IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 January 2023, accepted 10 March 2023, date of publication 14 March 2023, date of current version 17 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3257021

RESEARCH ARTICLE

Probabilistic Community Detection in Social Networks

STAVROS SOURAVLAS[1,2], (Member, IEEE), SOFIA D. ANASTASIADOU[2], THEODORE ECONOMIDES[1], (Student Member, IEEE), AND STEFANOS KATSAVOUNIS[3]

[1]Department of Applied Informatics, University of Macedonia, 54636 Thessaloniki, Greece
[2]Department of Midwafery, School of Health Sciences, University of Western Macedonia, 50020 Ptolemaida, Greece
[3]Department of Production and Management Engineering, Democritus University of Thrace, 69100 Xanthi, Greece

Corresponding author: Stavros Souravlas (sourstav@uom.edu.gr)

This work was supported by the Hellenic Academic Libraries Link (HEAL-Link).

ABSTRACT The detection of community structures is a very crucial research area. The problem of community detection has received considerable attention from a large portion of the scientific community. More importantly, these articles are spread across a large number of different disciplines, from computer science, to statistics, and social sciences. The analysis of modern social networks becomes rather cumbersome, as their size and number keeps growing larger and larger. Moreover, in the modern communities, users participate in large number of groups. From the network perspective, efficient methods should be developed to automatically identify overlapping communities, that is, communities with overlapping nodes. In this work, we use a probabilistic network model to characterize and identify linked communities with common nodes. The innovative idea in this work is that the communities are represented as Markovian networks with continuously changing states. Each state represents the number of users within a cluster, that have specific characteristic classes. Based on the current state, we introduce a fast, linear on the number of newly added users, approach to estimate the probability of each cluster to be homogeneous in terms of sets of user characteristics and to determine how well the new user fit within a community. Because of the linear computations involved, our proposed probabilistic model can detect communities and overlaps with low execution time and high accuracy, as shown in our experimental results. The experimental results have shown that our probabilistic scheme executes faster and provides more robust communities compared to competitive schemes.

INDEX TERMS Community detection, social networking, closed networks, linear complexity.

I. INTRODUCTION

Nowadays, more than ever before, the social media and the social communities produce vast data amounts, which are used in a variety of ways: Recommendation engines, marketing, crime detection. Examples abound: Innovative companies like Netflix and Amazon have used predictive analytics for years as their basis to develop highly accurate recommendation systems (like the products recommended by Amazon or the movies and television shows suggested specifically to each Netflix customer). The data collected by users as they browse Facebook or Instagram is used to match them with a large number of companies which offer products and services that, based on statistical models, they would probably be interested. Facebook and Instagram maintain the biggest and most comprehensive databases of personal information. These databases are expanding rapidly every day.

The meaningfulness and usability of the data retrieved from the social networks depend highly on the existing relationships among the social media users. Apparently, people that follow the same groups are likely to be suggested similar products or services. In other words, the extraction of meaningful relationships among the billions of the social media provides high value to many applications. The *community detection* paradigm mainly uses datasets that include the likes, the opinions, and the current relationships among social media users, in order to detect underlying clusters

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Guidi.

and to verify the accuracy of the already formed clusters within an entire network. In this regard, community detection can facilitate the uncovering of hidden relationships between social network users and it has nowadays become one of the most important research areas in the field of computing and social networking.

The typical view of a community is that it is a set of user profiles of same interests and likes. Such a community keeps enlarging by searching, proposing and adding new members with the same characteristics, that are likely to interfere with the existing members. Typically, the social communities are represented as clusters of an entire network. Most of the community detection techniques are based on modeling the communities as graph structures [1]. The analysis of modern social networks becomes rather cumbersome, as their size and number keeps growing larger and larger, the network topologies are rather irregular (which causes difficulties in balancing the computational load among the available processors) and the networks themselves are regularly updated (users of one community join more communities or depart from certain communities). In this sense, it is necessary to apply more computationally efficient strategies to reduce the excessive computational costs.

In this work, we present a new approach to community detection, which is based on modeling communities as probabilistic networks. In this way, we can introduce conditional dependencies among clusters within a community and identify possible overlaps, which can be considered as new communities or *overlapping communities*. The main idea is to split the communities into small clusters and to examine the probability of user transitions (a transition here means that a user may fit better to another cluster than the one he/she currently belongs) from one cluster to another. The main contributions of our work are the following:

1) It proposes a model where the computations involved are linear.
2) The model can be extended to a large number of clusters and users because users with similar characteristics can be handled in a similar manner.
3) The model provides strict rules which determine in which cases there are overlaps.
4) The simulation results indicate that it can produce robust and homogeneous, in terms of user characteristics, clusters.

The rest of this paper is organized as follows: Section II presents the related work on community detection. Section III presents the preliminaries of our work and the problem formulation. In Section IV, we present the details of our probabilistic community detection scheme. In Section V we present and discuss our experimental results. Section VI concludes this work and present aspects for future work.

II. RELATED WORK

Typically, a community can be defined as a set of users with similar characteristics, opinions and likes [1], [2]. The nodes of a community are quite similar to each other and dissimilar to nodes outside the community [3]. However, there are cases where there may be considerable similarities among users of different communities. In this sense, these users can possibly be members of the same community. In other words, communities may be *overlapping*, that is, they may include common members. Researchers in the field of community detection focus their efforts on finding such overlaps because the connections between multiple members of different communities may be a strong indication that these communities may well be joined to a single one. Although there has been some work on disjoined communities (a good example is the work of Staudt and Meyerhenke in [4]), the majority of the latest algorithms study the problem of overlapped communities.

In this section, we divide the community detection schemes into three main categories: graph-based (GB), machine learning (ML), and Probabilistic network schemes. Also, different approaches exist: for example, in [5] the authors define a community as a set of nodes spanning the same subspace. A community is differentiated from another community based on the difference of the subspace spanned by each.

The GB schemes are strictly based on graph theory and data structures while the ML schemes introduces ML approaches for community detection. Our proposed work belongs to the probabilistic community detection category.

A. GRAPH-BASED SCHEMES

The graph-based schemes can be divided into two categories: (a) top-down approaches and (b) bottom-up approaches. The *top-down* approaches divide the overall network into small groups, in order to detect communities (link partitioning). In [6], the authors proposed the *Weighted Community Clustering (WCC)*, which computes the level of cohesion of a set of nodes S. A similar triangle-based approach, called $k-$mutual-friend subgraph was also used in [7]. Another solution to the aforementioned issue is the use of a measure called the "belonging degree", a measure which is based on the link coefficients and indicates how well a node fits within a community. Specifically, in [8] and [9] the nodes are considered to be parts of a community by comparing the belonging degrees to pre-specified threshold values. The picaso strategy introduced by Qiao et al. [10] is another top-down approach which detects overlapping communities using a modularity-based *mountain* model. Specifically, the network is divided into chain groups (top-down) which are sorted by the weights of edges. Based on the community features, some edges fall down and others raise like mountains. New communities are formed by the mountains produced. An update-modularities phase is also included.

The bottom-up approaches start from local structures and expand to the overall network. While this process expands, more communities are formed. *Optimization* bottom-up approaches have been used for community detection. Generally, these strategies consider communities as subgraphs identified by the maximization of a measure called the nodes

fitness. This measure is based on the total internal and external degrees of the nodes of a group (or module). The aim of optimization schemes is start from a specified node and to find subgraphs such that the fitness value is lowered upon the inclusion or elimination of a new node [11], [12], [13], and [14]. The *label propagation* is a bottom-up approach, which assigns labels to previously unlabeled data points. It starts from a few labeled nodes and while proceeding, the nodes adopt the labels currently used by the majority of the neighboring nodes [3], [15]. Another interesting label-based approach is [16], where the authors use a binary label-based representation scheme and propose an evolutionary computation-based algorithm called evolutionary-based local community detection (ELCD) algorithm to detect local communities in complex networks. Finally, the *clique percolation* bottom-up approaches assume that a community consists of fully connected subgraphs, may overlap. The community detection is based on searching and identifying neighboring cliques. Initially, all the cliques in the network are found and represented in the graph by a vertex. When two cliques share a predefined number of members, then their corresponding vertices are connected. Thus, connected vertices on the graph represent network communities [17], [18], and [19]. The approach of using subgraphs and representing them as a single node will be used in this work as well.

B. MACHINE LEARNING SCHEMES

A more recent trend in community detection uses ML strategies [20]. Although the majority of works that use ML do not focus on overlapping communities, there are certain approaches that have been employed. The Autoencoders (AEs) and variational graph autoencoders (VGAEs) are very common in community detection because of their ability to represent efficiently nonlinear real-world networks. The deep neural network based auto-encoders are used to learn data codings (representations of data sets) in an unsupervised manner [21], [22], [23], [24], [25], [26]. An interesting AE-based approach to the detection of overlapping communities is DeCom [27]. The idea is to use autoencoder pipelines to extract overlapping communities in large-scale networks. The DeCom strategy is composed of 3 phases: The first phase is the seed selection phase, where the features of the network's vertices are learned in a stepwise greedy approach. In the second phase, the topology is learned; the goal here is to get multiple communities in a completely unsupervised manner, where each community is defined by the seed. The communities are expanded from each seed using a random walk with the seed as the starting point. In the third phase, the formed clusters are refined by modularity optimization. The network structure is reconstructed and better representations are detected. Similar approaches are found in [28], [29]. The main advantages of the auto-encoder based method is that it is a good method to overcome the efficiency problems of linear optimization and to combine the topology and node content information, which are considered as objective functions of a linear model. Also, they work very well when we try to map data points into the lower dimensional spaces.

The Graph Neural Networks (GNNs) are also used in community detection and they are a powerful tool used to extract the spatial localization. The GNNs are particularly useful in cases where the numbers of nodes connections vary and the nodes are not ordered (irregular on non-Euclidean structured data). Their main drawback is that the overall performance degrades as more graph convolutional layers are added. This affects the quality of the neighbor propagation. The reason behind this issue is that the addition of many convolutional layers is similar to multiple Laplacian smoothing procedures. This in fact smooths the node features and the nodes from different clusters become indistinguishable (in the sense that their attribute values converge and they become equal). To resolve this issue, many proposed strategies try to keep the number of layers to relatively small numbers (up to three). Some examples are [30], [31], [32]) An interesting work based on GNNs is found in [32]. The authors addressed the issue of overlapping communities using four datasets as benchmark. They combine the GNNs with the Bernoulli-Poisson model which is able to produce a variety of community topologies (e.g. nested, hierarchical) and lead to dense overlaps between communities. The main advantage of the GNN approaches is that they consider both data structure and node features as they can encode the graph structure and node features for the representation. This information is very important in data representation learning.

Recently, the Generative adversarial networks (GANs) have been introduced to community detection problems. Typically, the GANs combine two neural networks: a generator and a discriminator. The latter discriminates if an input sample comes from the prior data distribution or from the generator and the former is trained to generate the samples in such a way that the discriminator is convinced that the samples come from a prior data distribution. The main drawback of GANs is that they cannot handle dense overlapping. An idea to resolve this issue is to combine the effective GANS with affiliation graph models (AGM), which can model densely overlapping community structures. Thus, the performance of GANs and the direct vertex-community membership representation of AGMs join forces to solve the dense overlapping issues. An interesting such approach is [33], which jointly solves overlapping community detection and graph representation learning. Two models are trained: a generator that tries to produce a vertex subset to compose a motif (clique) and a discriminator which tries to deduce if the vertex subset is a real or unreal motif. The two models are combined by participating in a game where the generator tries to fool the discriminator by approximating a conditional probability value (the preference distribution of motifs covering an edge over all the remaining motifs), in order to produce the vertex subsets which are closer to the real motifs covering a specific edge. Accordingly, the discriminator distinguishes the ground-truth motifs from the ones produced by the generator. Given the distribution samples and the samples produced by

the generator, the discriminator tries to maximize the probability of correct classification of these samples. A similar approach can be found in [33]. The Deep learning strategies (AE- or GNN- based) focus on preserving the structure relationship and generally ignore the latent data distribution of the representation. Using GANs, the new data generated have the same distribution as real data, and this is a powerful tool for analyzing network data.

There are two important disadvantages of the ML strategies, which still need to be addressed: (1) Their vast majority do not deal with overlapped communities, and (2) The analysis of modern social networks with ML schemes becomes rather cumbersome, as their size and number keeps growing larger and larger. Generally, there is still a growing need for overlapping community detection algorithms that do not fail to scale linearly with the increasing number of users and the growing complexity of their relationships. Although ML strategies can produce more accurate results compared to probabilistic schemes, these aforementioned issues are still a burden and subject of future concern. A detailed analysis of machine learning community detection schemes can be found in [34].

C. PROBABILISTIC COMMUNITY DETECTION

The probabilistic community detection is based on the idea of computing the probability of a node to be a member of a community. To do so, some form of a probabilistic model is employed [35]. Several probabilistic methods have been proposed: in [36], the authors proposed the First Passage Probability Method (FPPM) strategy, which is equipped with a similarity measure that incorporates the complete structural information within the maximal step length. The diameter of the network is chosen as an appropriate boundary of random walks, which is adaptive to different networks. In [37], the authors proposed a PDS (probabilistic data structure) which is used to reduce the memory requirements for storage and the time required to retrieve and process the data related to community detection. They proposed a Bloom filter has been used for clustering and Quotient filter has been used for storage and access of cluster nodes. In [38], the authors propose a unified Bayesian generative model to detect generalized communities. An efficient Markov chain Monte Carlo (MCMC) algorithm is proposed in [39], which makes clear the need to perform reversible jump MCMC on the number of clusters. PODCD [40] is another probabilistic method, which considers the task of detecting communities as a non-negative matrix factorization problem. The proposed method considers the more likely assumption of dense connections between communities and utilizes a probabilistic model to control the dynamics of community structure.

Generally, a disadvantage of the probabilistic methods is that they consider random walks of maximum length. This poses difficulties in distinguishing the cluster and topology information. Also, this approach can be very expensive in terms of execution time. The proposed methods that were discussed in this paragraph try to resolve this issue by employing random walks with a fixed number of steps. This may improve the situation, however there is no fixed number of steps that can be employed universally for all the existing clusters, because of their different sizes and their irregular topologies. A different approach, which was proposed in [36] uses the cluster diameter. Again, there may be differences between different cluster, but this method at least can give a clear picture of the cluster topology and works relatively faster.

Our proposed scheme is a Markov-based probabilistic model, which consider each cluster as a probabilistic network with changing states. For all the possible states, we perform linear computations to find the probability of having pure (or homogeneous) clusters, based on their dominant classes of characteristics. Then, we use the current statistics of each cluster combined with the computer probabilities, to determine if the newly added users fit better to one cluster or another, or if they should form a new cluster. In this way, we avoid heavy computations and issues regarding the topology of each cluster. We summarize the main advantages of our probabilistic scheme as follows:

1) The computations involved are linear.
2) The model can be employed for any topology.
3) We do not need to perform random walks, instead we only need to examine the connections of the newly added users within and outside a given cluster.

III. PRELIMINARIES

Let $G = (V, E)$ be a weighted, undirected graph, where V and E are the sets of nodes and edges, respectively. Nodes represent users and edges represent the relationship between two users (e.g., friendship in Facebook). The *similarity matrix* $s(x, y)$ is a matrix that shows the extent to which a group of i classes of user characteristics (the term ''characteristics'' includes features, opinions, likes etc) are similar between users x and y. Each class has m discrete characteristics (for example, a class of characteristics regarding education, a class of characteristic regarding employment etc).

Each value of $s(x, y)$ value lies in the interval $[0 \ldots 1]$. As will be described in the Experimental Results and Discussion section, such classes of user data can be found in the real data sets collected for various social networks. In the example of Fig. 5, there are 3 community clusters with a total of 14 users. There are $i = 3$ classes of characteristics. For users 0 and 1, we have: $s(0, 1) = [0.8, 0.7, 0.88$. The following subsection provides the necessary definitions.

A. DEFINITIONS USED IN THIS WORK

Definition 1: The connectivity degree $\mathcal{C}$ of a cluster c is a matrix indexed $[0, \ldots, i-1]$ (i elements) that stores the average values of the class similarities among all the cluster

couples (x, y), that is:

$$\mathcal{C}(c) = \left[\sum_{\substack{\text{for all } i \\ \text{all pairs } (x,y)}} \frac{s_i(x,y)}{n(x,y)} \right], \tag{1}$$

where $n(x, y)$ is the number of user pairs within a cluster c. For the example of Fig. 5, for cluster 0 we have:

$$\mathcal{C}(0) = \left[\frac{0.8+0.75+0.8}{3}, \frac{0.7+0.8+0.7}{3}, \frac{0.88+0.9+0.85}{3} \right] = [0.78, 0.73, 0.87].$$

Definition 2: The *dominant class*, $\mathbf{i}_c$, within a cluster c is the class whose value dominates the values of the other classes of characteristics within the cluster. It indicates which class characteristic is the one that more strongly relates the members of a cluster.

For cluster 0 of Fig.5, the class $\mathbf{i} = 2$ dominates the values of classes $i = 0$ and $i = 1$: for pair (0,1): $0.7 < 0.8 < 0.88$, for pair (0,3): $0.75 < 0.8 < 0.9$, for pair (2,3): $0.7 < 0.8 < 0.85$.

Definition 3: The *belonging degree* $\mathcal{B}_c(\mathbf{x})$ of a specific user $\mathbf{x}$ to a cluster c is the ratio of the average class values of x within c to the average of the class values of x outside c:

$$\mathcal{B}_c(\mathbf{x}) = \frac{\sum_{\text{all pairs } (x,y) \text{ within c}} s_i(\mathbf{x}, y)/e_{\mathbf{x}}}{\sum_{\text{all pairs } (x,y) \text{ outside c}} s_i(\mathbf{x}, y)/e'_{\mathbf{x}}}, \tag{2}$$

where $(\mathbf{x}, y)$ are all the links involving a specific user $\mathbf{x}$ $e(\mathbf{x})$ is the number of links of this user within cluster c and $e'(\mathbf{x})$ is the number of links of $\mathbf{x}$ outside c.

For the example of Fig. 5, we have three classes and for node 3 we have:

$$\mathcal{B}_0(\mathbf{3}) = \left[\frac{(0.8+0.75)/2}{0.9/1}, \frac{(0.7+0.8)/2}{0.85/1}, \frac{(0.9+0.85)/2}{0.88/1} \right] = \left[\frac{0.775}{0.9}, \frac{0.75}{0.85}, \frac{0.875}{0.88} \right] = [0.86, 0.88, 0.99]$$

By its definition, the belonging degree can show if a user is more strongly linked to a different cluster than the one he/she currently belongs:

- If all the elements of the matrix $\mathcal{B}_c(x) \geq 1$, then the user is absolutely more related to cluster c.
- If at least the dominant class value in $\mathcal{B}_c(x) \geq 1$, then the user is more strongly related to cluster c based on the dominant class of characteristics.
- If the dominant class value in $\mathcal{B}_c(x) < 1$, then the user is more strongly related to another cluster, other than c, with a different dominant class of characteristics than the one found in c.

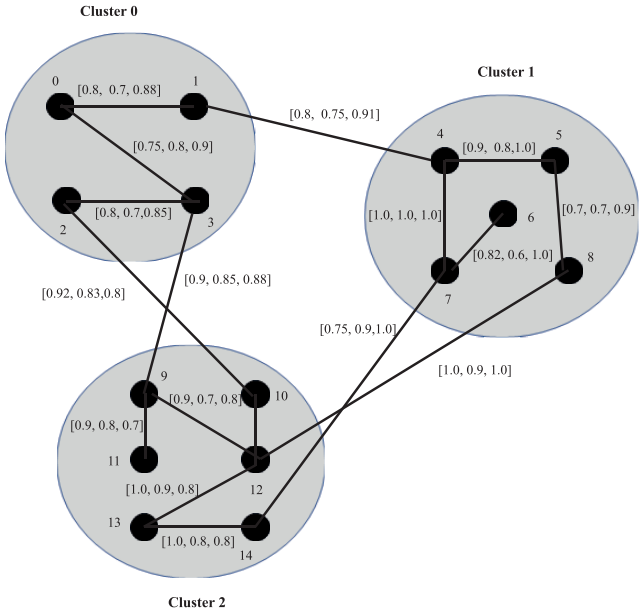

FIGURE 1. A community with 3 clusters.

- If $\mathcal{B}_c(x) < 1$ for all the elements of this matrix, then this user is more closely related to a different cluster, other than c.

In our example, user 3 has a belonging degree of [0.86, 0.88, 0.99] which indicates that means that he/she is more strongly linked to cluster 2, based on class $i = 0$. In this case, since class 0 dominates in cluster 2, then a *transition* of user 3 to cluster 2 is necessary. The definition of transition is given next:

Definition 4: A *transition* is a movement of one user from a cluster to another, based on its belonging degree. Such a movement will change the status of the overall network, as will be described later in the following subsection, where we typically formulate the problem.

B. PROBLEM FORMULATION

In this work, we represent the network as a Markovian network. The *state* of this network can be defined by a vector $\mathbf{N} = (N_0, N_1, N_2 \ldots N_{i-1})$, where N_i is the number of users whose dominant class of characteristics is i. These users may initially be found in different clusters for the simple reason that a user is linked to a community due to a relationship with current community members and he/she is surely unaware of the characteristics of all the other community members. It is known that a Markov proposes model is irreducible, that is, each state can be reached from any other state with non-zero probability. Therefore, there exists an equilibrium state probability distribution:

$$P(\mathbf{N}) = p_0^{N_0} p_1^{N_1} \cdots p_{i-1}^{N_{i-1}} \tag{3}$$

where p_i are the probabilities of each state. The equilibrium describes the fact that the probability of transitioning from a state $\mathbf{N}$ to any other state equals the probability of transitioning from any other state to $\mathbf{N}$. Also, the well-known

TABLE 1. Notations used in this paper.

Notation	Description
$G=(V,E)$	A weighted, undirected graph
$s(x,y)$	Similarity matrix between users x and y
i	the number of classes of characteristics stored in matrix s
C	Connectivity degree
c	Cluster index
$n(x,y)$	The number of all the user pairs (x,y) within a cluster
n	The number of users per cluster
$e_{\mathbf{x}}$	Number of internal links of a user x within a cluster c
$e'_{\mathbf{x}}$	Number of external links of a user x outside a cluster c
$\mathcal{B}_c(\mathbf{x})$	The belonging degree of a user x within a cluster
$\mathbf{i}_c$	Dominant class a cluster c

formula holds:

$$\sum_{\text{all N}} p(\mathbf{N}) = 1 \tag{4}$$

The following formula has been proven in [41]:

$$p(\mathbf{N}) = \frac{1}{F(N)} \prod_{k=0}^{n-1} \mathcal{C}_i^{N_k}, \tag{5}$$

where n is the number of users per cluster, $\mathcal{C}_i$ is the average of class i values within the cluster as computed by Eq. 1, and $F(N)$ is computed as follows:

$$F(N) = \sum_{\text{for all states}N} \prod_{k=0}^{n-1} \mathcal{C}_i^{N_k}, \tag{6}$$

The exponent terms N_k in Eq. 5 and Eq.6 are all the possible distributions of users within a cluster.

For clarity, let us return to the example of Fig.5. In this example, there are $i = 3$ classes. In cluster 0 the dominant class is $i = 2$ with an average value $\mathcal{C}_2 = 0.87$. In cluster 1, the dominant class is again $i = 2$ with an average value $\mathcal{C}_2 = \frac{1.0+1.0+0.9+1.0}{4} = 0.975$. Finally, in cluster 2, class $i = 0$ dominates, with an average value of $\frac{0.9+0.9+1.0+1.0}{4} = 0.95$. The average values that will be used in Eq. 5 are $\mathcal{C}_0 = 0.87$ for cluster 0, $\mathcal{C}_1 = 0.975$ for cluster 1, and $\mathcal{C}_2 = 0.95$ for cluster 2.

With the definitions given, the problem we try to solve can by defined as follows:

Given:

1) *An initial set of clusters which include a number of users with their characteristics defined.*
2) *An unknown number of users with different characteristics, which will enter the network,*

we use a probabilistic network to find the cluster to which each new user fits better so that the connectivity degree of each cluster and the probability of having more homogeneous classes increase. Also, we define the conditions under which we consider two or more clusters as overlapping, that is, as a new community.

Table 1 provides the notations used in this paper.

IV. OUR PROBABILISTIC COMMUNITY DETECTION STRATEGY

One of the basic ideas of our probabilistic approach is, at any time, to compute the probability of having *homogeneous* clusters, in the sense that they are *purely* composed of user with a common dominant class of characteristic $\mathbf{i}$. This probability, $p_{\mathbf{i}}$, is computed as follows:

$$p_{\mathbf{i}} = \frac{\prod_{\substack{k=0 \\ N_{\mathbf{i}}=n}}^{n-1} \mathcal{C}_{\mathbf{i}}^{N_k}}{\sum_{\substack{\text{for all N,} \\ N_{\mathbf{i}}\neq n}} \prod_{k=0}^{n-1} \mathcal{C}_i^{N_k}}, \tag{7}$$

where the numerator is the product of all the $\mathcal{C}_{\mathbf{i}}$ (average value of the dominant class values in a cluster c) for the case where $N_{\mathbf{i}} = N$ (all users have the same dominant class) and the denominator is the product $\mathcal{C}_i$ (average value of the non-dominant class values in cluster c) for all the possible user distribution within c, such that not all of the users have the same dominant class.Proposition 1 shows that $p_{\mathbf{i}}$ can be efficiently computed using a recursive formula.

Proposition 1: Regardless of the number of users the complexity of computing $p_{\mathbf{i}}$ is linear.

Proof: Let us assume that the dominant class is $i = 0$ (the proof is similar for any other dominant class if we mutually change the position of $\mathcal{C}_0$ with any of the other $\mathcal{C}_i$'s). The numerator computations are a simple product, as there is only one possible distribution such that $N_{\mathbf{i}} = n$, which is $(N, 0, 0, \dot{;}0)$. Thus, the product here is simply $\mathcal{C}_{\mathbf{i}}^n$ as all the other N_k values are zero so the terms produced are equal to 1. Therefore, we need to show how the denominator can be efficiently computed. Let us assume, with no loss of generality, that there are 3 classes of characteristics (the derivation that follows can be easily extended to more than 3, although the discrimination of user attributes to three classes is usually enough for experimental purposes). The sum of products in the denominator has the following form:

$$\begin{aligned}
&\mathcal{C}_{\mathbf{0}}^{n-1}\mathcal{C}_1^1\mathcal{C}_2^0 + \mathcal{C}_{\mathbf{0}}^{n-1}\mathcal{C}_1^0\mathcal{C}_2^1 \\
&\quad + \mathcal{C}_{\mathbf{0}}^{n-2}\mathcal{C}_1^2\mathcal{C}_0^0 + \mathcal{C}_{\mathbf{0}}^{n-2}\mathcal{C}_1^1\mathcal{C}_0^1 + +\mathcal{C}_{\mathbf{0}}^{n-2}\mathcal{C}_1^0\mathcal{C}_0^2 + \\
&\quad + \ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots + \\
&\quad + \mathcal{C}_{\mathbf{0}}^0\mathcal{C}_1^{n-1}\mathcal{C}_2^0 + \mathcal{C}_{\mathbf{0}}^0\mathcal{C}_1^0\mathcal{C}_2^{n-1}
\end{aligned} \tag{8}$$

Now, let us analyze these sums of products, starting from $n = 1$ and keep on increasing the number of users. We denote as $Y_1(n)$ the sum of products produced for the first two classes, $\mathbf{i} = 0$ (the dominant) and $i = 1$.

- for $n = 1$, we get $Y_1(1) = \mathcal{C}_0 + \mathcal{C}_1$
- for $n = 2$, we get

$$\begin{aligned}
Y_1(2) &= \mathcal{C}_0^2 + \mathcal{C}_0\mathcal{C}_1 + \mathcal{C}_1^2 \\
&= \mathcal{C}_0^2 + \mathcal{C}_1(\mathcal{C}_0 + \mathcal{C}_1) \\
&= \mathcal{C}_0^2 + \mathcal{C}_1 Y_1(1)
\end{aligned}$$

- for $n = 3$ we get

$$\begin{aligned}
Y_1(3) &= \mathcal{C}_0^3 + \mathcal{C}_0^2\mathcal{C}_1 + \mathcal{C}_0\mathcal{C}_1^2 + \mathcal{C}_1^3 \\
&= \mathcal{C}_0^3 + \mathcal{C}_1[\mathcal{C}_0^2 + \mathcal{C}_1\mathcal{C}_0 + \mathcal{C}_1^2]
\end{aligned}$$

$$= \mathcal{C}_0^3 + \mathcal{C}_1[\mathcal{C}_0^2 + \mathcal{C}_1(\mathcal{C}_0 + \mathcal{C}_1)]$$
$$= \mathcal{C}_0^3 + \mathcal{C}_1 Y_1(2)$$

By proceeding in a similar manner, we obtain the linear recursive formula:

$$Y_{i-1}(n) = \mathcal{C}_0^n + \mathcal{C}_1 Y_1(n-1) \tag{9}$$

Now, let us analyze $Y_2(n)$ the sum of products produced for all three classes, $\mathbf{i} = 0$ (the dominant) and $i = 1$, $i = 2$.

- for $n = 1$, we get $Y_2(1) = \mathcal{C}_0 + \mathcal{C}_1 + \mathcal{C}_2$
- for $n = 2$, we get

$$Y_2(2) = \mathcal{C}_0^2 + \mathcal{C}_0\mathcal{C}_1 + \mathcal{C}_0\mathcal{C}_2 + \mathcal{C}_1\mathcal{C}_2 + \mathcal{C}_1^2 + \mathcal{C}_2^2$$
$$= \mathcal{C}_2(\mathcal{C}_0 + \mathcal{C}_1 + \mathcal{C}_2) + \mathcal{C}_0^2 + \mathcal{C}_1(\mathcal{C}_0 + \mathcal{C}_1)$$
$$= \mathcal{C}_2 Y_2(1) + Y_1(2)$$

- for $n = 3$, we get

$$Y_2(3) = \mathcal{C}_0^3 + \mathcal{C}_0^2\mathcal{C}_1 + \mathcal{C}_0^2\mathcal{C}_2 + \mathcal{C}_0\mathcal{C}_1^2 + \mathcal{C}_0\mathcal{C}_1\mathcal{C}_2 + \mathcal{C}_0\mathcal{C}_2^2 + \mathcal{C}_1^3 + \mathcal{C}_2^3 + \mathcal{C}_1^2\mathcal{C}_2 + \mathcal{C}_1\mathcal{C}_2^2$$
$$= \mathcal{C}_2(\mathcal{C}_0^2 + \mathcal{C}_0\mathcal{C}_1 + \mathcal{C}_0\mathcal{C}_2 + \mathcal{C}_1\mathcal{C}_2 + \mathcal{C}_1^2 + \mathcal{C}_2^2) + \mathcal{C}_0^3 + \mathcal{C}_1(\mathcal{C}_0^2 + \mathcal{C}_1\mathcal{C}_0 + \mathcal{C}_1^2)$$
$$= \mathcal{C}_2 Y_2(2) + Y_1(3)$$

By proceeding in a similar manner, we obtain the linear recursive formula:

$$Y_i(n) = \mathcal{C}_2 Y_2(n-1) + Y_1(n) \tag{10}$$

From Equations 9 and 10, it is clear that the computations of $p_\mathbf{i}$ can be implemented in linear time using these recursion formulas. It can easily be seen that these formulas can be extended for more classes and users. ■

Proposition 2: The value of $p_\mathbf{i}$ increases when $\mathcal{C}_\mathbf{i}$ and decreases when the average value of the other characteristics $\mathcal{C}_i$ increase.

Proof: The proof is straightforward from Eq. 7.

Our community detection scheme is comprised of a series of steps, which are presented below:

Step 1: We start with a number of clusters, each having its own dominant class.

Step 2: We compute the average connectivity degree for each cluster $\mathcal{C}$.

Step 3: We import a set of n' new users within a cluster c with dominant class $\mathbf{i}$. We use Eq. 7 to estimate the probability of having a purely homogeneous cluster if these n' users connect in cluster c. These new users may change the average $\mathcal{C}_\mathbf{i}$ and $\mathcal{C}_i$ values which are used in Eq. 7. Then, we apply the following rules:

`Rule 1 - Best or Perfect Fit:` If a new user $\mathbf{x}$ is added into a cluster c via a number of $e_\mathbf{x}$ connections of the form $(\mathbf{x}, y)$, such that:

$$\begin{cases} z_\mathbf{i} = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x}, y)}{e_\mathbf{x}} \right] > \mathcal{C}_\mathbf{i}, \text{ for class } \mathbf{i} \\ z_i = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x}, y)}{e_\mathbf{x}} \right] < \mathcal{C}_i, \text{ for all classes } \neq \mathbf{i}. \end{cases} \tag{11}$$

where $\mathbf{i}$ refers to the dominant class and i refers to all the other classes within c, then a perfect or best fit incurs.

Proof: A perfect fit indicates that this new user fits perfectly in cluster c. Suppose that $z_\mathbf{i} > \mathcal{C}_\mathbf{i}$ and $z_\mathbf{i} < \mathcal{C}_\mathbf{i}$. Then, to re-compute $p_\mathbf{i}$ using Eq.7, we will embed $z_\mathbf{i}$ to $\mathcal{C}_i$ (the numerator in Eq. 7). Also we will embed the other z_i values to the products of the form $C_i^{N_k}$ (the denominator in Eq. 7). Thus, after embedding the new values $z_\mathbf{i}$ and z_i into Eq. 7, it is rewritten as follows:

$$p_\mathbf{i} = \frac{\prod\limits_{\substack{k=0 \\ N_\mathbf{i}=n}}^{n-1} (\mathcal{C}_\mathbf{i} z_\mathbf{i})^{N_k}}{\sum\limits_{\substack{\text{for all N,} \\ N_\mathbf{i} \neq n}} \prod\limits_{k=0}^{n-1} \mathcal{C}_i z_i^{N_k}} > \frac{\prod\limits_{\substack{k=0 \\ N_\mathbf{i}=n}}^{n-1} \mathcal{C}_\mathbf{i}^{N_k}}{\sum\limits_{\substack{\text{for all N,} \\ N_\mathbf{i} \neq n}} \prod\limits_{k=0}^{n-1} \mathcal{C}_i^{N_k}} \tag{12}$$

Therefore, the probability $p_\mathbf{i}$ of cluster c to be homogeneous increases. ■

`Overlapping Issues Under Best Fit`

To consider overlapping, we need to examine the belonging degree of the new user $\mathbf{x}$. We set:

$$z'_\mathbf{i} = \frac{\sum s_\mathbf{i}(\mathbf{x}, y)/e_\mathbf{x}}{\sum\limits_{\text{all } (x,y) \text{ outside c}} s_\mathbf{i}(\mathbf{x}, y)/e'_\mathbf{x}} \text{ and } z'_i = \frac{\sum s_i(\mathbf{x}, y)/e_\mathbf{x}}{\sum\limits_{\text{all } (x,y) \text{ outside c}} s_i(\mathbf{x}, y)/e'_\mathbf{x}}.$$

In other words, $z'_\mathbf{i}$ is the belonging degree of $\mathbf{x}$ in c regarding the dominant class of characteristics of c and z'_i is the belonging degree of $\mathbf{x}$ in c regarding its non-dominant classes. We examine the following cases:

(1) If $z'_\mathbf{i} < 1$, it follows that there are connections of $\mathbf{x}$ in other clusters (let us symbolize them using the notation c'), outside c, with which $\mathbf{x}$ has larger similarity values regarding the dominant class, compared to its similarities within c. These connections should also be considered as members of c, thus c overlaps with clusters c'. By including these overlaps, we increase the updated probability of $p_\mathbf{i}$ of c.

(2) If $z'_\mathbf{i} > 1$, there are no overlaps. The connections of $\mathbf{x}$ within c are stronger than those outside c, regarding the dominant class.

(3) If $z'_i < 1$ for one or more classes other than the dominant $\mathbf{i}$, then:

(3.i) $z'i \geq \mathcal{C}_{c'}$ for class i, which is dominant for c', then $\mathbf{x}$ becomes member of c' as well, thus overlapping occurs. The addition of x to c' will increase $p_\mathbf{i}$ for community c'.

(3.ii) $z'i < \mathcal{C}_{c'}$, then, no overlapping occurs. The addition of x to c' does not add anything to $p_\mathbf{i}$ of c'.

`Rule 2 - Good Fit:` If a new user $\mathbf{x}$ is added into a cluster c via a number of $e_\mathbf{x}$ connections of the form $(\mathbf{x}, y)$,

such that:

$$\begin{cases} z_{\mathbf{i}} = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x},y)}{e_{\mathbf{x}}} \right] > \mathcal{C}_{\mathbf{i}}, \text{ for class } \mathbf{i} \\ z_i = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x},y)}{e_{\mathbf{x}}} \right] \geq \mathcal{C}_i, \text{ for some class } j \neq \mathbf{i}. \end{cases} \tag{13}$$

where, as in Rule 1, $\mathbf{i}$ refers to the dominant class and i refers to all the other classes within c, then a good fit incurs.

Proof: A good fit indicates that thin new user fits well in cluster c, but one should consider if it fits better to another cluster, c'. To see this fact, suppose that we re-compute $p_{\mathbf{i}}$ using Eq.7. We embed $z_{\mathbf{i}}$ to $\mathcal{C}_i$ (the numerator in Eq. 7). Also we will embed the other z_i values to the products of the form $C_i^{N_k}$ (the denominator in Eq. 7). Thus, after embedding, the denominator part of Eq. 7 will be:

$$\sum_{\substack{\text{for all N,} \\ N_{\mathbf{i}} \neq n,}} \prod_{k=0}^{n-1} \mathcal{C}_i \mathbf{A} z_i^{N_k}$$

where:

$$\begin{cases} \mathbf{A} = 1 \text{ for classes } i \neq j \\ \mathbf{A} > 1 \text{ for class } i = j \end{cases}$$

It follows, that, all the product factors produced for $i = j$, $\mathcal{C}_j^{N_k}$ will be larger than the ones computed in the initial computation of $p_{\mathbf{i}}$. In other words, of $z_j > z_{\mathbf{i}}$ (that is, for $i = j$), then the new value of $p_{\mathbf{i}}$ will be reduced. ■

Overlapping Issues Under a Good Fit In a scenario of good fit, we either consider transitioning the new user to community c' or overlap the two communities. Three cases are examined:

(1) If j is the dominant class of c' and the belonging degree of $\mathbf{x}$ in c' as far as class j is concerned is larger compared to the current connectivity degree of c' regarding j, that is

$$\mathcal{C}(c'_j) = \sum_{\substack{\text{for } j \\ \text{all pairs } (x,y) \text{ in } c'}} \frac{s_i(x,y)}{n(x,y)} \tag{14}$$

and

$$\sum_{\text{all pairs in c'}(\mathbf{x},y)} \frac{s_i(\mathbf{x},y)}{e_{\mathbf{x}}} \geq \mathcal{C}(c')_j \tag{15}$$

it follows that the $\mathbf{x}$ should be considered as member of both c and c' (overlap) as this will increase the average value of $p_\rangle$ $p_|$ for c and c' respectively. In other words, including $\mathbf{x}$ to both c, c' will increase their probabilities to be homogeneous.

(2) If j is the dominant class of c' and the belonging degree of $\mathbf{x}$ in c' as far as class j is concerned is smaller compared to the current connectivity degree of c' regarding j (Eq. 15 is reversed), then it follows that $\mathbf{x}$ shoukd remain in c.

(3) If j is not the dominant class of c', we simply keep $\mathbf{c}$ in c.

Rule 3 - Bad Fit: If a new user $\mathbf{x}$ is added into a cluster c via a number of $e_{\mathbf{x}}$ connections of the form $(\mathbf{x}, y)$, such that:

$$\begin{cases} z_{\mathbf{i}} = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x},y)}{e_{\mathbf{x}}} \right] \leq \mathcal{C}_{\mathbf{i}}, \text{ for class } \mathbf{i} \\ z_i = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x},y)}{e_{\mathbf{x}}} \right] > \mathcal{C}_i, \text{ for some class } j \neq \mathbf{i}. \end{cases} \tag{16}$$

where $\mathbf{i}$ refers to the dominant class and i refers to all the other classes within c, then a bad fit incurs.

Proof: A bad fit indicates that the new user does not fits well in cluster c, but it can well fit to another cluster c'. Suppose that $z_{\mathbf{i}} < \mathcal{C}_{\mathbf{i}}$ and $z_{\mathbf{i}} > \mathcal{C}_{\mathbf{i}}$. Then, if we try to re-compute $p_{\mathbf{i}}$ using Eq.7, we know that the numerator will become

$$\prod_{\substack{k=0 \\ N_{\mathbf{i}}=n}}^{n-1} (\mathcal{C}_{\mathbf{i}} z_{\mathbf{i}})^{N_k} < \prod_{\substack{k=0 \\ N_{\mathbf{i}}=n}}^{n-1} \mathcal{C}_{\mathbf{i}}^{N_k} \tag{17}$$

because the average values of the terms $z_{\mathbf{i}}\mathcal{C}_{\mathbf{i}}$ will be less than the ones of the $\mathcal{C}_i$ terms. On the other hand, as in Rule 2, the denominator terms will increase. As a result, the probability $p_{\mathbf{i}}$ is reduced and therefore the new user $\mathbf{x}$ does not fit well in c. ■

Overlapping/Transitioning Issues Under a Bad Fit In the case of a bad fit, our overlapping or transitioning decision depends on the values of the non-dominant classes of c :

(1) If class j for which the second part of Eq.16 holds is dominant for a cluster c' and Eq. 15 do hold, then the new user transitions to c'. In this case, it increases its probability to be homogeneous. Moreover, this transition recovers the last (and larger) $p_{\mathbf{i}}$ value for cluster c.

(2) If class j is dominant in c' and Eq. 15 does not hold or if j is not dominant in c' the new user is transitioned to a new cluster, with dominant characteristic the higher value within the belonging degree matrix of $\mathbf{x}$. This new cluster overlaps with the connections of $\mathbf{x}$ and grows under the rules presented in this paragraph, as more users are added.

Rule 4 - Worst Fit: If a new user $\mathbf{x}$ is added into a cluster c via a number of $e_{\mathbf{x}}$ connections of the form $(\mathbf{x}, y)$, such that:

$$\begin{cases} z_{\mathbf{i}} = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x},y)}{e_{\mathbf{x}}} \right] < \mathcal{C}_{\mathbf{i}}, \text{ for class } \mathbf{i} \\ z_i = \left[\sum\limits_{\text{all pairs } (\mathbf{x},y)} \dfrac{s_i(\mathbf{x},y)}{e_{\mathbf{x}}} \right] < \mathcal{C}_i, \text{ for some class } j \neq \mathbf{i}. \end{cases} \tag{18}$$

where, as in the other rules, $\mathbf{i}$ refers to the dominant class and i refers to all the other classes within c, then a worst fit incurs.

In the worst fit, the new user does not fit well in any of the cluster.

Overlapping/Transitioning Issues Under a Bad Fit Here, we act as in the second case of overlapping/transitioning issues discussed under the bad fit case. We prefer to create a new cluster including **x**. As before, this new cluster overlaps with the clusters where the connections of **x** belong and, as users are being added on, it grows using the rules presented in this paragraph.

A. COMPUTATIONAL COMPLEXITY

To compute the overall complexity, we have to consider the computations of the clusters connectivity degree (Eq. 1), the belonging degree of each user within the cluster (Eq.2), the probabilistic computations of $p_{\mathbf{i}}$. These computations are required for the application of our rules. For the probabilistic computation of Eq. 7, we have already proved that we can implement numerous of computations within the cluster in linear time. The connectivity degree values depend on $n(x, y)$ for the first time the cluster is generated and at most $n(x, y)\cdot m$, where m is the number of new entries within each cluster. Thus, the overall cluster update requires $I \cdot n(x, y) \cdot m$ for a total of I clusters, which is linear.

V. SIMULATION RESULTS AND DISCUSSION

In this section, we verify the functionality of the community detection algorithm which is used for data distribution across the network. To verify the community detection scheme, we carried out a series of simulations implemented using a Java simulator and we compare it to well-known algorithms in the literature. To implement our simulations, we used the well-known benchmark provided by Lancichinetti and Fortunato [42]).

The following parameters are required by the benchmark:

- The degree of each node, which is taken by a power law distribution with exponent γ, with k_{min} and k_{max} being the extreme values of the distribution chosen in such a way, that the average degree is k. Typically, $2 \leq \gamma \leq 3$.
- The mixing parameter, which is the most important parameter for the implementation of our scheme. The mixing parameter is the average ratio of external degree/total degree for each node, which is denoted by μ. Specifically, each node shares $1 - \mu$ of its links with the members of its community and μ with members of other communities.
- The exponent for the community size power law distribution β. Typically, $1 \leq \beta \leq 2$. The community sizes should sum to n, the number of nodes of the cluster.

For our experiments, we set the above parameters as shown in Table 2. We set the mixing parameter to $\mu = 0.8$ and $\gamma = 2$ and based on the computations defined in [42], we have obtained k_{min} and k_{max}.

To have fair comparisons (in terms of network sizes) with other probabilistic schemes found in the literature, we used two real-world datasets: ego-Facebook, and ego-Twitter.

TABLE 2. Simulation parameters.

Parameter	Value
Number of nodes N	up to 4000
Mixing parameter μ	0.1-0.8
Node degree power law distribution exponent γ	2
Minimum node degree, k_{min}	8
Maximum node degree, k_{max}	24
Average node degree, k	16
Community size power law distribution exponent β	1 to 2

Larger real world datasets are also available, but they can be used for larger networks. Examples are: Pokec, LiveJournal or Google+. The statistical properties of these networks are given in Table 3.

For each of the two networks, we used a combination of characteristics which were organized in 3 classes. The original data can be found in [43] and since they were not in a proper form for the purpose of our algorithm, so we had to make some kind of transformations, so that the data was suitable for our implementation. Here, we describe the transformations we made for the ego-Facebook network. For Twitter, we worked similarly.

For each user examined (called *ego*), a set characteristics has been created. In [43], one can find data for a total of 26 attribute classes, including hometowns, education, birthdays, political affiliations, schools. For our simulations, we used three of these classes, namely education, occupation and demographic characteristics. For our purposes, 3 classes suffice to evaluate our work, however one can use more classes without adding a lot of computational effort. In Table 4 one can see 5 of the characteristics for class education. Note that the characteristics are encoded as integer values, to maintain users' privacy. All the characteristics in [43] are accompanied by an ID.

Each ego forms circles (clusters according to our terminology) with a set of nodes, based on the attribute values. These circles will be used as the initial communities for our algorithm. Based on the data given in [43], circle_13 is formed by the node ID's (or users) 138, 131, 68, 143, 86. The binary values at positions 0,21,30,53,72, and 100 of their bit_vectors are given in Table 5.

To find the similarity matrix between each pair of users x and y, for a given class we simply perform a bitwise XNOR between the values within the class and we divide the number of 1s produced as a result of the XNOR operation by the number of characteristic. For example, consider users 68 and 143 of Table 5. The bitwise XNOR gives 5 1s. Thus, the similarity of the two users regarding class "Education" is 5/6.

To verify the functionality of our community detection scheme, we have conducted a series of experiments and compared our scheme with two recent probabilistic community detection schemes: [36] and [38].

In the first set of simulations, we have compared the Normalized Mutual Information (NMI) of our scheme against the FPPM scheme [36]. This scheme is based on the first

TABLE 3. Statistical properties of the datasets for ego-Facebook and ego-Twitter.

Network	Num. of Nodes	Num. of Edges (longest shortest path)	Diameter	Average local clustering coefficient (*alcc*)
Facebook	4,039	88,234	8	0.6055
Twitter	81,306	1,768,149	7	0.5653

TABLE 4. Some characteristics for class "Education".

Attribute ID	Combination	Attribute Description
0	*birthday;anonymized feature 0*	A birthday value
21	*education;degree;id;anonymized feature 21*	Education and degree (For example, University, Informatics)
30	*education;school;id;anonymized feature 30*	Education and school user graduated from
53	*education;type;anonymized feature 53*	Education and type (for example High education and college)
72	*education;year;id;anonymized feature 72*	Education and year or period of years
100	*languages;id;anonymized feature 100*	Languages spoken

TABLE 5. Bit_vectors for the class "Education" for the ego_13 network.

Node ID	Bit_vector values
EGO=0	[0 0 0 1 0 0]
138	[0 0 0 0 0 0]
131	[0 0 0 0 0 0]
68	[0 0 0 0 0 0]
143	[0 0 0 1 0 0]
86	[0 0 0 0 0 0]

passage probabilities and uses the diameter of the network as the maximal step length of random walks which can be adaptively appropriate to different networks. The similarity measure used is based on the average correlation between first passage probabilities at multiple times. This measure considers the overall structural information, not just one step of fixed length. Hierarchical clustering is used to group the nodes into communities and a post-processing procedure removes very small communities to improve accuracy.

The NMI compares the similarity between parts of different network, thus, when the ground truth is known, it is a good indicator of the similarity between the ground truth and the partition delivered by probabilistic schemes algorithms. Reference [36] and [38] The NMI computations involve computing the similarity (mutual information) between parts of different networks and it is a good indicator of the robustness and homogeneousness of a clustering method. The larger the value of the NMI, the better the results are. The results shown in Figures 5 and 5 are mean values taken from 10 simulations. The results indicate that both strategies perform well in terms of NMI, for various cluster sizes (250-1000). The performance is improved when the γ value becomes 3 (a higher node degree usually indicates higher similarities, especially between neighboring nodes). The performance degrades as we increase the mixing parameter (more external than internal links). However, our strategy, as described in the previous section, takes into account the importance of each user in the cluster structure and its purity. Also, instead of completely removing small communities (as in [36]), our scheme regularly transitions nodes that have bad or worst fit, thus it guarantees that the NMI values are high for different parts within a cluster (even for small ones) and between clusters with the same dominant class of characteristics. The results indicate that, although the performance degrades as μ increases, this is done in a smooth fashion compared to FPPM. Also, we have included the NMI values of the very well-known Louvain approach [44], to show the deviation of our approach to the ground truth. Apparently, the NMI values of the Louvain scheme approach 1 (very close to the ground truth), so this approach can well indicate how well a probabilistic method can approach the ground truth.

In the second set of simulations, we evaluate the running time of our scheme against the running time of another recent probabilistic strategy for comminity detection introduced by Tian et al. [38]. This work uses unified Bayesian generative model to detect generalized communities. Also, it provides semantic descriptions using a combination of network topologies and node attributes. The proposed scheme has two parts. First, it applies a mixture model to describe network regularities and then it adjusts the classic Latent Dirichlet Allocation (LDA) topic model to define community semantics.

The comparison results are shown in Figure 5. In Fig. 5(a), we used the relatively smaller ego-Facebook network with smaller number of nodes per cluster. In Fig. 5(b), we have used the ego-Twitter network, which is larger, with more dense clusters. In both scenarios, our scheme executes faster and its running time increases smoothly as more users added in the clusters, because of the linearity of the computations involved. On the contrary, the two-part model in [38] involves more complex computations and this is affected in the execution times.

Finally, we tested the accuracy of our work by comparing the total number of communities (including overlaps) that our probabilistic scheme detects against our previous effort which is based on threaded binary trees [45]. The results indicate that the probabilistic scheme detects about 15% fewer communities for the ego-Facebook network and about 25% fewer communities compared to the ego-Twitter network. This is because the threaded binary tree produces more overlaps in a more loose way (simply when "stronger" paths are found within "weaker" ones, based on the link weights). Instead, the probabilistic strategy requires specific conditions to hold

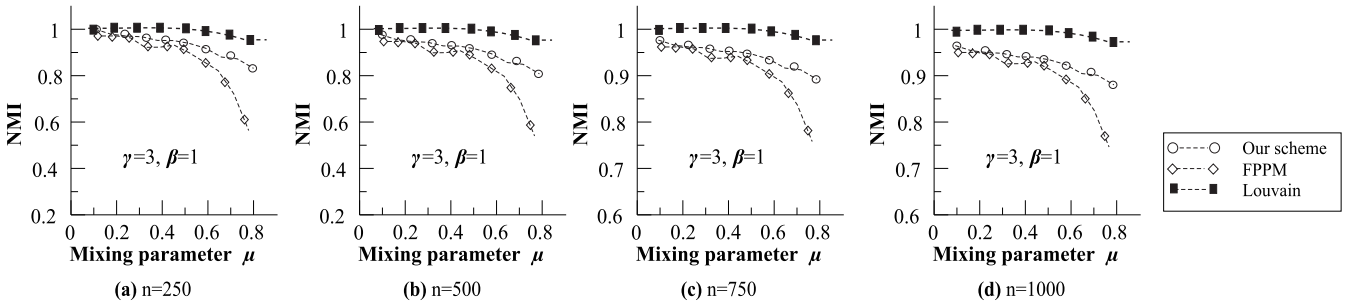

FIGURE 2. NMI results for $\beta = 1, \gamma = 3, n = 250 - 1000$.

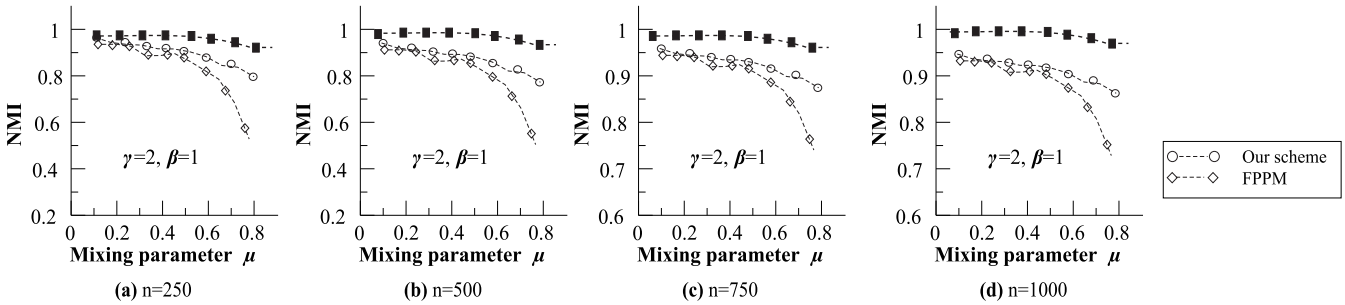

FIGURE 3. NMI results for $\beta = 1, \gamma = 2, n = 250 - 1000$.

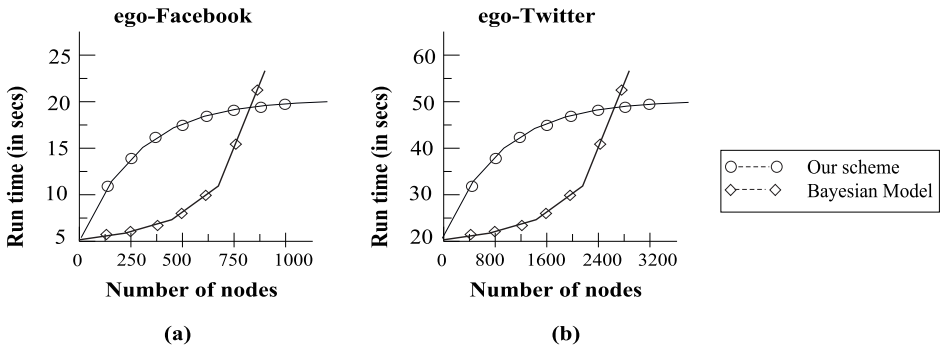

FIGURE 4. Execution times for the ego-Facebook and ego-Twitter networks.

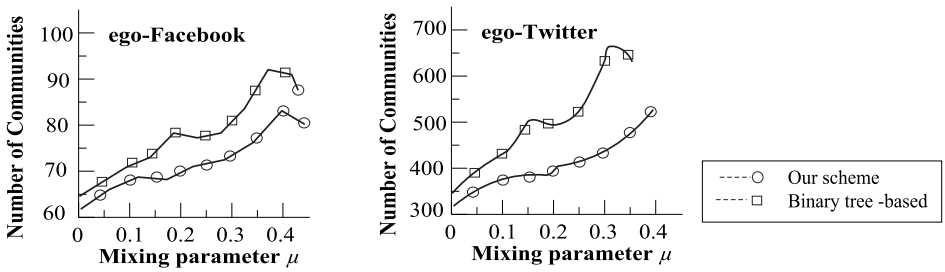

FIGURE 5. Number of detected communities for the ego-Facebook and ego-Twitter networks.

for each of the four possible fits(see the rules described in the previous section).

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a probabilistic community detection scheme. The clusters are represented as a network of changing states, where each state represents the number of users in the cluster with a specific class of characteristics. Our scheme computes the probability of a class being homogeneous based on its current state. Using this probability and the statistic metrics that describe each cluster, we determine each new user's fit within a cluster. The main advantages of

our scheme is the linearity of the computations involved and that it can be used independently of the cluster topology. For our simulations, we used data from the ego-Facebook and ego-Twitter networks. The experimental results have shown that our strategy detects robust and homogeneous clusters and its accuracy (regarding the number of clusters detected) is satisfactory compared to other schemes. Also, it executes faster than comparable schemes.

In the future, we will try to organize our strategy in such a way that it facilitates parallelism using both parallel CPU and GPU devices. Also, we plan to perform extensive experiments on more and even larger networks, using similar system configurations. In this regard, we need to make proper changes to allow a trade-off between certain parameters, in order to target larger communities. Finally, we need to consider the memory requirements for storage, especially in case where GPU processing is involved.

REFERENCES

[1] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2009.
[2] Z. Bu, C. Zhang, Z. Xia, and J. Wang, "A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network," *Knowl.-Based Syst.*, vol. 50, pp. 246–259, Sep. 2013.
[3] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 76, no. 3, 2007, Art. no. 036106.
[4] C. L. Staudt and H. Meyerhenke, "Engineering parallel algorithms for community detection in massive networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 171–184, Jan. 2016.
[5] A. Mahmood and M. Small, "Subspace based network community detection using sparse linear coding," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 801–812, Mar. 2016.
[6] A. Prat-Pérez, D. Dominguez-Sal, J. M. Brunat, and J.-L. Larriba-Pey, "Shaping communities out of triangles," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2012, pp. 1677–1681.
[7] F. Zhao and A. K. H. Tung, "Large scale cohesive subgraphs discovery for social network visual analysis," *Proc. VLDB Endowment*, vol. 6, no. 2, pp. 85–96, Dec. 2012.
[8] D. Chen, M. Shang, Z. Lv, and Y. Fu, "Detecting overlapping communities of weighted networks via a local algorithm," *Phys. A, Stat. Mech. Appl.*, vol. 389, no. 19, pp. 4177–4187, Oct. 2010.
[9] Z. Lu, X. Sun, Y. Wen, G. Cao, and T. La Porta, "Algorithms and applications for community detection in weighted networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 11, pp. 2916–2926, Nov. 2015.
[10] S. Qiao, N. Han, Y. Gao, R.-H. Li, J. Huang, J. Guo, L. A. Gutierrez, and X. Wu, "A fast parallel community discovery model on complex networks through approximate optimization," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1638–1651, Sep. 2018.
[11] M. C. V. Nascimento and L. Pitsoulis, "Community detection by modularity maximization using GRASP with path relinking," *Comput. Oper. Res.*, vol. 40, no. 12, pp. 3121–3131, Dec. 2013.
[12] D. Džamić, D. Aloise, and N. Mladenović, "Ascent–descent variable neighborhood decomposition search for community detection by modularity maximization," *Ann. Oper. Res.*, vol. 272, nos. 1–2, pp. 273–287, Jan. 2019.
[13] R. Santiago and L. C. Lamb, "Efficient modularity density heuristics for large graphs," *Eur. J. Oper. Res.*, vol. 258, no. 3, pp. 844–865, May 2017.
[14] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, Jun. 2006.
[15] S. Gregory, "Finding overlapping communities in networks by label propagation," *New J. Phys.*, vol. 12, no. 10, Oct. 2010, Art. no. 103018.
[16] C. Lyu, Y. Shi, and L. Sun, "A novel local community detection method using evolutionary computation," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3348–3360, Jun. 2021.
[17] I. Farkas, D. Ábel, G. Palla, and T. Vicsek, "Weighted network modules," *New J. Phys.*, vol. 9, no. 6, p. 180, Jun. 2007.
[18] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki, "Sequential algorithm for fast clique percolation," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 78, no. 2, Aug. 2008, Art. no. 026109.
[19] X. Zhang, C. Wang, Y. Su, L. Pan, and H.-F. Zhang, "A fast overlapping community detection algorithm based on weak cliques for large-scale networks," *IEEE Trans. Computat. Social Syst.*, vol. 4, no. 4, pp. 218–230, Dec. 2017.
[20] M. Li, S. Lu, L. Zhang, Y. Zhang, and B. Zhang, "A community detection method for social network based on community embedding," *IEEE Trans. Computat. Social Syst.*, vol. 8, no. 2, pp. 308–318, Apr. 2021.
[21] A. Sarkar, N. Mehta, and P. Rai, "Graph representation learning via ladder gamma variational autoencoders," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 5604–5611.
[22] H. Shi, H. Fan, and J. T. Kwok, "Effective decoding in graph auto-encoder using triadic closure," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 906–913.
[23] L. Wu, Q. Zhang, C.-H. Chen, K. Guo, and D. Wang, "Deep learning techniques for community detection in social networks," *IEEE Access*, vol. 8, pp. 96016–96026, 2020.
[24] J. Cao, D. Jin, L. Yang, and J. Dang, "Deep learning techniques for community detection in social networks," *Neurocomputing*, vol. 297, no. 5, pp. 71–81, 2018.
[25] J. J. Choong, X. Liu, and T. Murata, "Optimizing variational graph autoencoder for community detection with dual optimization," *Entropy*, vol. 22, no. 2, pp. 1–21, 2020.
[26] J. Choong, X. Liu, and T. Murata, "Learning community structure with variational autoencoder," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2018, pp. 69–78.
[27] V. Bhatia and R. Rani, "A distributed overlapping community detection model for large graphs using autoencoder," *Future Gener. Comput. Syst.*, vol. 94, pp. 16–26, May 2019.
[28] M. Shao, S. Li, Z. Ding, and Y. Fu, "Deep linear coding for fast graph clustering," in *Proc. 24th Int. Conf. Artif. Intell.*, 2015, pp. 3798–3804.
[29] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th Int. Conf. Artif. Intell.*, 2014, pp. 1293–1299.
[30] D. He, Y. Song, D. Jin, Z. Feng, B. Zhang, Z. Yu, and W. Zhang, "Community-centric graph convolutional network for unsupervised community detection," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3515–3521.
[31] G. Sperli, "A deep learning based community detection approach," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, 2019, pp. 1107–1110.
[32] S. G. O. Shchur, "Overlapping community detection with graph neural networks," in *Proc. 1st Int. Workshop Deep Learn. Graphs*, 2019, pp. 1–7.
[33] Y. Jia, Q. Zhang, W. Zhang, and X. Wang, "CommunityGAN: Community detection with generative adversarial nets," in *Proc. World Wide Web Conf.*, May 2019, pp. 784–794.
[34] S. Souravlas, S. Anastasiadou, and S. Katsavounis, "A survey on the recent advances of deep community detection," *Appl. Sci.*, vol. 11, no. 16, p. 7179, Aug. 2021.
[35] H. Zare, M. Hajiabadi, and M. Jalili, "Detection of community structures in networks with nodal features based on generative probabilistic approach," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 7, pp. 2863–2874, Jul. 2021.
[36] Z. Wu, X. Wang, W. Fang, L. Liu, S. Tang, H. Zheng, and Z. Zheng, "Community detection based on first passage probabilities," *Phys. Lett. A*, vol. 390, Feb. 2021, Art. no. 127099.
[37] A. Singh, S. Garg, S. Batra, and N. Kumar, "Probabilistic data structure-based community detection and storage scheme in online social networks," *Future Gener. Comput. Syst.*, vol. 94, pp. 173–184, May 2019.
[38] Q. Tian, W. Wang, Y. Xie, H. Wu, P. Jiao, and L. Pan, "A unified Bayesian model for generalized community detection in attribute networks," *Complexity*, vol. 2020, pp. 1–15, Aug. 2020.
[39] J. Geng, A. Bhattacharya, and D. Pati, "Probabilistic community detection with unknown number of communities," *J. Amer. Stat. Assoc.*, vol. 114, no. 526, pp. 893–905, Apr. 2019.
[40] S. Bahadori, H. Zare, and P. Moradi, "PODCD: Probabilistic overlapping dynamic community detection," *Expert Syst. Appl.*, vol. 174, Jul. 2021, Art. no. 114650.
[41] W. J. Gordon and G. F. Newell, "Closed queuing systems with exponential servers," *Oper. Res.*, vol. 15, no. 2, pp. 254–256, 1967.

[42] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 80, no. 1, 2009, Art. no. 016118.
[43] J. Leskovec and A. Krevl. (Jun. 2014). *SNAP Datasets: Stanford Large Network Dataset Collection*. University of Stanford. [Online]. Available: http://snap.stanford.edu/data
[44] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.
[45] S. Souravlas, A. Sifaleras, and S. Katsavounis, "Hybrid CPU-GPU community detection in weighted networks," *IEEE Access*, vol. 8, pp. 57527–57551, 2020.

STAVROS SOURAVLAS (Member, IEEE) is currently an Assistant Professor of computer architecture and digital logic design with the Department of Applied Informatics, School of Information Sciences, University of Macedonia, where he joined, in 2014. His research interests include computer architecture and performance evaluation, parallel and distributed systems, big data stream scheduling, cloud computing, and systems modeling and simulation.

SOFIA D. ANASTASIADOU is currently a Professor of statistics and research methodology, iostatsitics, and biomedicine research with the Department of Midwafery, School of Health Sciences, University of Western Macedonia. Her research interests include qualitative and quantitative methods in social sciences, applied statistics, implicative statistic analysis, multivariate statistical analysis, biostatistics, meta-analysis, structural equation models, data analysis, and big data.

THEODORE ECONOMIDES (Student Member, IEEE) is currently pursuing the degree in applied informatics with the University of Macedonia, Greece. He has studied for one semester abroad with the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. His research interests include software engineering, computer architecture and parallel processing, and social networking computer networks.

STEFANOS KATSAVOUNIS is currently an Associate Professor with the Department of Production Engineering and Management, Democritus University of Thrace, Greece. His research interests include scheduling, RCMPSP, project management, graph theory and modeling, heuristics for NP-hard problems in social networks, transportation and supply chain management, grey analysis, and data processing in material science.