, for the different combinations of p and g. The figures show that the scaling of *DYNMOGA* is lower than quadratic in the number of nodes.

We now want to study the influence of population size and number of generations on the accuracy of the method. It is worth to point out that population size p can be viewed as a measure of the parallel search level a GA supports, since p different solutions are considered at the same time to find the local optimum. The more complex the problem to solve, the larger the population to use because the chance of finding optimal solutions augments with increasing p. Because of the computation requirements, the choice must balance the



(a) (b)

Fig. 19. Error variation (a) and execution times (b) for different combinations of population size p and number of generations g on synthetic dataset #2: merging and splitting, n=1000, m=7728.

quality of the solution versus the computation times. A similar reasoning applies to the number of generations. The longer the algorithm works, the fitter the solution it can obtain. However, after a number of generations, that depends on the problem, continuing the execution does not provide any improvement of the local optimum obtained so far because the algorithm gets stuck in that local optimum. Figure 19(a) shows how the error varies for different combinations of population size p and number of generations g on synthetic dataset #2 (merging and splitting). In particular both p and g assume values in the range [50,100, 200]. The figure points out that the error can drastically drop down from around 9500 with $p = 50$ and $g = 50$ to around 600 if $p = 200$ and $g = 200$. However the running time (Figure 19(b)) sensibly increases. Thus, if the computing resources are limited, the choice of p and g should be done by considering the trade-off between execution time and desired accuracy.

It is worth to point out that, though Genetic Algorithms are naturally suited to be implemented on parallel architectures [47], the capabilities of the approach to deal with very large networks could be degraded because of the demanding computing requirements.

6.9 Real-life data sets

In this section we apply our method to two real-life dynamic networks: *Cell Phone Calls* and *Enron mail*.

Cell Phone Calls: the former data set comes from the *VAST 2008 mini challenge 3: Cell Phone Calls* [1] and consists of cell call phone records among the members of the fictitious Paraiso movement, covering a period of ten days in June 2006. These records have been used to build a network where each node corresponds to a unique cell phone, and an edge between two cell phones is created when a phone call between the two cell phones occurs. For each edge, the day and time of the phone call are reported. The number of cell phones is 400. Thus ten weighted networks can be built by considering the phone calls for each of the ten days. Each edge is labelled with the number of phone calls occurred in that day between two members of Paraiso movement. Five persons are considered the most important in the network, Ferdinando Catalano (node 201) and his brother Estaban (node 6), David Vidro (node 2),

1. http://www.cs.umd.edu/hcil/VASTchallenge08/

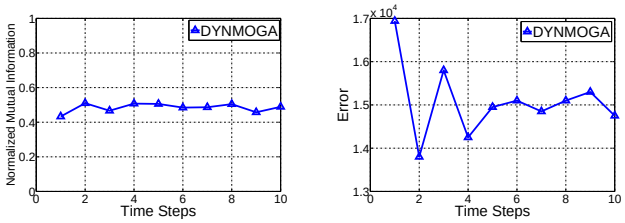
TABLE 2
Cell dataset statistics: T is the number of time steps (10 days on June 2006), M the modularity, $|C|$ the number of communities, $|V|$ the number of nodes, $|E|$ the number of edges, $|E|^*$ the number of distinct undirected edges (i.e. number of different phone calls between two members), Z the average degree, CC the *Watts-Strogatz* clustering coefficient, and B the betweenness of network.

T	M	|C|	|V|	|E|	|E|*	Z	CC	B
1	0.6640	32	370	987	525	2.6250	0.0328	0.2992
2	0.6561	35	373	964	499	2.4950	0.0192	0.2532
3	0.6587	30	374	953	509	2.5450	0.0137	0.2594
4	0.6540	31	374	1013	514	2.5700	0.0179	0.2289
5	0.6626	32	373	991	508	2.5400	0.0160	0.2895
6	0.6651	25	373	963	512	2.5600	0.0207	0.2246
7	0.6571	33	367	936	498	2.4900	0.0118	0.2356
8	0.6329	36	365	1005	511	2.5550	0.0203	0.3127
9	0.6538	34	374	982	518	2.5900	0.0290	0.2991
10	0.6467	32	384	1040	530	2.6500	0.0095	0.2253

and his two brothers Jorge and Juan (nodes 3 and 4). These five core members changed their cell phone numbers between days 7 and 8, therefore, for the last three days their node number changed from 201, 6, 2, 3, 4 to 301, 307, 310, 361, 398, respectively.

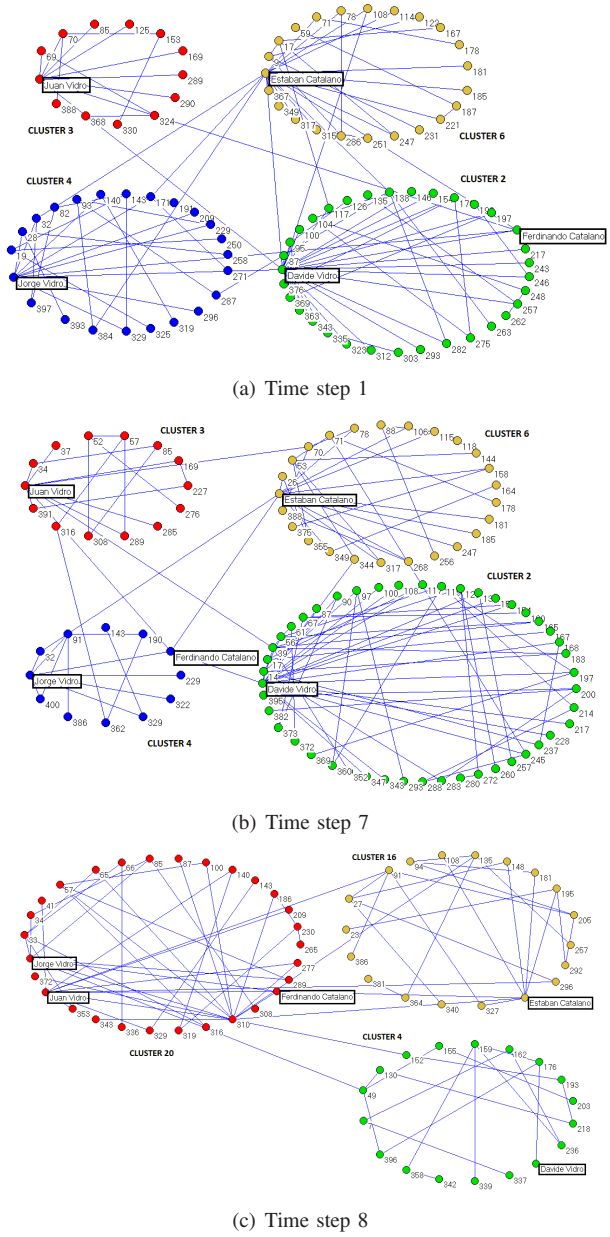
Some statistical informations regarding the network are reported in Table 2. Since the true community structure is not known, we followed the same approach of Lin et al. [5]. We first considered the overall network and computed the community structure by applying only the first step of our method. The resulting network division had an average modularity value of 0.52 and an average number of communities equal to 25.

After that, the clustering obtained on the overall network was considered as the ground truth division, and both the error rate and the normalized mutual information have been computed by executing *DYNMOGA* on the ten networks. The values obtained are reported in Figure 20. It can be observed that *DYNMOGA* finds communities that constitute a balance between the snapshot and the temporal costs. In fact, the similarity among the groups obtained for the overall network and those computed for each time step is around 0.5 (Figure 20(a)). It is worth to note that, for each time step, both the modularity values and the number of communities found are higher than those computed on the overall network. In fact we obtained a number of communities between 30 and 36 for the different time steps, with a modularity value varying from 0.64 to 0.66 (see Table 2). This means that the execution of the method on the dynamic network provides a more structured division and allows for a deeper analysis of the network.



(a) NMI (b) Error

Fig. 20. Normalized mutual information and error rate of the CELL dataset.



(a) Time step 1

(b) Time step 7

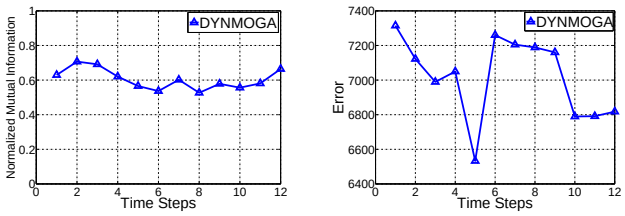
(c) Time step 8

Fig. 21. Community snapshots on CELL data set at different time steps.

Finally, in Figure 21, the communities of the five most important members, described above, at the first time step are visualized, along with the evolution of their communities at time steps 7 and 8. The three figures clearly point out that these five persons had a central role until the seventh day, directly communicating with almost all the other members of the same group. At the 8th day, instead, this peculiarity is lost, as expected and obtained by other studies relative to this network [48].

Enron mail data set: the second real dataset is an email collection from a US enterprise with potential anomalous email communications spanning over a time range of about 3 years (i.e., from 1999 to 2002). The original dataset[2] contains around 517,431 emails from 151 users distributed in 3500

2. available at http://www-2.cs.cmu.edu/∼enron/



(a) NMI (b) Error

Fig. 22. Normalized mutual information and error rate for the Enron dataset.

TABLE 3
Enron dataset statistics: T is the number of time steps (2001 monthly partitioning), M the modularity, $|C|$ the number of communities, $|V|$ the number of nodes, $|E|$ the number of edges, $|E|^*$ the number of distinct undirected edges (i.e. number of different mails between two employees), Z the average degree, CC the *Watts-Strogatz* clustering coefficient, and B the betweenness of network.

T	M	|C|	|V|	|E|	|E|*	Z	CC	B
1	0.6368	11	96	1070	180	2.3841	0.4415	0.0850
2	0.6803	7	93	1559	204	2.7020	0.5406	0.1002
3	0.6550	12	97	1844	218	2.8874	0.4646	0.0984
4	0.6571	12	108	1869	257	3.4040	0.4430	0.1462
5	0.5699	15	125	1919	292	3.8675	0.4993	0.4384
6	0.6943	10	120	1001	231	3.0596	0.3959	0.1389
7	0.6530	10	109	1325	252	3.3377	0.4882	0.1833
8	0.5356	9	131	2270	396	5.2450	0.4783	0.4331
9	0.6324	10	128	3152	361	4.7815	0.4950	0.3070
10	0.5302	13	135	8693	575	7.6159	0.4957	0.3494
11	0.5707	9	127	6276	469	6.2119	0.5062	0.1915
12	0.6152	8	113	2146	325	4.3046	0.4519	0.2467

folders. For our tests, we considered a cleaned version of this corpus [3] described in [49], and containing a subset of 252,759 emails from 151 employees. We further reduced the size to about 50,000 messages by considering only emails exchanged among Enron's employees.

In order to perform a monthly-based analysis we concentrated on the year 2001 since it encompasses the maximum number of emails (i.e. more than 33,000). We split it in 12 subsets, one for each month, by following the same approach used for the CELL dataset. The number of communities obtained on the overall network has been six. Figure 22 shows the NMI and Error for the 12 time steps. Also in this case, the dynamic approach provides a finer division of the network, as pointed out in Table 3, where some statistical informations regarding the network for each of the 12 time steps are reported.

7 Conclusions

A multiobjective method based on genetic algorithms 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 at the current time step, and the drift from one time step to the successive. The proposed approach can be considered as a general framework for evolutionary clustering since changing one of the two objective functions (or both) allows to implement and test different criteria for the analysis of dynamic networks. Experimental results on several kinds of synthetic data sets showed the good performance of our approach compared with other state-of-the-art methods.

It is worth to point out that genetic algorithms, compared with traditional optimization algorithms, require high execution time to generate a solution. Experiments showed that increasing population size p and number of generations g produces a positively influence on the accuracy of the method. However the running time sensibly increases. Thus, if the computing resources are limited, the choice of p and g should be done by considering the trade-off between execution time and desired accuracy.

3. available at ftp://ftp.isi.edu/sims/philpot/data/enron-mysqldump.sql.gz

8 Ackowledgments

We wish to thank Yu-Ru Lin for providing us the $FacetNet$ software, including the generation of the synthetic data set #1, Derek Green for the software generator of the synthetic dataset #2, and Min-Soo Kim for the synthetic data set generator, and the results obtained by his method on these 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*, vol. 3, no. 4, p. Paper 16, 2009.

[2] Y. Chi, X. Song, D.Zhou, K.Hino, and B. Tseng, "On evolutionary spectral clustering," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 4, Article 17, 2009.

[3] 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)*, 2009, pp. –.

[4] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks," in *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD'06)*, 2006, pp. 611–717.

[5] Y.-R. Lin, S. Zhu, H. Sundaram, and B. L. Tseng, "Analyzing communities and their evolutions in dynamic social networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 2, Article 18, 2009.

[6] L. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws,shrinking diameters and possible explanations," in *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD'05)*, 2005, pp. 177–187.

[7] G. Palla, A. Barabasi, and T. Vicsek, "Quantifying social group evolution," *Nature*, no. 466, 2007.

[8] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult, "Monic - modeling and monitoring cluster transitions," in *Proc. Int. Conf. on Know. Discovery and Data Mining (KDD'06)*, 2006, pp. 706–711.

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

[10] 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)*, 2007, pp. 677–685.

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

[12] ——, "Dirichlet process based evolutionary clustering," in *Proc. of the 8th IEEE International Conference on Data Mining (ICDM'08)*, 2008, pp. 648–657.

[13] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proc. of the 12th ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD'06)*, 2006, pp. 554–560.

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

[15] J. Leskovec, K. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proc. Int. World Wide Web Conference (WWW 2010)*, 2010, pp. 631–640.

[16] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, p. 026113, 2004. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0308217

[17] L. Danon, A. Daz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics*, vol. P09008, 2005.

[18] D. Datta, J. Figuera, C. 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)*, 2008, pp. 625–632.

[19] G. N. Demir, A. S. Uyar, and S. G. Ögüdücü, "Multiobjective evolutionary clustering of web user sessions: a case study in web page recommendation," *Soft Computing*, vol. 14, no. 6, pp. 579–597, 2010.

[20] C. Pizzuti, "A multiobjective genetic algorithm to find communities in complex networks," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 418–430, 2012.

[21] J. Handle and J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, 2007.

[22] F. Folino and C. Pizzuti, "A multiobjective and evolutionary clustering method for dynamic networks," in *Proc. of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM10),*, 2010, pp. 256–263.

[23] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Science*, vol. 328, pp. 876–878, 2010.

[24] P. Holme and J. Saramaki, "Temporal networks," 2011. [Online]. Available: http://arxiv.org/pdf/1108.1780.pdf

[25] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Identifying evolving groups in dynamic multimode networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 1, pp. 72–85, 2007.

[26] T. Y. Berger-Wolf, C. Tantipathananandh, and D. Kempe, "Dynamic community identification," in *in P.S. Yu et al. (eds.) Link Mining: Models, algorithms, and Applications*, 2010, pp. 307–336.

[27] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

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

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

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

[31] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[32] J. H. Holland, *Adaptation in Natural and Artificial Systems*. *Univ. of Michigan Press*, Ann Harbor Mich., 1975.

[33] R. Kannan, S. Vempala, and A. Vetta, "On clustering: Good, bad and spectral," *Journal of the ACM*, vol. 51, no. 3, pp. 497–515, 2004.

[34] Y. Park and M. Song, "A genetic algorithm for clustering problems," in *Proc. of 3rd Annual Conf. on Genetic Algorithms*, 1989, pp. 2–9.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms - Second Edition*. Mit Press, 2007.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[37] M. T. Jensen, "Reducing the run-time complexity of multiobjective eas: The nsga-ii and other algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 503–515, 2003.

[38] L. Can-Tao and H. Bao-Gang, "Mutual information based on renyi's entropy feature selection," in *Proc. of the IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 2009)*, 2009, pp. 816–820.

[39] S. K. Smit and A. E. Eiben, "Parameter tuning of evolutionary algorithms: Generalist vs. specialist," in *Applications of Evolutionary Computation, Springer*, 2010, pp. 542–551.

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

[41] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'07)*, 2007, pp. 217–226.

[42] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *International Conference on Advances in Social network Analysis and Mining (ASONAM'10)*, 2010, pp. 176–183.

[43] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 046110, 2008.

[44] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Phys. Rev. E*, vol. 80, no. 056117, 2009.

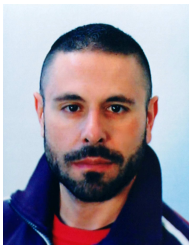
[45] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proc. National Academy of Science, USA*, vol. 104, no. 36, 2007.

[46] C. Granell, S. Gómez, and A. Arenas, "Unsupervised clustering analisys : A multiscale complex network approach," *Journal of Bifurcation and Chaos*, vol. 22, no. 7, p. 1230023, 2012.

[47] M. Tomassini, *Parallel and Distributed Evolutionary Algorithms: A Review*. in Evolutionary Algorithms in Engineering and Computer Science, J. Wiley and Sons, Chichester et al. eds., 1999.

[48] A. Perer, "Using socialaction to uncover structure in social networks over time," in *Proc. of IEEE Symposium on Visual Analytics Science and Technology (VAST '08)*, 2008, pp. 213–214.

[49] J. Shetty and J. Adibi, "The enron email dataset database schema and brief statistical report," 2004. [Online]. Available: http://www.cs.cmu.edu/~enron/



Francesco Folino is currently researcher at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy. He graduated in Computer Science Engineering in 2003, and he holds a Ph.D. in Computer Science Engineering from the University of Calabria (Italy) in 2006. His research interests mainly concern the fields of Information Systems, Data Mining, Process Mining and Social Network Analysis. He is involved in several projects at ICAR-CNR concerning applications of data mining and knowledge discovery techniques.

Clara Pizzuti received the Laurea degree in Mathematics from the University of Calabria, Italy. She is a senior researcher at the Institute of High Performance Computing and Networking (ICAR) of the Italian National Research Council (CNR). Since 1995 she is also a contract professor in the department of Computer Science at the University of Calabria. In the past, she worked in the research division of a software company on deductive databases, advanced logic based systems, and abduction. She has published more than seventy papers in conference proceedings and journals. Her research interests include knowledge discovery in databases, data mining, data streams, bioinformatics, e-health, social network analysis, evolutionary computation, genetic algorithms, and genetic programming. She is serving as program committee member of international conferences, and as reviewer for several international journals.