Received October 2, 2018, accepted October 28, 2018, date of publication November 9, 2018,
date of current version December 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2880157

A Novel, Interdisciplinary, Approach for Community Detection Based on Remote File Requests

STAVROS SOURAVLAS[1], (Member, IEEE), ANGELO SIFALERAS[1], AND STEFANOS KATSAVOUNIS[2]
[1]Department of Applied Informatics, University of Macedonia, 54636 Thessaloniki, Greece
[2]Department of Production and Management Engineering, Democritus University of Thrace, 67100 Xanthi, Greece

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

This work was supported by the Unified Insurance Fund of Independently Employed (ETAA), Greece, through the Algorithms and Applications in Social Networks and Big Data Systems Project.

ABSTRACT Community structures are formed in many real-world networks, e.g., biological or medical groups, student groups, and so on. Communities are perhaps the most important feature of today's networks, since the majority of people who join a network also tend to join one or more communities. Therefore, several researchers find that the detection of hidden communities is a very interesting and challenging research field. Communities are represented as the groups of nodes on a graph, corresponding to users with similar interests. This paper introduces a novel, interdisciplinary, approach for community detection, combining social networks and distributed systems, where remote access to shared files is offered in a networked environment. A new metric, based on data requests, is introduced and used as a measure of the belonging degree of a node in a certain formed community. Two sets of simulations are used to verify our scheme: simulation results on synthetic networks and results derived from real data.

INDEX TERMS Social networks, community detection, distributed systems, data replication.

I. INTRODUCTION

Groups of users organize modern social networks [1], join a network, create their own profiles, requesting data and finding other users with the same interests. Smaller or larger groups within networks, formed in this way, are referred to as *communities*. Since networks are normally represented as graphs, an acceptable, though not universal, definition of a community describes it as a set of nodes and links in a network, such that its internal connections are more than its external connections [2]. The nodes, representing the users, of a community are considered similar to each other and dissimilar to the other nodes of the network [3]. The edges represent the connection between the users of one community or between users of different communities.

A. COMMUNITY DETECTION METHODS

aim to group the network nodes based on the relationships that hold among them, in order to form strongly linked subgraphs from the entire graph that represents the whole network [4]. Apparently, the community detection has turned out to be a graph problem and thus, graph-based methods have been developed to effectively solve the problem. In the remainder of this section, we categorize the community detection schemes found in the literature, discussing briefly the most representative strategies from each category.

1) CLIQUE PERCOLATION

The Clique Percolation Method (CPM) assumes that, a community consists of fully connected subgraphs that may, however, overlap. The community detection is based on searching and identifying neighboring cliques. The method finds initially all cliques in the network, consequently representing them in the graph by a vertex. If two cliques share a predefined number of members then, their corresponding vertices are connected. Thus, connected vertices on the graph represent network communities [5]–[8].

2) LINE GRAPH AND LINK PARTITIONING

This group of algorithms partitions links instead of nodes to detect a community. A node is assumed to be overlapped if the links connected to this node are found in more than one community. Generally, the nodes gain a community membership based on their *belonging degrees* (a measure based on the coefficients of links), [9]–[11]. First, these degrees are computed and then they are compared against a threshold. A node belongs to a community if its belonging degree overcomes this threshold. The schemes proposed in this category use a bottom-up approach in the sense that, they start from separately formed communities and try to expand them (overlap) to the entire network.

3) OPTIMIZATION

These approaches use metrics that characterize the quality of an interconnected node group. The community is considered as a cluster of vertices identified by the maximization of its modularity. This measure is based on the total internal and external degrees of the nodes of a group (or module). The aim of this optimization problem is to find a subgraph starting from a specified node such that, the inclusion of a new node, or the elimination of one node from the subgraph would lower the fitness value [5], [12]–[15].

4) LABEL PROPAGATION

In this class of algorithms, each node is initialized with a unique label and at every step, each node adopts the label that most of its neighbors currently have [3]. This is done by propagating these labels between neighboring vertices so that, members of a community are aware of their community membership. Moreover, the labels are able to include information about more than one community thus, each vertex can overlap, i.e., they can belong to more communities [16].

Taha [17] proposed a hybrid system called DCD_RAM that detects disjoint communities. It is partially based, on a number of techniques following different approaches that, work well only with certain topologies. The limitations posed by these approaches are overcome by measuring the influence of pairs of vertices over the flow of information in the entire network, characterizing the overall influence of each vertex in the network, discovering natural divisions of a network, and discovering disjoint communities with a new belonging formula. The results showed that, this method performs better than the strategies it is based on.

Community detection schemes that rely on information obtained from the distributed environment were proposed by Bu *et al.* [18] and Zhi-Xiao *et al.* [19]. Specifically, Bu *et al.* [18] proposed a method to detect communities based in the distributed environment. In particular, their method, called AOCCM, uses nodes that act as agents to pick the neighboring node with the largest structural similarity as the candidate node, and thus determine whether this candidate should be added into local community based on the modularity gain. In this sense, each agent interacts with the local environment to spot the local community and this method is further expanded to locate global communities. Another strategy that uses the distributed topology was proposed by Zhi-Xiao *et al.* [19]. Specifically, they proposed a new overlapping community detection method based on node location analysis. In the proposed method, they initially evaluate the node mass and the community structure is determined based on the node positions in the topology structure. Experimental results show that, the proposed method shows excellent performance both for artificial and real-world networks.

In this paper, we extend our previous work [20] and propose a scheme that combines the information obtained from the distributed environment to the information taken from the social media profiles, to identify users' membership to a community. A user's belonging degree to a community is determined not only by his/her likes, preferences in the social media, but also by the data they request. The latest, is a kind of information retrieved from a distributed environment. The belonging degree is then compared to a threshold value of the community. The main contributions of this work are the following:

- It introduces a novel method for community detection that, combines a distributed environment with communities.
- It uses fast and fully pipelined computations; thus, its cost is significantly reduced.
- It is efficient in the sense that, it shows competitive (and under certain circumstances superior) performance in terms of accuracy and the number of detected communities when compared to other strategies, while its execution time is less or equal to other well-known schemes.

The remainder of this paper is organized as follows: Section II-A defines our perspective of the community detection problem and we expose the usefulness of considering the user's data requests in a community detection scheme. Section III presents our strategy of community detection. Section IV presents our simulation results. Section V concludes the paper and presents aspects of future work.

II. PROBLEM DEFINITION- MOTIVATION BEHIND THIS WORK

It is commonly known that, people's personal networks can be quite big and their identification is very cumbersome. Generally, the idea to construct social networks is based on the assumption that, a society's users will have some features in common. In this sense, researchers have developed a number of techniques that, usually start from small predefined communities and implement some type of algorithm that 1) expands these communities, 2) detects newly formed communities, and (not in all cases) 3) finds overlaps between communities. Normally, the algorithms use some kind of similarity measure between users, in order to determine their membership to a community.

To feed the algorithm with some form of initial communities, researchers normally use real data sets, which are collected by observing the user's behavior in the net. The term "behavior" includes personal information, likes and preferences, Web sites and file requests, which altogether, can

TABLE 1. List of variables used in this paper.

Parameter	Description
$G(V,E)$	A weighted, undirected graph
C	A Community
V	Set of graph nodes, each node is a user
E	Set of graph edges, each edge represents a relationship between two users
N	A community node, $N \in V$
w_{ij}	Weight of the link connecting nodes i and j, denoting the *similarity* between i and j. Also, we use $w_{i \to j}$ if $i \to j$ is the working link processed by Algorithm 1.
NCD_i	Network connectivity degree
n_C	number of edges connecting users of a community
R_{new}	The new root at each iteration of Algorithm 1
L	List of direct connections of a user
L_i	Elements of list L
$U \to R_{new}$	The *working link*, during the execution of Algorithm 1, that is, the link being processed.
$\to$	Indicates the direction in which a working link is used, because during each pass of Algorithm 1, links are used only in one direction.
$.prev$	Pointer, linking to the previous node of a path processed by Algorithm 1
a	Variable that holds the current path similarity
ACC_C	Average Community Connectivity of community C
$\delta_{N,j}$	Distance between two nodes N and j, where j is a node requesting a file
f	a requested file
$S_{f,N}$	Scope of file f requested by node N
$\mathcal{T}$	Threshold value for the number of requests made by a user. It is used to decide the extent to which a single request enhances the similarity between a user and the members of a community.

be used to generate a user's profile that, will be later used with some algorithm to detect similarities between users. Such sets are freely available on the web by sources like the collection of 10th DIMACS Implementation Challenge[1] or the Stanford Large Network Dataset Collection:.[2]

A. PROBLEM DEFINITION

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 connections between two users. The *similarity* $w_{i,j}$ between users i and j is the weight of the edge that connects i and j. This value lies in the interval $[0 \ldots 1]$ and indicates how similar are their interests in terms of the files they request through the network. These users, i and j, may be researchers that belong to a research community or to a sports or a political community, where all users request similar files during the same time period (e.g., the members of a research community tend to request a number of files regarding a project, the members of a sports community tend to request news pages of their favorite sports team, and the members of a community interested in politics tend to request pages informing about governmental decisions, etc). In this case, a value equal to 0.78 shows that, the interests of i and j can be considered quite converging, at a percentage of 78%.

The network nodes are also weighted: the weight of a node i indicates its *Network Connectivity Degree* (NCD), i.e., how well the preferences, likes, views of a user are fitted to a community. The network connectivity degree is also between $[0 \ldots 1]$. The letters are the node names, the edge values indicate view similarities and the node values indicate similarity degrees. The network connectivity degree NCD_i of the i user is computed as follows:

$$NCD_i = \frac{\sum w_{i,j}}{n} \tag{1}$$

where $w_{i,j}$ is the weight of any edge that connects user i with any user j that lies in the same community and n is the number of such edges.

To decide if an external node is eligible to be considered as community member, we use the metric of *Average Community Connectivity* (ACC) for the target community C, which is defined as the average of all the NCD in C, that is:

$$ACC_C = \frac{\sum_{i=1}^{n} NCD_i}{n} \tag{2}$$

As we describe later, a node can be considered as member of C, if its similarity to the members of C, as computed by the proposed scheme exceeds ACC_C.

The variables that, will be henceforth used in this paper are listed in Table 1.

B. A MOTIVATING EXAMPLE

Let us use a brief example, to indicate the usefulness of taking into account the remote file requests when examining a user's membership to a community. Consider community C_2 in Figure 1. User N_9 has two internal links, namely with users N_7 and N_8. We compute NCD_{N_9}, as $NCD_{N_9} = \frac{(0.8+0.8)}{2} = 0.8$. This value is stored in node N_9 and indicates that, the requests of N_9 match the requests of the overall community by 80%. In this Figure note that, users $N_{10} - N_{12}$ are *independent*, i.e., they do not belong to any community.

According to Eq.2, $ACC_{C_3} = 0.9$. Assume that, we need to examine if node N_{12} can be a member of C_3 and we execute our community detection scheme (details and example execution are provided in the next paragraph) without

[1]see http://www.cc.gatech.edu/dimacs10/ downloads.shtml
[2]see http://snap.stanford.edu/data/index.html

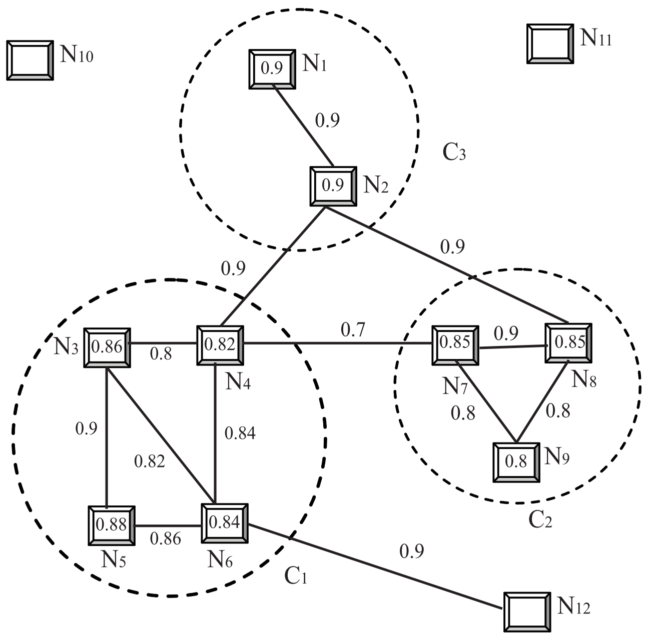

FIGURE 1. A network with three communities.

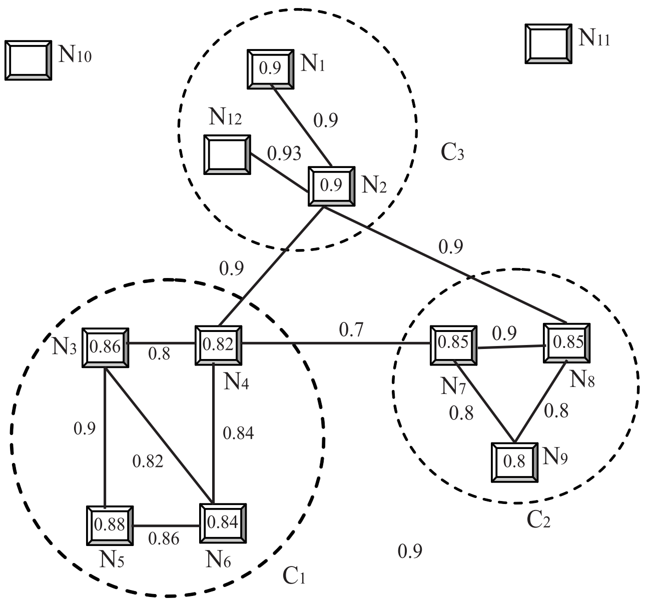

FIGURE 2. User, N_{12} is found to be a member of C_3, based on similarity metrics and its data and web requests.

considering the users' file requests, to find the similarity between user N_{12} and the members of C_3 (N_1, N_2). Unfortunately, the maximum similarity found is 0.68. Since $0.68 < 0.9$, N_{12} can't be considered as a member of C_3. Now, assume that we also decide to consider the file requests of N_1, N_2, and N_{12}, in order to find similar web page and file requests. Such information is available from server log files or can be found in encoded binary form in the data related to some real social networks (for example, see Stanford Large Network Dataset Collection Stanford collection). If N_{12} is found to have requested similar data as N_1 and N_2, and this data is considerably related to the community C_3, it makes sense for an algorithm to scale up the similarity between these nodes and compare it again to the community's ACC. As will be seen in the execution example presented in the next paragraph, this similarity will reach $\approx 93\%$ and in such a case C_{12} can be considered as member of C_3.

For a more concrete example, the students of the Department of Applied Informatics maintain a Facebook group, and a community detection algorithm will find many similarities between these members, by examining their features (the Experimental Results section explains how these features can be quantified as connectivity degrees). However, computer science professionals, or scientists working in the public sector may well be identified as community members based on their requests for the Department's Syllabus, course descriptions, the Departments Master Courses, the timetables etc. Indeed, this community includes a variety of members with highly different features, which could not be possibly identified by any algorithm based just on similarities.

PROBLEM DEFINITION: *Given a network $G = (V, E)$ and a number of independent nodes, our aim is to assign the independent nodes to a community, if this assignment is possible, in the most appropriate way, based on their data requests. This new assignment can lead to overlapping communities, i.e., communities with members in common.*

The next paragraph discusses our approach to the community detection problem.

III. COMMUNITY DETECTION SCHEME

Consider a network of irregular topology, such as the one shown in Fig.1, where there are already three communities C_1, C_2, and C_3. Also, assume that the links between nodes indicate that there exists some relationship between them, in the sense that they have similar opinions on topics or have the same interests and likes, based on their profile information [20]. A similarity close to 0 indicates almost no relationship, while a rank close to 1 indicates very close relationship. Fig.1 reveals some aspects than need to be considered:

1) User N_8 is related to user N_2 that belongs to C_3. Thus, there is a chance that N_8 also belongs to C_3, that is, there is *overlapping* between C_2 and C_3.
2) Not all users of a network are necessarily related (e.g., two Facebook users may be unrelated, but they may have common friends and may belong to the same communities). In Fig. 1, N_3 is not related directly to N_2, but it is related to N_4. In its turn, N_4 is related to N_2. To continue with the Facebook analogy, this is a case that a user (for example N_3) is unaware of a community (for example C_3) until he/she visits the profile of another user (like N_4) that is related to some members of this community. In other words, the path relating N_2 and N_3 (through N_4) may be strong enough, so the two nodes can be members of the same community.

3) The path strength described in the topic above can be further enhanced if N_3 requests files of particular interest to community C_3.
4) Newly connected users, like N_{10}-N_{12}, should also be examined and, based on their behavior, be characterized as members of one or more communities.

In this work, we investigate the existence of communities based on information retrieved by user profiles and by the users' data requests. To quantify the similarity between profiles, we investigate the similarities of the paths connecting pairs of users. To quantify the similarity between file requests, we introduce a *new metric*, called file scope. This metric has the advantage that, it does not require frequent communication between users. Instead, the user's interests are reflected by their periodical data requests and the belonging degree to a certain community is determined by the extent to which a user requests data that, is greatly requested by the existing members of the community. In the remaining of this section, we describe the community detection strategy. The main ideas behind the community detection strategy are summarized as follows:

- If the edge weight between two nodes is adequately large, chances are that the two nodes belong to the same community.
- In case of indirectly connected network nodes (via intermediate nodes), their relationship needs to be computed, based on their user profiles. We consider this computation as *static* in the sense that, profiling information that has already been collected and processed, usually remains unchanged or it is subject to slight modifications.
- Once the weights between all nodes have been computed then, the file requests metric is used to determine if the derived weight values can be boosted. If so, such boosting can act as an indication of a user's membership to a community. This computation is considered to be *dynamic*, in the sense that data requests apparently change from time to time and describe better the users' behavior. For example, an algorithm that has formed communities based on user profiles, may well add new members to them, when it is fed with recent information regarding user data requests.

A. STATIC ESTIMATION USING THE USER PROFILES AND LIKES

A user may be related to a community either *directly*, or *indirectly*, through one or more users. The similarities are obtained by using the information found in user profiles. These similarities are necessary to identify if users not directly connected can be members of the same community. Our approach suggests that, a user is indirectly related to a community via other communities, so we need a set of estimated values that quantify the relationship of a user U to a community C through communities formed in the network.

To get the static profile-based estimations, we will use the assumption that the similarity of two unconnected users is higher when the product of weights on the links that connect them (i.e., the *path similarity*) is as large as possible. We use the product of weights to reflect the case that, the similarity between two users decreases when their profile features and likes differ (the similarity values lie in the interval $[0\ldots 1]$). The algorithm uses a number of *passes*. Each pass is a series of links from a root node that, can lead either to a community C or to a node that can offer no possible way out to C. In both cases, the pass stops and another root is determined to continue with the next pass. The passes terminate when we have found the highest path similarity from the node examined to C. The procedure used to determine the similarities between unrelated users works as follows: We start from a user node that is examined for possible overlaps with a community C. This user is assigned as the new root R_{new}. The direct links of this node are placed in a list L in decreasing order of link weights w. We start with the first element L_1 of L where, w_{R_{new},L_1} is the maximum. L_1 becomes the new root R_{new} and the link $\mathrm{U}\rightarrow\ \mathrm{R}_{new}$ becomes the *working link*. Then, L is updated with the direct connections of R_{new}. To keep track with all the root updates, we use a pointer *prev* as follows:

$$L_1.prev = \mathrm{R}_{new} \tag{3}$$

$$\mathrm{R}_{new} = L_1 \tag{4}$$

We subsequently update the new root R_{new} and L in a similar way, until we reach a point where L_1 is a node in C. In each updating step, we also update the similarity a of the path formed by the selected roots:

$$a = a \times w_{\mathrm{R}_{new}\rightarrow L_1} \tag{5}$$

where a gets an initial value equal to 1 and successively it stores the similarity between the selected R_{new} and L_1. After reaching a node in C, we successively examine the remaining elements $L_2, L_3, \ldots L_n$ of L (L_n is the last element of L) in a similar way. Finally, we return to the current root and work backwards by setting as new root its *.prev* value:

$$\mathrm{R}_{new} = \mathrm{R}_{new}.prev \tag{6}$$

and repeat the process described.

The most important feature of the proposed scheme is cycle avoidance, that is, we avoid computations using a link that has already been used for computational purposes. To achieve this, we use two rules:

1. During each pass, a link can be used only in one direction (if $\mathrm{N}_i \rightarrow \mathrm{N}_j$ is used in a pass, then $\mathrm{N}_j \rightarrow \mathrm{N}_i$ can't be used in the same pass).
2. The list L of the direct links of a node excludes links that, have already been used during a pass or lead back to U.

Algorithm 1 gives the pseudocode for the procedure just described. The algorithm repeats until all possible paths to C have been processed.

We now use the example of Figure 1 to illustrate how this procedure works. Assume that, we wish to find the similarity

Algorithm 1 Computing Similarities Between Unrelated Users

```
input : All similarities between connected users,
        a community $C$, and a user U
output: The highest similarity between U and $C$
1  Set U = $R_{new}$; // new root node
2  Set $a = 1$; // stores the current similarity of the path
3  $LR_{new} = \{L_1, L_2, \ldots L_n\}$; // Links to the root in
   descending order of link weights
4  Start from link $R_{new} \rightarrow L_1$;
5  Mark link $R_{new} \rightarrow L_1$ for path as processed;
6  Update path similarity using (5);
7  Update the root using (3) and (4) and define $LR_{new}$ ;
8  while not all members of the list L are fully canceled do
9      if ($LR_{new}$ is fully canceled) or ($C$ is reached) then
10         Select new root defined by (6) // New pass ;
11         Define $LR_{new}$;
12         if $LR_{new}$ is not fully canceled then
13             return to line 4;
14         else
15             return to line 10;
16         end
17     else
18         return to line 4;
19     end
20 end
```

between a newly-entered independent node N_{12} and community C_3. Initially, we set $U = N_{12}$ and the corresponding list $L_{N_{12}}$ of node N_{12} will be $L = \{N_6\}$. We set $L_1 = N_6$ and now N_6 becomes the new root R_{new}. The working link $N_{12} \rightarrow N_6$ is marked as processed (line 5) and, the path similarity is updated using (5) $a = a \times w_{N_{12} \rightarrow N_6} = 1 \times 0.9 = 0.9$ (line 6). Then, the root is updated as shown in (line 7) thus, $N_6.prev = N_{12}$ and $R_{new} = N_6$. We have, $L_{N_6} = \{N_5, N_4, N_3\}$ (in decreasing order of the weights that link the root to its neighbors). The list L_{N_6} is not exhausted (it has unprocessed paths, see line 12) thus, we return to line 4 and select as new root R_{new} the node that is first in L (with the largest link weight), so $R_{new} = N_5$. Table 2 presents a complete pass. The asterisks near iteration numbers indicate the change of current root. For this example, two more passes will be required: In the second, the working link will successfully change from $N_{12} \rightarrow N_6$ (1st iteration) to $N_6 \rightarrow N_4$ (2nd iteration) and then, the pass will continue in a similar manner, as described in Table 2. In the third, the working link will successfully change from $N_{12} \rightarrow N_6$ (1st iteration) to $N_6 \rightarrow N_3$ (2nd iteration) and then, the pass will continue in a similar manner, as described in Table 2. Once the three passes are completed we get that, N_{12} has at least a 68.04% relationship to C_3 (the product of links $N_{12} \rightarrow N_6$, $N_6 \rightarrow N_4$, and $N_4 \rightarrow N_2$) and a 42.8% to C_3 (the product of links $N_{12} \rightarrow N_6$, $N_6 \rightarrow N_4$, $N_4 \rightarrow N_7$, $N_7 \rightarrow N_8$, and $N_8 \rightarrow N_9$). Both of these values are obtained in the second pass.

TABLE 2. Algorithm results for Fig.1, U= N_{12} and C_3 (first pass).

Pass	Iteration	Working Link	a	R_{new}	$L_{R_{new}}$
1.	1.	$N_{12} \rightarrow N_6$	0.9	N_6	$\{\cancel{N_{12}}, N_5, N_4, N_3\}$
	2.	$N_6 \rightarrow N_5$	0.774	N_5	$\{N_3, \cancel{N_6}\}$
	3.	$N_5 \rightarrow N_3$	0.696	N_3	$\{\cancel{N_5}, \cancel{N_6}, N_4\}$
	4.	$N_3 \rightarrow N_4$	0.571	N_4	$\{N_2, \cancel{N_6}, \cancel{N_3}, N_7\}$
	5.*	$N_4 \rightarrow N_2$	0.513	N_4	$\{\cancel{N_2}, \cancel{N_6}, \cancel{N_3}, N_7\}$
	6.	$N_4 \rightarrow N_7$	0.399	N_7	$\{N_8, N_9, \cancel{N_4}\}$
	7.	$N_7 \rightarrow N_8$	0.36	N_8	$\{N_2, \cancel{N_7}, N_9\}$
	8.*	$N_8 \rightarrow N_2$	0.324	N_8	$\{\cancel{N_2}, \cancel{N_7}, N_9\}$
	9.*	$N_8 \rightarrow N_9$	0.288	N_9	$\{\cancel{N_7}, \cancel{N_8}\}$
	10.*	–	–	N_8	$\{\cancel{N_2}, \cancel{N_7} \cancel{N_9}\}$
	11.*	–	–	N_7	$\{\cancel{N_8}, \cancel{N_9} \cancel{N_4}\}$
	12.*	–	–	N_4	$\{N_2, \cancel{N_6} \cancel{N_3}, \cancel{N_7}\}$
	13.*	–	–	N_3	$\{\cancel{N_5}, \cancel{N_6} \cancel{N_4}\}$
	14.*	–	–	N_5	$\{\cancel{N_3}, \cancel{N_6}\}$
	15.*	–	–	N_6	$\{\cancel{N_{12}}, \cancel{N_5}, N_4, N_3\}$

To decide if a node is eligible to be considered as community member, the estimated maximum path similarity is compared to the *Average Community Connectivity* (*ACC*) for the target community C, which is defined as the average of all the *NCD* in C, that is:

$$ACC_C = \frac{\sum_{i=1}^{n} NCD_i}{n} \tag{7}$$

If the estimated path similarity is greater than ACC_C then, the node examined can be considered as community member. In our example, $C = C_3$ and both nodes have $NCD = 0.9$ thus, $ACC_{C3} = 0.9$. Since the maximum path similarity computed was only 68.04% (through C_1) we conclude that, N_{12} can't be considered as member of C_3. In the next paragraph, the way that this path similarity measure can be boosted up by the data requests will be discussed so that, the membership of C_{12} can be reconsidered. In the remaining of this paragraph, we will discuss some performance issues regarding Algorithm 1.

1) COMPUTATIONAL ANALYSIS

To analyze the complexity of the estimation procedure just presented, one community C is initially considered. Since, for computational purposes, each link is visited only once, the maximum number of visits to a node is d, where d is the maximum number of direct connections that one community node may have. This means that, the maximum number of passes required is directed by the maximum d value among all communities, $\max_d$. Assuming that, each community C has n_C links and that, $m - 1$ communities are processed in total (the internal links of the community that we check for possible node memberships are not included in the computations since, once we reach a node in this community the current iteration stops), we need at most $\sum_{C=1}^{m-1} max_d \times n_C$ computations for our estimation procedure. In our previous example, the estimation process requires 26 computations (9, 9, 8 for passes 1, 2, and 3 respectively), and $\sum_{C=1}^{m-1} \max_d \times n_C = 4 \times 5 + 4 \times 3 = 32$. In Table 2, iterations 10 to 15 involve no computations since they are just root updates.

	Steps	1	2	3	4	5	6	7
Estimation for independent node	C12	RD	S1	S2	FIN			
	C13		RD	S1	S2	FIN		
	C14			RD	S1	S2	FIN	
	C15				RD	S1	S2	FIN

FIGURE 3. Timing diagram of pipelined computations for the network of Fig.1, U=N$_{12}$ − N$_{15}$ and $C = C_3$.

2) PERFORMANCE IMPROVEMENT

The main idea to efficiently improve the time required to run the static estimation procedure is to take advantage of the fact that, for all nodes that have similar direct links to a community, we can use the same set of computations. For example, in the network of Fig. 1, assume that three newly inserted nodes N$_{13}$− N$_{15}$− are directly connected to N$_6$ with different probabilities and we wish to find their similarities to C_3 through communities C_1 and C_2. The computations required are the same as the ones required for N$_{12}$. Thus, this problem example, as well as any other community detection problem involving similar computations is reduced to a typical pipeline computation where:

- RD is the segment that, reads the weights of the links between the four independent nodes and N$_6$.
- S1 is the segment that, estimates the path similarities between each independent node and C_3 through C_1 as presented in Algorithm 1.
- S2 is the segment that, estimates the path similarities between each independent node and C_3 through C_2 as presented in Algorithm 1.
- FIN is the segment that, reads the maximum similarity from S1 and S2, compares them, and assigns the final similarity value.

Figure 3 gives the timing diagram of this pipeline. For convenience we assume that, all stages require equal steps, which generally is not true and thus, delays need to be inserted in the actual diagram.

Pipelining speeds up the estimation times, especially for increasing number of newly inserted nodes. To compute the similarities of ℓ new nodes to a given community C using paths that pass through k other communities, we need $k+2$ pipeline stages: k stages to estimate path similarities through each community, a read stage (RD) and a finalizing stage (FIN). Totally, we need $\ell + (k+2) - 1 = \ell + k + 1$ steps are required, and each k step costs at most $\sum_{C=1}^{m-1} \max_d \times n_C$. This gives an overall computational cost via pipelining equal to $(\ell + k + 1) \times \sum_{C=1}^{m-1} \max_d \times n_C$.

B. DYNAMIC ESTIMATION BY IMPORTING THE SCOPE OF FILE REQUESTS

The explosive growth of the Web contents and Internet users has given rise to the issue of high file availability and sharing across the network. As the Internet continues to grow, efficient methods need to be implemented to handle the numerous data requests. In distributed computing, replication strategies are often used to improve file availability. In this paragraph, we combine the metric of file scope we introduced in [21] to achieve efficient data replication, with the community detection scheme presented in the previous paragraph, to dynamically estimate the probability of a node's membership to a community, based on its file requests.

A network carries huge amounts of data and a good part of it is of universal interest, e.g., a breaking news story. However, there is a considerable amount of data that is popular for restricted number of users with common interests. Such examples are data files belonging to a certain discipline, which can be shared and processed by specific users or webpages concentrating on a special topic, such as political discussions or news pages concerning a football team. Requests for such data can be crucial in determining the membership of a node to a community. This paragraph discusses the way that, the file scope can be imported, in order to define community memberships.

To quantify the extend to which independent users may have the same interests to the members of a community in terms of requested data, we need to quantify the relevance between their data requests and the data being processed by members of a community. For this reason, we introduce the notion of *file scope*. The file scope, as the name suggests, is a per-file estimated value that determines the relevance between the data requests of a user and the data requested mostly inside a community. To estimate the scope of a file, we first express the length of the path from a requesting node j for file f to node N, in terms of probabilities.

The file scope, as the name suggests, is a per-file estimated value that determines the relevance between the data requests of a user and the data requested mostly inside a community. To estimate the scope of a file, we first express the length of the path from a requesting node N for file f to node N', in terms of probabilities.

$$\mathcal{L} = \left[\prod_{r=1}^{\delta_{N,N'}} (1 - prob_r) \right] \times prob_N, \tag{8}$$

where $\delta_{N,N'}$ is the distance between N and N' measured in number of nodes and $prob_r$ is the probability that a file is cached in a node r, on the path from N to N'. Equation (8) expresses the fact that, a request for a file f is forwarded from N to N' through a number of nodes that do not cache f. This probability is the product of the probabilities that each of these nodes do not cache f, multiplied by the probability that node N' caches it. When the requested file is of special interest to a community, chances are that this file should be located in the community nodes or (with much lower probabilities) to nodes that have some connections to community members [10].

Having estimated $\mathcal{L}_{N,j}$, we now estimate the scope for file f using the following rules:

1. File requests between members of a community are assumed to have zero scope.
2. The larger the value of $\mathcal{L}$, the smaller the scope assigned to the requested file. A large value of $\mathcal{L}$ indicates that the requested data may be found nowhere, but in nodes residing in a community or at least have some reasonable similarity to it.

Based on the following rules, we estimate the scope $S_{f,N}$ of file f requested by node N, as follows:

$$S_{f,N} = 1 - \mathcal{L} \qquad (9)$$

Apparently, the scope increases for file requests made by users that appear to have weak relationships to a community (thus, a request may need to be forwarded between many nodes until it is completed by a node inside the community). When the request is completed after a few only forwardings, it is suggested that, there is a stronger relationship to the community. If the request is made and completed internally in the community, the scope becomes 0. This is because the probability of *not* finding a file popular in a community inside a node equals to 0 and the probability of node N caching the file is 1; making $\mathcal{L}$ equal to 1 and $S_{f,N}$ equal to 0.

The use of the file scope relies on the important principle of *spatial locality*, which states that data related to previously requested data have very high probability to be requested soon [21]. The file scope is imported in the community detection scheme in the following steps:

1. We set a threshold of $\mathcal{T}$ data requests of a user U, whose membership to a community C is examined. These data requests will be used to decide whether the similarity, estimated using the comment-based metric, will be augmented or not. To have a fairer estimation, the value of w needs to reflect the average number of file requests made by the members of community C over a period of time.
2. We estimate $\mathcal{L}$ for these data requests using (8).
3. If the data requested is found *only* in nodes inside the community then: (a) it follows that the requested data is of special interest to the members of C, and (2) due to the *spatial locality* principle, more files of special interest to C will probably be requested.
4. For the data requests mentioned in the previous step, we compute the scope using (9).
5. We increment the estimated static similarity by $S_{f,N}/\mathcal{T}$ and check if the new boosted value has grown higher than the average community connectivity of C (ACC_C). If so, the node examined can be considered as community member. Otherwise, the process described repeats until either the ACC_C is overcome or the number of requests is completed and, based on these requests, the node can't be considered as community member.

To return to the network example of Fig. 1, let us assume that $\mathcal{T} = 20$ and that node N_{12} makes some file requests which are served by N_2. With $prob_r$ =50% and using the highest similarity path found during the static computation, $\mathcal{L} = 6.25\%$. This gives $S_{f,n} = 93.75\%$ Then, $S/\mathcal{T} = 4.69\%$. Thus, each request would increment the computed similarity of 68.04% by 4.69% and if there are seven requests for data found only in C_3, the similarity would successfully increase to 71.23% 74.57%, 78.06%, 81.72%, 85.55%, 89.56%, 92.76%, which exceeds the average community connectivity of $C_3 = 90\%$. In such a scenario, the algorithm decides that, although node N_{12} has a similarity of 68% to C_3 based on the profiling information and user preferences and likes, the data it requests indicate much higher relationship to C_3, so it can be considered as member. Then, N_{12} will enter C_3, the weight that links N_{12} to N_2 will be $\approx 93\%$ and this in turn will increase the overall connectivity of C_3 (see Figure 2).

IV. SIMULATION RESULTS

To verify the community detection scheme, two sets of simulations were implemented:

(a) *Synthetic networks:* Synthetic networks which have been regularly used in a variety of research works (for example, [10], [22]). Because synthetic networks are generated randomly, they are useful in studying the behavior of a proposed scheme. For these sets of simulations, we produced random data based on statistic distributions.

(b) *Real-world networks:* We used two real networks, `Facebook` and `Google+`. For these simulations, we used real data, taken from the Stanford Large Network Dataset Collection Stanford collection

The file scope is generally a distributed-based metric, which is used in applications such as data replication. For the purpose of our experiments, we treat the communities detected by our static estimations as distributed networks, where their links and nodes are distributed over geographical locations. This approach with simulated data has already been used in other research works like [18]. In the simulations followed, we used three widely used metrics for comparisons (see [23]): The number of communities detected, the Normalized Mutual Information (NMI) and the execution times.

A. SYNTHETIC NETWORKS

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 generate the network, we used the benchmark provided by Lancichinetti and Fortunato [24]). The following parameters are required by the benchmark:

- The degree of each node, where χ_{min} and χ_{max} are the minimum and maximum node degrees, respectively.
- The mixing parameter, that 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.

Session	IP Address	Date/Time	Requested File	File Size	User Agent
1	195.251.213.23	12/Sep/2018 10:15:32	http://www.uom.gr/media/docs/efpl/SYLLABUS-2015-2016-EN.pdf	1594702 bytes	Chrome
1	195.251.213.23	12/Sep/2018 12:23:42	http://www.uom.gr/media/docs/efpl/ep-parartima-diplomatos-en.pdf	301056 bytes	Chrome
1	195.251.213.23	12/Sep/2018 12:25:38	http://afroditi.uom.gr/erasmus/?q=node/95/ Application_Form.doc	792576 bytes	Chrome
1	195.251.213.23	12/Sep/2018 12:28:02	http://www.uom.gr/modules.php?op=modload&name= Cv_eng&file=index&id=1386&tmima=6&categorymenu=2	14336 bytes	Chrome
2	195.251.213.23	12/Sep/2018 13:05:11	https://developer.ibm.com/code/community/events/IEAA3JYPKQG5QKMQ-london-java-community-ljc-meetup-ibm-london-09-18-2018	13276 bytes	Chrome
2	195.251.213.23	12/Sep/2018 13:15:11	https://github.com/IBM/dsx-twitter-auto-analysis/blob/master/data/examples/sample_output_files/ sample_output_14_0.png	15052 bytes	Chrome
2	195.251.213.23	12/Sep/2018 13:21:18	https://github.com/IBM/dsx-twitter-auto-analysis/blob/master/data/examples/sample_output_files/ sample_output_20_0.png	44748 bytes	Chrome
2	195.251.213.23	12/Sep/2018 13:25:00	https://github.com/IBM/dsx-twitter-auto-analysis/blob/master/data/examples/ sample_output_files/sample_output_41_0.png	40755 bytes	Chrome

FIGURE 4. Data requests from a user of IBM community: Session 2 requests are for files processed by members of the community. Session 1 are requests for files mostly requested by the students of the Department of Applied Informatics, University of Macedonia.

- 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 graph.

For our experiments, we defined the average node degree of each node equal to 15 nodes and the mixing parameter (which is the average ratio of external degree/total degree for each node, denoted by μ) from 0.1 to 0.4. Each node shares $1 - \mu$ of its links with the members of its community and μ with members of other communities. Also, we run our simulations for small-sized networks including from 1000-5000 nodes.

When the network is generated by the benchmark, the static estimation is executed to find the path similarities. This will give an initial set of communities. These communities were labeled (for example, `IBM Employees, High School Teachers, Baseball Fans, etc` and sets of files relevant to all these communities were collected, to facilitate the simulation process. The files mostly relevant to each community are considered to be processed by all the community members. We named these files as community's Class A files. We assigned 50 Class A files per community. Apart from class A files, each community has Class B files, which include the Class A files of all the other communities. So, each community member requests a number of files from the repository of Class B files. These file requests are produced using the zipf distribution. The zipf distribution represents a situation, where the probability of making a request for some files is higher compared to the probability of requesting other files. Thus, the request probabilities of Class B files rely on their contents. This is close to reality, as for example, a male IBM employee is more likely to request the syllabus of a computer science course than a file containing sets of images of female dresses.

To satisfy the spatial locality principle, a request for a file belonging to a subclass is followed by a few subsequent requests for files of the same subclass. In other words, when a node requests a files from the repository of another community's Class A files, it is more likely to request another file from the same repository.

An example of file requests is given in Fig 4. A member of the IBM employees community requests 4 files (Session 1), from Class A files of the community of students of the Department of Applied Informatics, University of Macedonia. Session 2 includes requests for files used mostly by the members of the IBM employees community. Such requests may indicate that, this user is somehow related to this department (for example, he is looking for a master's degree program, he has graduated from this department, or looking for the contents of a programming course being taught).

In this section, we compare our strategy with a number of other known schemes. The metrics we used for comparisons are the number of communities detected and the NMI (Normalized Mutual Information). To compare the number of strategies generated by our scheme to other known strategies, we run four simulation sets on relatively small synthetic networks with simulated data. In the first one, we measured the number of communities detected for a network of 1000 nodes, with an average number of file requests for each node, $\mathcal{T}$ equal to 20. The mixing parameter ranged from 0.1 to 0.4. In the second set, the network includes 2500 nodes and R increased to 40. In the third set, our network has 5000 nodes and there are $\mathcal{T} = 50$ data requests per node. Finally, in our last set we computed the simulation time required to find the communities for each network, while increasing the number of independent nodes entering the network. Table 3 shows the

TABLE 3. Simulation parameters.

Parameter	Value
Number of nodes N	1000 - 5000
Mixing parameter μ	0.1 - 0.4
Minimum node degree, χ_{min}	10
Maximum node degree, χ_{max}	30
Average node degree, χ	20
Average number of data requests per node, $\mathcal{T}$	20 - 50

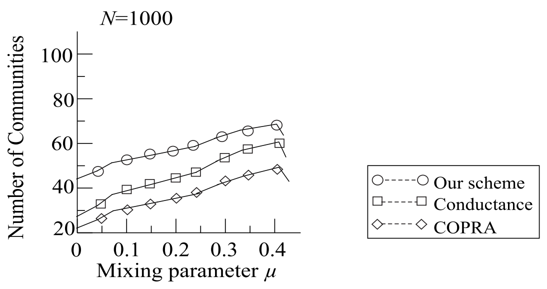

FIGURE 5. Simulation results for $N = 1000$, $R = 20$.

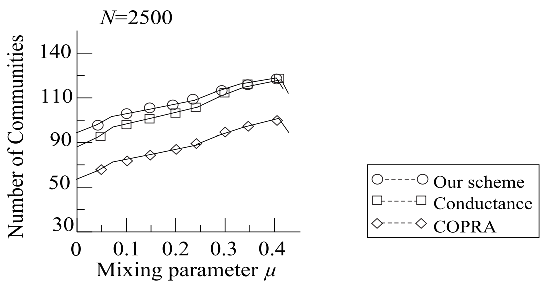

FIGURE 6. Simulation results for $N = 2500$, $R = 40$.

parameters used in our simulations. In the following Figures, each point is an average value taken from 10 simulation runs.

Figure 5 presents the simulation results for a small network of $N = 1000$. The proposed scheme detected 55 communities on the average (the number varied for different values of μ) and some of them included very few nodes (4 to 10). Our scheme detects more communities compared to Conductance and COPRA, due to the fact that, when a node requested class A data from a community, the value of R was small enough to converge quickly to a similarity value that permitted the node to be considered as community of this community.

Figures 6 and 7 shows the simulation results when the network becomes larger, that is $N = 2500$ and $N = 5000$. Generally, since the average number of data requests per node, R, increases, it is not easy to converge to a large similarity value; thus the boosting factor $S_{f,N}/R$ becomes rather small. Therefore, the proposed scheme detects smaller number of overlaps based on the data requests and the majority of communities is detected through the static profile based estimations.

Moreover, as the network enlarges, the communities detected by our algorithm tend to have higher ACCs and the path similarities estimated tend to have lower values

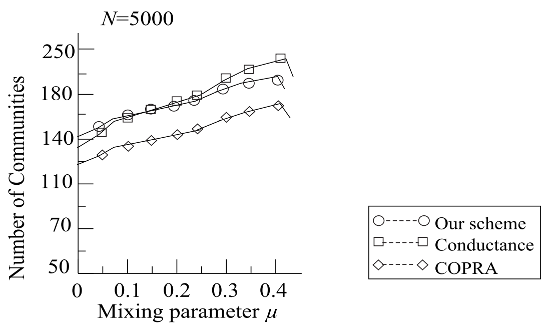

FIGURE 7. Simulation results for $N = 5000$, $R = 50$.

(since the paths become longer and our scheme is based on products of path values). A fairer solution may be to set w to half of the average number of requests so that, the path similarities can be increased by larger factors. As can be seen from the Figures, as the network becomes larger, the results of our scheme and the conductance algorithm are very close (for $N = 5000$ and values of $\mu > 0.2$ the conductance strategy detects slightly more communities compared to our scheme), but still overcome the results of label propagation based scheme (COPRA). Finally, note that in all simulation sets, when μ exceeds 0.4, the number of detected communities falls off. This is due to the fact that, a community can't be strongly defined with increasing number of external links, regardless of the file requests.

In the last set of simulations, we examined the benefit of pipelining the computations of the static path similarity estimations, for the various parameters of our experiments. Note that, as the network is generally small (1000 nodes), the increase of the total time of execution is very smooth. The time required to complete the data detection execution slightly increases from 2 seconds, as we keep increasing the independent nodes from 20 to 40. However, as the network becomes larger the increase is not so smooth (see for example the line that corresponds to a network of 5000 nodes). This is due to the fact that, the independent nodes have increasing number of connections to more communities, so it takes more time to examine their similarity to a community (more different paths need to be examined). For 20 independent nodes, the total time of execution is more than quadruple compared to the execution with the same parameters on a network with $N = 1000$. Figure 8 shows the results for this set of experiments.

To evaluate the NMI, we used larger synthetic data networks of 50,000 nodes, while keeping values of the other parameters as shown in Table 3. We compared our strategy to two well-known strategies, weak cliques ([8]) and FOCS, a fast overlapping community scheme based on the idea of local expansion and optimization ([5]). We used the typical parameter values for β and an average node degree equal to 20 (minimum node degree 10 and maximum node degree 30). The other parameters are as shown in Table 3. Figures 9 and 10 show the experimental results. Each point

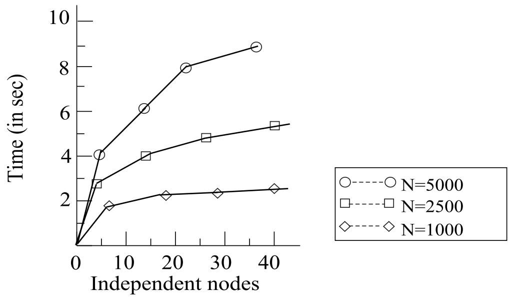

FIGURE 8. **Simulation time of execution for the various parameters of the experiments.**

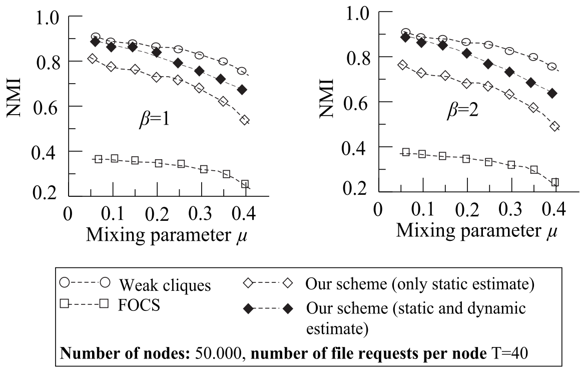

FIGURE 9. **NMI comparisons, for $\mathcal{T} = 40$.**

is an average value taken from 10 random realizations of networks with the aforementioned parameters. We plot two curves related to our strategy: one is taken from the static estimations of similarity paths, while the second shows how the dynamic estimations (file requests) increase the strategy's NMI. In both figures, our static approach outperforms FOCS but the weak cliques approach presents higher NMI values.

In Fig. 9 we assigned the number of file requests, $\mathcal{T} = 40$ for our dynamic estimations. Our NMI values approach (and, for values of μ up to 0.1 and $\beta = 1$ outperform) the weak clique approach. In Fig. 10 we assigned the number of file requests, $\mathcal{T} = 20$. In this case, our strategy (static and dynamic estimation) outperforms the weak clique approach, because, as $\mathcal{T}$ decreases, a request for a community's class A files made be a user will increase the user's similarity to this community by higher percentage (see the last paragraph of Section III). Finally, as the communities tend to have similar size (increasing β from 1 to 2) our scheme's NMI values slightly decrease because when communities have almost equal sizes then, only parts of them have similarities between them, in terms of profile information and file requests. On the other hand, when there are large and small communities, a small and a large community may be entirely overlapping.

To summarize the main conclusions of our simulations using synthetic data networks:

1. For small synthetic networks, our scheme detects more communities compared to well-known strategies like COPRA and Conductance

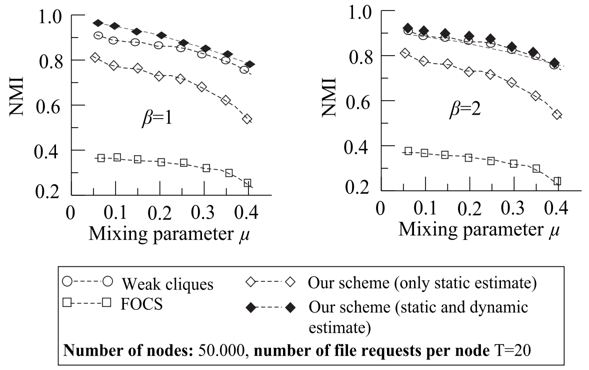

FIGURE 10. **NMI comparisons, for $\mathcal{T} = 20$.**

TABLE 4. **Statistic properties of the datasets for facebook and google+.**

Network	Num. of Nodes	Num. of Edges	Num. of Triangles
Facebook	4.039	88.234	1.612.010
Google+	107.614	13.673.453	1.073.677.742

2. Our static estimation approach (without considering the file requests) outperforms the FOCS algorithm, but not the weak clique approach.
3. When the file requests are taken into account, then
   a. When a relatively small number of requests for class A files of a community are enough to boost the similarity between the requesting node and members of this community ($\mathcal{T}$ is relatively small), then our strategy slightly outperforms the weak clique approach.
   b. When more requests for class A files of a community are required to boost the similarity between the requesting node and members of this community (the threshold value $\mathcal{T}$), then our strategy is outperformed by the weak clique approach.

B. REAL NETWORKS

To evaluate the performance of our proposed scheme on real networks, we used two real-world datasets: `Facebook` and `Gplus`. Our algorithm will use this data, in search of *social circles* or *social lists*.[3]

Table 4 presents the basic statistic properties of the networks being used. The diameter values show the number of nodes that should be traversed to travel from one vertex to another, excluding backtracks and loops.

For each network, a combination of features describing the users was connected. The data, as presented in Stanford Large Network Dataset Collection were not in a proper form for the purpose of our algorithm, so we had to make some kind of adjustment, to adapt the data to the input demands of our scheme. Here, we describe the transformations we made for Facebook network. For Gplus, we worked similarly. For each user examined, a set of combined features

[3]The first term is used in Google+, while the second is used in Facebook, but they practically mean the same thing.

TABLE 5. Some combined features of users related to user with ID = 0, as shown in stanford large network dataset collection

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
77	*gender;anonymized feature 77*	Male or female (see Attribute ID=88 also)
78	*gender;anonymized feature 78*	Male or female (see Attribute ID=87 also)
79	*hometown;id;anonymized feature 79*	User's hometown
100	*languages;id;anonymized feature 100*	Languages spoken
140	*work;employer;id;anonymized feature 140*	Current work position, employer (for example programmer, IBM)

TABLE 6. Bit_vectors for the attributes subset of Table 5.

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

was generated. This set includes all the combined features possessed by *at least* two users, with whom the user is related. In the collection, one can find data for 26 attribute categories, including education, birthdays, political affiliations, schools, etc. Table 5 shows 10 combined features, collected from the users related to with ID = 0. For these users, there are 224 different combined attributes (shown in file named 0.featnames). Also, note that the attributes are encoded as integer values, to maintain users' privacy. In Table 5, the attribute with ID = 0 corresponds to an age (apparently, one can find many more age attribute IDs, that, taken together, describe the range of ages of the users related to user with ID = 0), the attribute with ID = 21 describes a combination of education and degree (again there are many combinations of education types and degrees with different IDs), etc.

Each user forms circles with a set of nodes, based on the attribute values. These circles will be used as the initial communities for our algorithm. First, we give a circle example and then we will formally describe how to transform the given data to produce our initial communities in the form of weighted graphs. Based on the data given in Stanford Large Network Dataset Collection, circle_13 of user with ID = 0 includes the node ID's (or users) 138, 131, 68, 143, 86. The binary values at positions 0,21,30,53,72,77,78,79, 100, and 140 of their bit_vectors are given in Table 6. From these values, it follows that, taking into consideration the 10 attributes of Table 5, the similarities $w_{i,j}$ between each pair of nodes (i, j) can be computed by taking the bitwise Exclusive-OR (XOR) of their corresponding bit_vectors I, J into bit_vector W, counting the number of 1s resulted, and divide this number by the total number of attributes examined, in this case, 10. Mathematically:

$$w_{i,j} = \frac{\text{Num. of 1s in vector } W}{\text{Num. of Attributes}}, \quad \text{where} \quad W = I \oplus J \qquad (10)$$

By implementing Eq.10, we get the following similarity values between the pairs of nodes of the circle:

$$\begin{aligned} &w_{0,138} = 0.9, \quad w_{0,131} = 0.9, \quad w_{0,68} = 0.7 \\ &w_{0,143} = 0.8 \quad w_{0,86} = 0.7 \; w_{138,131} = 1 \\ &w_{138,68} = 0.7, \quad w_{138,143} = 0.7, \quad w_{138,86} = 0.8 \\ &w_{131,68} = 0.7, \quad w_{131,143} = 0.7, \quad w_{131,86} = 0.8 \\ &w_{68,143} = 0.8, \quad w_{68,86} = 0.9, \quad w_{143,86} = 0.9 \end{aligned}$$

For the simulations described in the next paragraphs, we used the communities (circles) found in [25] for given users as our initial communities and then we implemented the static estimations to detect new communities, examine a user's membership to these communities and spot community overlaps. Then, we used the data requests generated by our simulator for synthetic networks, to examine how they can enhance the link weights computed by Eq. 10. The value of $\mathcal{T}$ was chosen to be 40, Our experimental results were compared to other known schemes implemented on networks with similar sizes as Facebook and Gplus. The metrics used for comparisons were the running time, the number of detected communities and the NMI. Table 7 shows the results found by a set of algorithms on a number of real networks. These networks have similar sizes to Facebook and Gplus, so our results are quite comparable. However, the Gplus has the disadvantage of containing a huge number of edges, a problem that is reflected in the execution time of our scheme.

From table 7, we can make some observations. First, the FOCS strategy shows higher efficiency in terms of total execution time. Our scheme performs well in small networks like Facebook, but when it comes to a larger network like Gplus, its efficiency drops. However, this is normal since our strategy is based on examining the edge values and Gplus has a huge number of edges compared to the other networks. Still, the time required by our strategy outperforms COPRA and matrix factorization scheme.

Second, our strategy outperforms all the compared schemes in terms of the number of detected communities. Our scheme detects 26200 communities for Gplus, which is the maximum value found in Table 7. For small networks like Facebook, our scheme detects 230 communities, which is satisfactory compared to the number of communities found by the other schemes. One exception is the matrix factorization scheme that detects 1078 communities, but in a network

TABLE 7. Various simulation results on a number of real networks.

Strategy	Networks	# Nodes	# Edges	Ex. Time	NMI	# Communities
FOCS [5]	Amazon	334863	925872	2 sec	0.2075	20900
	DBLP	317080	1049866	2 sec	0.2135	24200
	Human PPIN	9300	32200	<1 sec	0.2471	32
Weak Clique [8]	Amazon	334863	925872	106 sec	0.3097	11000
	DBLP	317080	1049866	103.6 sec	0.1760	56500
Matrix Factorization [26]	Amazon	334863	925872	1.18h	0.2421	151000
	DBLP	317080	1049866	33 min	0.1448	39600
	Human PPIN	9300	32200	57 sec	0.0328	1078
COPRA [16]	Amazon	334863	925872	1183 sec	0.2076	8400
	DBLP	317080	1049866	180 sec	0.1484	14900
	Human PPIN	9300	32200	1 sec	0.2510	26
Our scheme	Facebook	4039	88234	<1 sec	0.33	230
	Gplus	107614	13673453	170 sec	0.35	26200

with double the Facebook size. Also, we have to note that our scheme uses an average number of $\mathcal{T} = 40$ requests. Generally, our scheme detects more communities if $\mathcal{T}$ is reduced and its performance drops when $\mathcal{T}$ increases. Similar observations were made on synthetic networks regarding the number of detected communities. However, real networks appear to have lower *ACC* values compared to synthetic networks (their community structure is rather vague). Thus, the file requests can increase the path similarities at a high extent, even for values of $\mathcal{T}$ up to 40. For synthetic networks, our results were better for values of $\mathcal{T} \leq 20$ (see Figures 9 and 10).

Finally, our strategy outperforms the competitive schemes in terms of NMI. This result was also confirmed for small values of $\mathcal{T}$, when we run our simulations on synthetic networks. However, in the case of real networks, the community structures are rather unclear (for example, the ACCs of the initially formed Facebook and Gplus communities were rather small, in some cases even 0.4). This means that, the file requests can act as a "correcting factor" that increases the NMI values. Experiments have shown that, even for values of $\mathcal{T}$ up to 40, our scheme provides better NMI values.

V. CONCLUSIONS-FUTURE WORK

This paper presented a novel approach for community detection in which, the profile information of users is combined with their data requests. To determine a user's membership to a community, the method tries to get an estimation of the user's similarity to the community based on static data such as profiling information. This work is done via a series of passes, where in each pass a link can be used only once for computations, to reduce the time required. A pipeline based pattern of execution can be employed to further reduce the time of execution. Then, user's data requests are recorded and assessed using a new metric called file scope. This metric boosts up the similarity computed using the static data in cases where, the user requests data exclusively used inside the community.

The simulation results have exposed some strengths and some limitations of the proposed scheme. The two main strengths are the following: first, it improves the quality of the detected communities in real networks, as this is expressed by the NMI value. For synthetic networks, with more compact structure, our scheme works competitively to well known schemes and it is superior when the chosen threshold for data requests is relatively small. Second, the execution time of our scheme can be pipelined to deal with processing of a large number of edge values. This execution strategy makes it competitive with well known strategies.

Our strategy has also shown two basic limitations: First, if the threshold value $\mathcal{T}$ of file requests is set to a high value, the path similarity values do not increase largely and as a result some possible overlaps do not appear. This is especially true in synthetic networks. Second, although our simulation results have shown our strategy's efficiency in terms of execution time/number of communities detected, it is clear that, as the networks become too dense (large number of edges), its execution time becomes a burden. We used a pipeline scheme to soothe this problem, but a better idea would be to carefully schedule a parallelizable version of this strategy. This is part of our future work.

Apart from parallelizing this scheme, our future work includes the consideration of the vectors produced by the real data available on the net, which may include useful information of data requests. An idea would be to examine separate parts of these vectors. For example, in the Gplus network, a sub-vector of the bit_vector related to a user may include information regarding his/her shopping interests. Such information may have been recorded using his/her data requests. Finally, this work has given rise to the idea of implementing community-based data replication.

REFERENCES

[1] 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.

[2] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2010.

[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, p. 036106, 2007.

[4] M. Wang, C. Wang, J. X. Yu, and J. Zhang, "Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework," *Proc. VLDB Endowment*, vol. 8, no. 10, pp. 998–1009, 2015.

[5] S. Bandyopadhyay, G. Chowdhary, and D. Sengupta, "FOCS: Fast overlapped community search," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 2974–2985, Nov. 2015.
[6] I. Farkas, D. Ábel, G. Palla, and T. Vicsek, "Weighted network modules," *New J. Phys.*, vol. 9, no. 6, p. 180, 2007.
[7] 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, p. 026109, 2008.
[8] 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. Comput. Social Syst.*, vol. 4, no. 4, pp. 218–230, Dec. 2017.
[9] 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, 2010.
[10] 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.
[11] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, no. 7307, pp. 761–764, 2010.
[12] S. Qiao *et al.*, "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.
[13] 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, 2013.
[14] D. Džamić, D. Aloise, and N. Mladenovič, "Ascent–descent variable neighborhood decomposition search for community detection by modularity maximization," *Ann. Oper. Res.*, to be published.
[15] R. Santiago and L. C. Lamb, "Efficient modularity density heuristics for large graphs," *Eur. J. Oper. Res.*, vol. 258, no. 3, pp. 844–865, 2017.
[16] S. Gregory, "Finding overlapping communities in networks by label propagation," *New J. Phys.*, vol. 12, no. 10, p. 103018, 2010.
[17] K. Taha, "Disjoint community detection in networks based on the relative association of members," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 493–507, Jun. 2018.
[18] Z. Bu, Z. Wu, J. Cao, and Y. Jiang, "Local community mining on distributed and dynamic networks from a multiagent perspective," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 986–999, Apr. 2016.
[19] W. Zhi-Xiao, L. Ze-chao, D. Xiao-Fang, and T. Jin-Hui, "Overlapping community detection based on node location analysis," *Knowl.-Based Syst.*, vol. 105, pp. 225–235, Aug. 2016.
[20] S. Souravlas and A. Sifaleras, "Efficient community-based data distribution over multicast trees," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 1, pp. 229–243, Mar. 2018.
[21] S. Souravlas and A. Sifaleras, "Binary-tree based estimation of file requests for efficient data replication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 1839–1852, Jul. 2017.
[22] 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.
[23] T. Chakraborty, A. Dalmia, and N. Gangul, "Metrics for community analysis: A survey," *ACM Comput. Surv.*, vol. 50, no. 4, pp. 1–54, 2017.
[24] 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, p. 016118, 2009.
[25] J. Leskovec and J. J. McAuley, "Learning to discover social circles in ego networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1, 2012, pp. 539–547.
[26] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, Rome, Italy, Feb. 2013, pp. 587–596.

STAVROS SOURAVLAS (M'13) received the Ph.D. degree in computer science from the University of Macedonia, Thessaloniki, Greece. In 2014, he joined the Department of Applied Informatics, School of Information Sciences, University of Macedonia, where he is currently an Assistant Professor of computer architecture and digital logic design. His research interests include computer architecture and performance evaluation, parallel and distributed systems, grid computing, cloud computing, systems modeling and simulation, and big data. He is an Associate Editor of the IEEE Access.

ANGELO SIFALERAS is currently an Assistant Professor with the Department of Applied Informatics, School of Information Sciences, University of Macedonia, Thessaloniki, Greece. His research focuses on mathematical programming and network optimization. He is a Senior Member of ACM.

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