Overlapping Communities via k-Connected Ego Centered Groups

Günce Keziban Orman*, Onur Karadeli†, Emre Çalışır†
*Computer Engineering Department, Galatasaray University
Ortaköy 34349 Istanbul, Turkey
Email: korman@gsu.edu.tr
†Vodafone Teknoloji Hizmetleri A.Ş. (OKSIJEN)
Maslak 34467 Istanbul, Turkey
Email: onur.karadeli@vodafone.com, emre.calisir@vodafone.com

***Abstract*—Overlapping community detection allows placing one node to multiple communities. Up to now, many algorithms are proposed for this issue. However, their accuracy depends on the overlapping level of the structure. In this work, we aim at finding relatively small overlapping communities corresponding family members or very close friend groups in reality. We define *k*-connected node groups as cohesive groups in which each pair of nodes has at least *k* different node disjoint paths from one to another. We propose the algorithm EMOC first finding *k*-connected groups from the perspective of each node and second merging them to detect overlapping communities. We evaluate the accuracy of EMOC on artificial networks by comparing its results with foremost algorithms. The results indicate that EMOC's performance is not affected by overlapping level of neither nodes nor communities. Results on real-world networks show that EMOC finds relatively small but consistent communities. Overlapping or community-less nodes might have important topological positions such as being hubs or bridges.**

I. Introduction

In network analysis, one of the most studied research domain is communities [1]. Roughly speaking, a community corresponds to a group of nodes with denser inner links and sparser outer links [1]. The communities are widely studied for discovering functionally related objects, finding interactions between modules, inferring missing attribute values and predicting unobserved connections, etc. [2]. Because of this popularity, there are hundreds of different works dedicated to community detection [1]. In general, there is not a consensus of the formal definition of community but it depends on the need of application or studied system. Right now, we encounter several approaches considering communities as disjoint node groups [3], [4]. Besides, there is also a high interest to overlapping communities [5]–[7]. An overlapping community structure can be defined as a cover including different communities in which any two communities might have common nodes.

In [8], the authors show that overlapping communities are significant features in social networks. Finding them benefits to discover the dynamics of social interactions. For instance, in a social environment, one person can belong to multiple communities at same time, e.g. his coworkers, his social friends or his family members. Overlapping parts of different communities may represent similar sides of those social groups. Or, people belonging to only one single community may be identified by one single interest. Moreover, intersection amount of two communities may give an idea about their future state, i.e. those groups might merge. As a result, detection of overlapping communities becomes an important task in social network analysis.

Up to now, many works are proposed for overlapping community detection [5]–[7], [9]–[11]. However, it is seen that their accuracy depends on the overlapping level of the communities [5], i.e. the number of communities that a node can belong or the amount of overlapping nodes in the network. Moreover, in reality, we encounter that some people are community-less although most of the methods put all the nodes into at least one community [6], [7], [10]. In this work, we aim at identifying overlapping communities independently than overlapping level. Those communities correspond small closely related groups like family members or very close friend groups in reality. In this context, we also want to extract community-less nodes. For this aim, we are oriented at cohesive node groups that each node can belong in its ego-centered network.

In [11], the authors propose an approach that we call EGO-BASED in the rest of this article. EGO-BASED is based on the groups around each ego. It, first finds the *friendship groups* related to an ego which correspond to different connected nodes after removal the ego in its ego-centered network at radius 1. Then final communities are detected by merging friendship groups found for whole network. Merging criterion of two groups is having at least one less intersecting nodes than the size of one of those groups. This approach is limited because of some reasons; (1) not taking into account larger radius than one at ego-centered network creation, (2) not regarding the amount of connectedness for the nodes for being in the same friendship groups and (3) considering to have only one node difference while merging. These limits can cause neglecting densely connected large friendship groups, finding non cohesive groups or not merging highly similar large groups respectively. We propose a generalized approach of [11] for overcoming the limits we notice. First, we consider ego-centered network at a specific user-defined radius. Second, we regard the numbers of node disjoint paths as a criteria of cohesiveness of node groups. Third, we use a user-defined similarity threshold for merging. As a natural result of this approach, the nodes which are not connected enough to any ego are labeled as community-less.

Our first contribution is defining *k-connected* node groups and overlapping communities through them. Our second contribution is developing and implementing the algorithm EMOC to find such communities. Our third contribution is evaluating the accuracy of EMOC on artificial networks. We use LFR benchmark [12] for network generation and modified NMI [13] measure for comparing estimated and reference communities. We also perform a comparative analysis of EMOC with some prominent overlapping methods such as OSLOM [9], GCE [7], MOSES [10], COPRA [6] and EGO-BASED [11]. Our last contribution is testing EMOC on real-world networks. In the following section, we explain some prominent works about overlapping community detection. In section III, we give the definition of node groups we deal with and explain the details of EMOC. In section IV, we describe LFR model, accuracy results of the algorithms, time performance of EMOC and results of EMOC on real-world networks respectively. Finally, in last section, we give a brief conclusion and explain future aspects of this work.

II. Related Works

In this section, we focus on some prominent works about overlapping community detection. Our main purpose is explaining the approaches that we consult in our comparative analysis. That is why; we shortly overview the general idea behind many algorithms but give the details about specific algorithms used in this work.

Overlapping community detection is widely seen as an expansion problem of local objects in the networks [7], [9], [10]. Many approaches are proposed using different types of seeds such as nodes, cliques or links. These approaches differs from each other by not only the types of seeds but also the strategy of expansion. For example the method OSLOM developed by Lancichinetti et al. [9] is using nodes as the seeds of communities. For the expansion, it takes into account statistically significance w.r.t global null model of the candidate communities. It not only finds overlapping structure but also hierarchical one. Moreover, one can detect the outliers or singleton communities with OSLOM. Another expansion method GCE is based on cliques [7]. GCE at first searches the maximum cliques and use them as the seeds of communities. It then tries to maximize a local fitness function proposed by Lancichinetti et al. [13] by using greedy optimization. Because this method does not search all cliques and expansion phase performed by greedy, it works fast.

The algorithm MOSES proposed in [10] uses links as seeds and expands them according to an objective function by using heuristic approximation. This objective function is based on the joint distribution over the set of communities, single connection probability of inner and outer of community. The authors claims that MOSES is successful for detecting highly overlapping structures. COPRA is another expansion algorithm using information propagation strategy [6]. Each node takes the most commonly appearing label among its neighbors synchronously at each expansion time interval. Algorithm continues until the labels do not change. This algorithm is faster than most of the other algorithms. However, there can be a significant difference of the results after two different running of COPRA at the same network.

Differently from previously described methods, EGO-BASED is not based on expansion but merging friendship groups of ego-centered networks [11]. This algorithm extracts at first ego-centered network of each node. It then detects the friendship groups in ego-centered networks as the connected components after removal of the ego node. At last, it merges the friendship groups if intersecting node number of two groups is one less than the size of one of those groups. This algorithm uses first level neighbors of the ego nodes when creating ego-centered network. Thus, it is possible to ignore big friendship groups. Furthermore, when merging groups, the criteria they propose can be too restricted to find highly similar ones.

A descriptive state-of-art work about overlapping community detection is given in [5]. There is also an extensive comparative analysis of all methods we explained above, except EGO-BASED, regarding their performance on artificial networks generated with different (1)overlapping node number and (2)number of communities that overlapping nodes can belong. According to the results of modified NMI [13], the performance of all algorithms decreases by the increase of the value of those two parameters. The authors conclude that *the detection in networks with high overlapping density and high overlapping diversity still has space for improvements*.

III. Method

The general view of our method *EMOC (ego based merged overlapping communities)* contains two main steps: (1) finding *k*-connected node groups of each ego and (2) merging detected groups to create the communities. In this section, we explain the concepts and the definitions related to those two steps. At first, we explain the preliminaries, then we detail the definition of *k*-connected node groups and explain how we find them, finally, we give the description of merging process and EMOC algorithm. As a first attempt, our proposal is designed for plain networks without any direction, weight or attributes assigned to nodes or links. However, it can easily be extended for directed and weighted networks.

A. Preliminaries

Given a plain network $G = (E,V)$, V is a set of *n nodes* and E is a set of *m links*. For each node i in V, its *ego-centered network at radius d_i, $G_i^{d_i}$*, is the subnetwork of G centered on node i and surrounded with d_i^{th} level neighbors of i. Note that if $d_i = 1$, all the nodes in $G_i^{d_i}$ are the direct neighbors of i. If d_i is equal to the diameter of G then, $G_i^{d_i}$ is as same as G. Two *paths* between two nodes i and j are *node independent*, a.k.a *node disjoint*, if they do not have any common internal nodes except i and j [14]. We say i and j are k-connected if there is at least $k \in [0,\infty[$ different node disjoint paths from i to j.

B. k-Connected Node Groups

Seeing real-world social environments from the perspective of each person, we might discover multiple cohesive groups that he belongs. Some of those groups may represent his family members, his coworkers, his social friends and so on. It is also possible that he contacts individually with some unique people without creating a group. Intuitively, we assume that the people

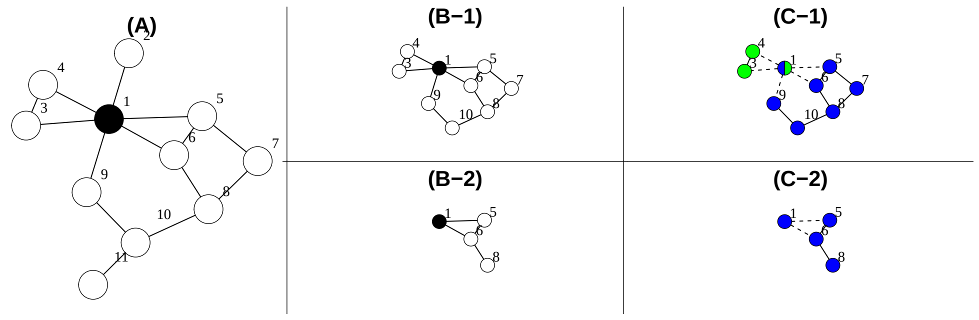

Fig. 1. (A) Ego-centered network of Node 1 at radius 3. Black node is ego and white nodes are its neighbors of radius 3. (B-1) Same ego-centered network after removing nodes which are not 2-connected to ego. (B-2) Same ego-centered network after removing nodes which are not 3-connected to ego.(C-1) 2-connected node groups in same ego-centered network. Each different color shows belonging to a different node group. Ego belongs to all groups. That is why, it has two colors.(C-2) 3-connected node group in same ego-centered network.

in the same group should directly connect with each other or should be easily accessible through other members. In network modeling, one way of pointing to these kinds of cohesive groups is looking for cliques or clique-like objects. However, it can be too restricted because everybody does not necessarily be connected with each other or the number of connections related to each person cannot be same with each other. Another way can be regarding only simple connectivity. But it can be too relaxed for ensuring the cohesiveness of the group. For instance, a large circular network is also connected but the density of connection is too low to be named as cohesive. That is why; we want to concentrate on the notion of being *k-connected* of the nodes.

We define an *ego-centered k-connected node group* $C_{ip} \subseteq V_i^{d_i}$ of ego i as a group of nodes in which each pairs of nodes are *k-connected*. Note that $\cup_p C_{ip}$ is equal to $V_i^{d_i}$ depending on the value of k. If k is at its lowest value, $\cup_p C_{ip} = V_i^{d_i}$. However, if k is at its maximal value, several nodes may not belong to any group. We want to underline that in an ego-centered k-connected node group, $\forall k > 1$, the nodes stay connected even if ego is removed from the network because they are all *k-connected* before the removal. To find such groups, we propose an algorithm which at first eliminate the nodes which are not *k-connected* with the ego. We then extract each connected components after removing ego.

Let us try to explain the idea of finding ego-centered node groups for different k values with an example. We represent in figure 1-(A), ego-centered network of node $i = 1$ (colored as black) at radius $d_i = 3$. Assume that we search 2-connected node groups. After checking the number of node disjoint paths, nodes $j = 2$ and $j = 11$ will directly be eliminated. So, those nodes will not be placed in any group. Note that although node $j = 2$ has a direct contact with ego, it will not appear in any ego-centered 2-connected group because of not connecting with any node else. We see in figure 1-(B-1), ego-centered network after elimination of not 2-connected nodes. As it can be seen, there are two different cyclic groups whose meeting point is ego. Removing ego results separation of those groups but each of them still stay connected inside. In figure 1-(C-1), we represent two 2-connected node groups found after removing ego with different colors. Ego belongs to both of them. That is why; we paint it with two colors. In case of looking for 3-connected node groups in the same ego-centered network, we obtain different results. After removing not 3-connected nodes with ego, nodes $5, 6, 8$ are remaining (shown in figure 1-(B-2)). Removal of ego makes them stay connected so there is only one node group including all the nodes remaining (shown in 1-(C-2)).

Algorithm 1 Finding Ego-centered k-Connected Node Groups

```
Require: $G_i^{d_i}$, $k$, $i$
Ensure: $C_i = C_{i1}, \dots, C_{ip}$
1: for $j \in V_i^{d_i}$ do
2:     if $nodeDisjointPathNumber(i, j) < k$ then
3:         $G_i^{d_i} = remove(G_i^{d_i}, j)$
4:     end if
5: end for
6: $G_i^{d_i} = remove(G_i^{d_i}, i)$
7: $C_i = extractConnectedComponents(G_i^{d_i})$
8: $C_i = insertEgoToAll(C_i, i)$
```

We give the pseudo code of finding ego-centered k-connected node groups in algorithm 1. The algorithm takes the minimum number of node disjoint paths (k), ego-centered network ($G_i^{d_i}$) and ego (i) itself as input. It first eliminates the nodes whose node disjoint path numbers to the ego are less than k (lines 1, 2, 3, 4 and 5). Here, function `nodeDisjointPathNumber(...)` in line 2 calculates the number of disjoint paths. We use push relabel max flow algorithm [15] for this function. The complexity of this step is $O(|V_i^{d_i}|^3)$. We calculate it for all nodes, so it makes $O(|V_i^{d_i}|^4)$. Functions `remove(...)` and `insertEgoToAll(...)`, given in lines 6 and 8 respectively, can be computed in constant time. Extracting connected components after removal of ego (given in line 7 as function `extractConnectedComponents(...)`) can be done with BFS. Running time of this step is $O(|V_i^{d_i}| + |E_i^{d_i}|)$. So overall running time of this algorithm is $O(|V_i^{d_i}|^4 + |V_i^{d_i}| + |E_i^{d_i}|) \sim O(|V_i^{d_i}|^4)$.

C. Ego Based Merged Overlapping Communities

We encounter same or very similar node groups related to different egos if they share many common neighbors. We consider similarities of different groups in terms of their common nodes to detect final community structure. Previously, that process is done by merging the groups that match all nodes but one node of smaller group [11]. However, one might ignore very similar groups if their size is high with this approach. One of the node groups might be relatively very similar with another, i.e. there might be significant numbers of nodes in common, but there can be more than one distinct node. That is why; here, we propose to take into account the rate of the similar part to non similar part of a node group S_1 to any other node group S_2 during the merging process. We consider the measure $sim(S_1,S_2) = |S_1 \cap S_2|/|S_1|$ for the similarity of two groups S_1 and S_2.

Algorithm 2 Merging Node Groups

```
Require: $S$, $threshold$
Ensure: $M$
1: $sort(S)$
2: for $i \in 1 \ldots |S|$ do
3:     for $j \in i \ldots |S|$ do
4:         if $sim(S_i,S_j) \geq threshold$ then
5:             $S_j = union(S_i,S_j)$
6:             $S_i = \emptyset$
7:         end if
8:     end for
9: end for
```

The pseudo code of merging process is given in algorithm 2. Algorithm takes the set S, whose members are the node groups found for whole network, and the minimum ratio of the similarity (*threshold*) as the input parameters. It outputs overlapping community structure M. It firstly sorts S according to the size of node groups in increasing order (line 1). Computing size of each group can be done in constant time. We apply simple radix sort which requires $O(|S|)$ where $|S|$ is the total number of node groups. Then, the procedure checks for each couple of node groups, S_i and S_j, if their similarity is greater than given threshold (line 4). Similarity can be computed in $O(min(|S_i|,|S_j|))$ with the usage of hash table for storing the elements of one of the node groups. In case of being sufficiently similar, these two groups are merged (line 5) and the groups S_i and S_j are updated. Merging phase can also be done in $O(min(|S_i|,|S_j|))$ with hash table. The algorithm continues until each couple of node groups is processed. The average size of a node group can be given $n/|S|$. So, the overall time complexity of merging phase is $O(|S| + 2 \times (n/|S|) \times |S|^2) \sim O(n \times |S|)$

The algorithm EMOC, at first extracts ego-centered network of each node i at radius d_i. This can be done in constant time by using the adjacency matrix representation of the graph. It then finds k-connected node groups related to each ego by using algorithm1 and creates the global set S of node groups over whole network. Finally, it applies algorithm 2 and creates the overlapping community structure. Total time complexity of these two steps is $O(|V_i^{d_i}|^4 + n \times |S|)$. We assume that $|V_i^{d_i}|^4 \gg n \times |S|$. It results $O(|V_i^{d_i}|^4)$. This complexity highly depends on the density of network and the chosen radius value d_i to create ego-centered networks. If network is sparse and $d_i = 1$, then, size of ego-centered network will be $\log n$ [11]. So, time complexity will be $O((\log n)^4)$. In case of dense network, or high values of d_i, complexity can be $O(n^4)$. Indeed, considering each node as an ego gives the

TABLE I. LFR NETWORK GENERATION PARAMETER VALUES

	Parameters	Values
1	μ	$\{0.1, 0.3, 0.5\}$
2	(c_{min}, c_{max})	$\{(5,25),(10,50),(20,100)\}$
3	O_n	$\{50,100,500\}$
4	O_m	$\{2,3,4,5,6,7,8,9,10\}$

opportunity to parallelize the computation. Thus, in practice, this time complexity can be reduced very much.

EMOC considers three parameters: (1) d_i, radius of ego-centered network to adjust the size of the communities. As this parameter can be constant for whole network, it can also change from one node to another depending on the topology of the network. (2) k, number of node disjoint paths, to regulate the cohesiveness of node groups and (3), *threshold*, to decide the minimum ratio of node groups similarity. Adjusting values of these parameters requires topological analysis of the studied network. Nevertheless, in the most basic form, one can set constant $d_i = 1$, $k = 2$ and $threshold = 0.8$ for considering first-level ego-centered network, minimum cohesiveness inside the groups and high similarity of the groups. Note that decreasing the value of *threshold* may result high overlapping of the communities.

IV. RESULTS

We conduct experiments on artificial and real-world networks to see the performance and the limits of EMOC. We generate artificial networks with predefined overlapping community structure by using LFR model [12]. The accuracy of EMOC is determined by normalized mutual information, NMI, measure. NMI is frequently used to assess the accuracy of estimated communities [16]. Recently, a modified version of NMI[1] for overlapping communities is proposed by Lancichinetti et al. [13]. As the traditional one, modified NMI takes 0 if two compared overlapping structure is totally dissimilar and it takes 1 if they are exactly same. Previously, the modified version is used for comparing the accuracy of the foremost overlapping community detection algorithms in [5]. We also perform a comparative analysis in this work.

We evaluate NMI results of EMOC with the results of 5 different overlapping algorithms: GCE[2] [7], OSLOM[3] [9], COPRA[4] [6], MOSES[5] [10] and EGO-BASED [11]. We implement EGO-BASED by considering its pseudo-code given in [11]. In this section, we at first concentrate on the experiments on artificial networks. We explain LFR model by giving the parameter values that we use and the results of the experiments respectively. We then focus on the experiments on real-world networks. At this part, we explain the networks we use and we interpret the results of EMOC on these networks.

A. Experiments on Artificial Networks

1) LFR Artificial Network Generator: LFR model allows generating random networks with power-law degree distribution, predefined community structure and power-law community size distribution [12]. This is the most realistic model in

[1]https://sites.google.com/site/andrealancichinetti/mutual
[2]https://sites.google.com/site/greedycliqueexpansion
[3]http://www.oslom.org
[4]http://www.cs.bris.ac.uk/ steve/networks/software/copra.html
[5]https://sites.google.com/site/aaronmcdaid/moses

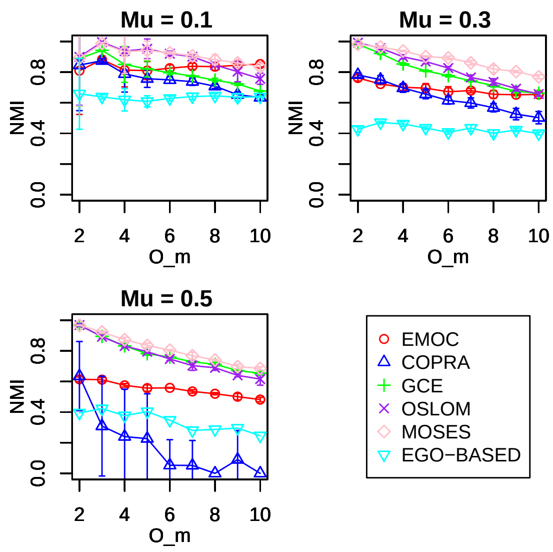

Fig. 2. NMI Result comparison of six algorithms for $(c_{min}, c_{max}) = (5,25)$, $O_n = 50$. Top-left plot is for well-separated communities ($\mu = 0.1$). Top-right plot is for medium-separated communities ($\mu = 0.3$). Bottom-left plot is for few-separated communities ($\mu = 0.5$)

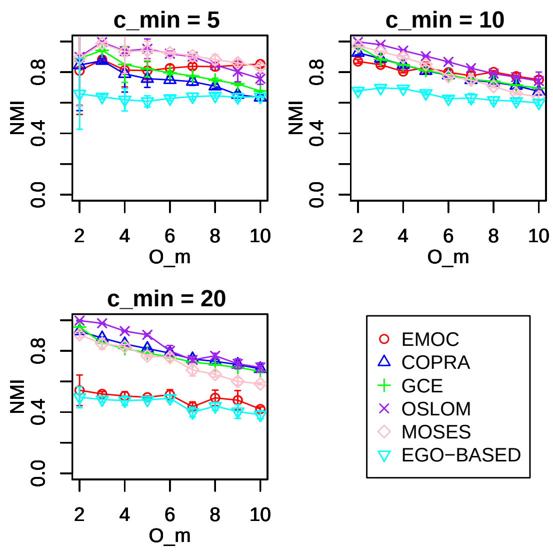

Fig. 3. NMI Result comparison of six algorithms for $\mu = 0.1$, $O_n = 50$. Top-left plot is for small communities ($(c_{min}, c_{max}) = (5,25)$). Top-right plot is for medium communities ($(c_{min}, c_{max}) = (10,50)$). Bottom-left plot is for large communities ($(c_{min}, c_{max}) = (20,100)$)

the literature [16]. One can change the topology of generated networks with various parameters, e.g. number of nodes n, desired average $\langle k \rangle$ and maximum degrees k_{max}, exponent γ for the degree distribution, exponent β for the community size distribution, c_{min} and c_{max} for minimum and maximum community sizes and mixing coefficient μ for the desired average proportion of outer community links. Hence, it provides a rich environment for evaluating the performance and the limits of community detection algorithms. It is previously used for this issue in many works [5], [16]. The version of LFR we consult at this work generates plain networks with predefined overlapping community structure. In this version, we can also control total number of nodes O_n which belong to more than one community and maximum number of communities O_m that a node can belong in overlapping structure.

LFR determines at first the possible number of inner links and membership numbers of each node by using μ, O_m, O_n, $\langle k \rangle$, k_{max} and γ and the possible sizes of communities with c_{min}, c_{max} and β. Once the basic network is generated by using configuration model [17], the assignment of the nodes to the communities is done by a rewiring process managing generation of a bipartite network whose two sides are the nodes and the communities of main network. In this bipartite network, each node will have as many links as its membership number and each community will have as many links as its size.

2) Results on Artificial Networks: In our experiments, we state the values of some parameters on the basis of previous works [5], [16]. Hence, we fix $n = 1000$, $\langle k \rangle = 10$, $k_{max} = 50$, $\gamma = 3$ and $\beta = 2$. The values of other parameters are given in table I. The increasing values of μ in table I allow us decreasing the level of community separation. We assume that we generate networks with well, medium and few separated communities for μ values given in table I respectively. The different values of the second parameter (c_{min}, c_{max}), minimum and maximum community size couple, in table I allow us generating small, medium and large communities respectively. We can adjust the overlapping level with O_n and O_m. We assume that we generate networks with small, medium and large numbers of overlapping nodes respectively for the values of O_n given in table I. The values of last parameter O_m allow us generating networks with increasing numbers, from 2 to 10, of overlapping communities that a node can belong. We mainly interpret the results of the algorithms performances according to their behavior to the increase of O_m. We generate 10 networks for the combination of each parameter values given in table I. We run all 6 algorithms on each network and consider the average performance on 10 networks. Among these algorithms, we run COPRA 10 times for each network to ensure its consistency. We set the three parameters of EMOC $d_i = 1$, $k = 2$ and $threshold = 0.8$. The explication we give in this section is valid for all the results we have found although we only present some examples of NMI-O_m plots.

In figure 2, we represent NMI results of the algorithms when we increase O_m for different μ values. For $\mu = 0.1$, all the algorithms take NMI > 0.8 (figure 2 top-left). When $O_m \leqslant 5$, OSLOM and MOSES are the most performing ones. GCE, EMOC and COPRA follow them. EGO-BASED seems the least performing one. We observe a few linear decrease of the performance with the increase of O_m. The two ego based methods are less sensitive to this fact. Among them, EMOC seems more performing than EGO-BASED. Considering the increase of μ values, we notice that especially the performance of EMOC, EGO-BASED and COPRA decreases. However, EMOC and EGO-BASED still can keep their performance stable to the increase of O_m. COPRA does not exhibit a robust behavior especially when $\mu = 0.5$ (figure 2 bottom-left).

We show NMI results of the algorithms according to different community sizes on different plots in figure 3. Apparently, for all algorithms, the easiest case is the case that the network has small communities whose sizes change between 5 and 25 (figure 3 top-left). When community sizes are between 10

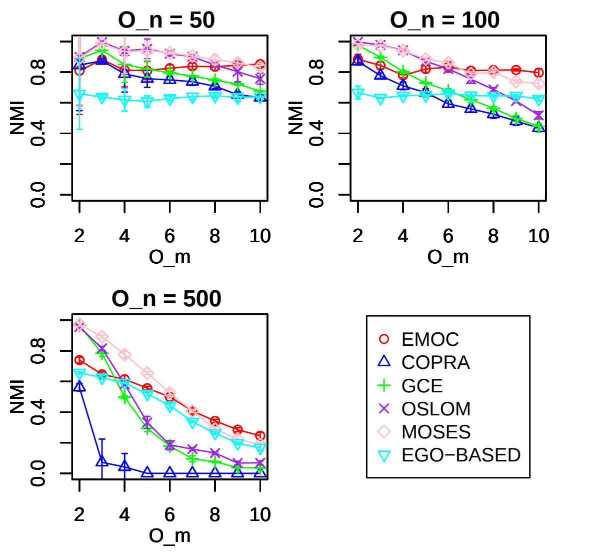

Fig. 4. NMI Result comparison of six algorithms for $\mu = 0.1$, $(c_{min}, c_{max}) = (5, 25)$. Top-left plot is when there is few number of nodes which belong to more than one community ($O_n = 50$). Top-right plot is when there is medium number of nodes which belong to more than one community ($O_n = 100$). Bottom-left plot is when there is large number of nodes which belong to more than one community($O_n = 500$)

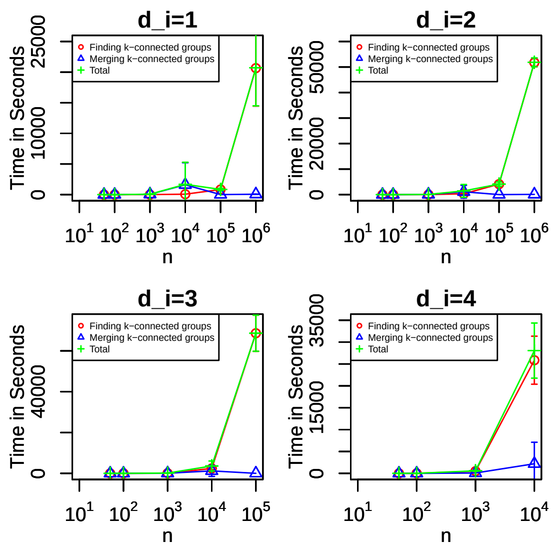

Fig. 5. Execution time of EMOC on different network sizes for different d_i values. Top-left, top-right, bottom-left and bottom-right plots correspond to the performance for $d_i = 1$, 2, 3 and 4 respectively. Given that the execution time span for larger networks are too much, they are not represented.

and 50, the algorithms performance is still good (figure 3 top-right). However, we observe a linear decrease for all algorithms except ego based ones with the increase of O_m. Two ego based methods keep a stable performance. Among them, EMOC results are as good as other algorithms while EGO-BASED is one step backward than them. Especially, when $O_m > 8$, EMOC and OSLOM perform the best.

In figure 4, we represent NMI results of the algorithms for networks generated with $O_n = 50$, 100 and 500 in plots top-left, top-right and bottom-left respectively. We see that the easiest case for all algorithms is $O_n = 50$. When the numbers of overlapping nodes increase to 100, we observe a visible linear decrease on the performance of all algorithms except ego based ones. EMOC's performance is as similar as the case of $O_n = 50$. Among the expansion based methods, MOSES exhibit better performance than the others even for high O_m values. It is claimed that MOSES is successful for detecting highly overlapping structures [10]. As seen in top-right plot, EMOC performs as well as MOSES for $O_m \geqslant 6$. Its performance is higher than MOSES for $O_m = 9$ and 10. In case that the half of the nodes in the network overlaps (figure 4 bottom-left), the decrease of the performance of all algorithms with the increase of O_m becomes more visible. The performance of the algorithms OSLOM, GCE and MOSES decrease logarithmically. Among them, MOSES has a smoother decreasing trend. COPRA has a very sudden decrease. Even for low O_m values, its performance is worse than the others. Two ego based methods have very similar performance trends and values.

By overall observation of the results for every parameter combination, we see that the performance of the algorithms GCE, OSLOM and MOSES are very similar and good when communities do not overlap very much. However, their performances are affected by overlapping density and diversity. In contrast to this fact, two ego based methods seems more stable than the others for changes on overlapping level. Among them EMOC exhibits better results than EGO-BASED in many cases. Especially, if the communities are well-separated and their sizes are small, EMOC results are as good as the most performing algorithms. Hence, comparing EMOC with EGO-BASED, we can say that the generalization we propose produces better results especially while finding small and well-separated communities. Nevertheless, a performance decrease is observed for large or mixed communities for EMOC.

We examine in detail EMOC communities for these types of networks. We notice that EMOC finds more and smaller communities than LFR reference for high μ and (c_{min}, c_{max}) values. One reason of this fact can be the amount of outer links causing cyclic connection with the nodes which are not in the same community. These are interpreted by EMOC as node disjoint paths. It causes to find more ego-centered k-connected node groups which do not have many common nodes between each other. We want to remind that in these experiments, we set $d_i = 1$, $k = 2$ and $threshold = 0.8$ for EMOC. We use static values of the parameters which are valid for all nodes. For finding large communities, increasing d_i or decreasing $threshold$ can be a solution. Or for the detection of non-separated communities setting higher k values can produce better results. A more sophisticated solution can be determining the parameter values for each node according to its topological properties. For instance, the most basic approach can be considering degree of the nodes. Or, a more complicated process inferring values of the parameters w.r.t combination of different topological measures such as degree, local transitivity or centrality can be used for adjusting the size and the cohesiveness of the communities.

3) Time Performance: In section III-C, we underline that time performance of EMOC depends on the density of network or the value of parameter d_i. To see these effects, we generate LFR networks with increasing n values. The

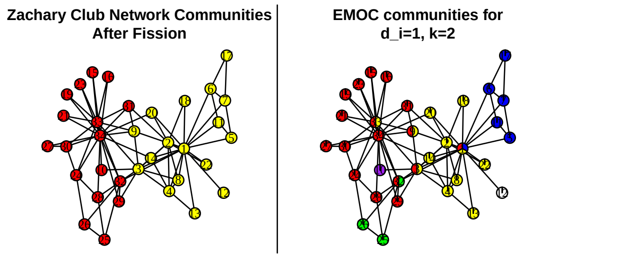
Zachary Club Network Communities After Fission
EMOC communities for d_i=1, k=2

Fig. 6. Zachary Karate Club Network. The different colors of nodes represent belonging to different communities. Nodes with multiple colors belong to multiple communities. Left and right plots represent ground-truth and EMOC communities for $d_i = 1$, $k = 2$ and $threshold = 0.8$ respectively.

values of the generation parameters are as follow: $n = \{5 \times 10^1, 10^2, 10^3, 10^4, 10^5, 10^6\}$, $\langle k \rangle = 5$, $k_{max} = 25$, $\gamma = 3$, $\beta = 2$, $c_{min} = 5$, $\mu = 0.1$, $O_n = 50$ and $O_m = 2$. We set the values of c_{max} according to the values of n. So, it takes 10, 25, 25, 250, 2500 and 25000 respectively. In figure 5, we see execution time of EMOC for the networks with increasing numbers of nodes in logarithmic plots. Each plot in the figure represents time performance for different d_i. Different colors of the curves symbolize the execution time of different steps or total time of EMOC. According to this, the larger the network, the higher the execution time for any d_i value. The execution time for the computation of merging process is ignorable comparing it with the computation of k-connected node groups. Almost all execution time of EMOC is spent for finding k-connected node groups for any n and d_i. For $n > 10^5$, execution time increases suddenly too much for $d_i = 1$ and 2. We observe a similar trend for $d_i = 3$ and 4 but for $n > 10^4$ and $n > 10^3$ respectively. Indeed, the analytic time complexity calculation is confirmed with experimental results. However, as we indicated before, EMOC is suitable for parallelization because of handling each node separately when finding k-connected node groups. This can be a solution for decreasing execution time.

B. Results on Real-World Networks

We apply our algorithm EMOC on two real-world networks. The first one is well-known Zachary karate club network [18]. This network is created by observing Zachary club members for 2 years. It shows the relations of 34 club members. Club members split into two groups because of political conflict between karate trainer John (node #34) and club president Mr. Hi (node #1). Thus there are two natural communities whose leaders are those two members. One can see these communities represented by different colors in left plot of figure 6. In the same study, the faction of each club member is also declared as strong or weak connection with one of the leaders or as no faction.

Running EMOC with $d_i = 1$, $k = 2$ and $threshold = 0.8$, we find 4 communities (shown in right plot of figure 6). Natural communities of Zachary network seem to be split by EMOC. Union of red and green communities of EMOC substantially correspond John's group. Likewise, yellow and blue communities of EMOC constitute Mr. Hi's group. There are 5 overlapping nodes (#1, 3, 9, 32, 33). Mr. Hi (node #1) belongs to three communities that 2 of them correspond to his group after split. We look at topological measures of the overlapping nodes and discover that except node #9, they have

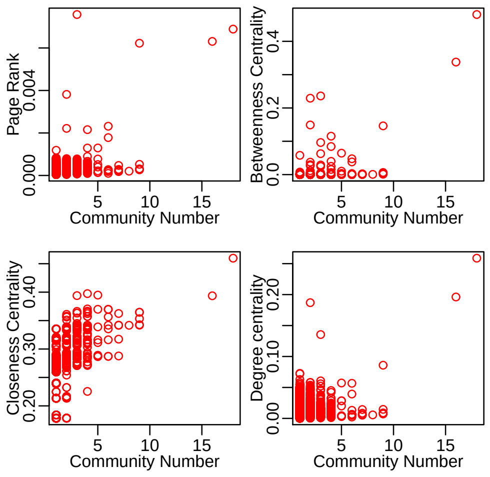

Fig. 7. Relation between nodal topological measures and the numbers of communities that Facebook users belong. Each red circle in the plots corresponds to the scores of one node. Top-left, top-right, bottom-left, bottom-right plots show the relation between community numbers and page rank, betweenness, closeness and degree centralities respectively

the highest betweenness scores in the network with node #34. As also seen in figure 6, they are lying in-between two groups or they have a central position in one of the groups. The most interesting result is about node #9. Its betweenness score is not as high as others. This node was a weak supporter of John before the split but he joined Mr. Hi's group afterwards [18]. EMOC puts him into two communities that each of them correspond to the groups of different leaders in reality. Two nodes (#10 and 12) are not placed into any community. Regarding their topological position, we see that node #12 is connected only with node #1 (Mr. Hi). He does not belong to any connected node group. That is why; he is not placed in any community by EMOC. Node #10 has only two connections. We discover that the faction of this node is declared as *none*. There are only 2 person (nodes #17 and 19) like him in Karate Club. Comparing them with community-less node #10, we see that nodes #17 and 19 have friends who are strong supporters of one of the group leaders. They are more under the influence of one of the factions. As a result, they can be more embedded in one of the communities. However, node #10's two friends are Mr. Hi's strong supporter and John himself. His idea about leaders is neutral and his friends are homogeneous. As a result, although he is placed in one of the groups in reality, we cannot claim that he is very embedded there. Briefly, it is seen that EMOC finds consistent communities with real groups in Zachary Karate network. Overlapping nodes have important topological situation. Community-less nodes can be either non-effective or easily effected by other people.

Second real-world network we deal with is Facebook Network[6] [19]. This network is a combination of 10 ego-centered networks that each of them includes the social circles of ten different Facebook users. There are 4039 nodes corresponding 10 ego and their friends and 88234 links representing the friendship relation of them. This network is ego-centralized by its construction. We apply EMOC with same parameter values as we used for Zachary karate club network. As a result, we

[6]http://snap.stanford.edu/data/egonets-Facebook.html

obtain 37 communities and 77 community-less nodes. All of the community-less nodes are connected only with one of ten egos. There are 923 overlapping ($\sim$ %23 of all) and 3039 non-overlapping ($\sim$ %75 of all) nodes. The most overlapping node belongs to 18 communities. It corresponds to one of 10 egos. The range of community sizes is between 3 and 1075.

We examine a possible relation between numbers of communities that a node belongs and topological properties. For this reason, we represent in figure 7 the scores of page rank, betweenness, closeness and degree centralities with the numbers of overlapping communities for each node. As it is case for Zachary karate network, in Facebook, the most overlapping nodes (#108 and 1685, points at the top-right corners of each plot)are the most central ones. Their page rank score is also very high. Those nodes correspond to two egos having hub position in the network. Observing the plots, we also notice that some nodes belonging to low numbers of community might have high page rank scores. Looking at them in more detail, we see that those nodes are not central themselves but they have direct connection with important hubs. As a result, their page rank score is high because of knowing important people.

Considering two real-world network experiments, we observe that EMOC in general finds small communities. Some of them corresponds to highly interconnected node groups which do not have relation between each other except connecting to the same hub. In general the most overlapping nodes are those hubs. Other highly overlapping nodes might be bridges connecting different communities. The community-less nodes might be different types of outliers such as the people having unexpected behavior or people who have no more connections than one in whole network.

V. Conclusion

In this work, we define a k-connected node group as a cohesive groups in which every couple of nodes have at least k different node disjoint paths between them. We concentrate on k-connected groups around each ego in the network. Based on merging different k-connected groups according to their similarity ratio, we define overlapping communities. We propose an algorithm EMOC to find such communities and evaluate its accuracy on the artificial networks by comparing its results with 5 foremost overlapping algorithms. The results show us that EMOC is very performing to find small and well-separated communities. Those types of communities might be family members or very close friend groups in reality. Its performance does not affected by the changing amount of overlapping nodes or communities. Hence, it can be used for detecting communities in any overlapping level. We also show time performance of EMOC. The results confirm our analytic complexity analysis. It is seen that EMOC's performance is affected by network size and radius of ego-centered network. As it handles each node separately, EMOC is convenient for parallelization. Thus, its performance can be ameliorated by this way. Applying EMOC on Zachary karate and Facebook networks, we see that it can find contextually accurate communities. Overlapping nodes corresponds to hubs which are meeting points of different node groups or bridges binding different communities. The community-less nodes are different types of outliers.

Some prominent perspectives appearing from this study can be listed as examining the roles of different parameters of EMOC on the topology of detected communities, applying EMOC on different types of real-world networks and interpreting the results, increasing time performance by algorithm parallelization and developing a strategy to automatically and dynamically determine EMOC parameter values according to the topological positions of the nodes. We also want to modify EMOC for weighted and directed networks. Modifications for directed networks can easily be done by considering link directions at the ego-centered network creation and finding the numbers of node disjoint paths. For weighted networks, it is possible to put an additional criterion for node disjoint path search or to merging strategy.

References

[1] S. Fortunato, "Community detection in graphs," *Phys. Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[2] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM*, 2013, pp. 1151–1156.

[3] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *PNAS*, vol. 105, no. 4, p. 1118, 2008.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *JSTAT Mech.*, p. P10008, 2008.

[5] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 43:1–43:35, Aug. 2013.

[6] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, p. 103018, 2010.

[7] C. Lee, F. Reid, A. McDaid, and H. Neil, "Detecting highly overlapping community structure by greedy clique expansion," in *SNA-KDD10*, 2010.

[8] F. Reid, A. McDaid, and N. Hurley, "Partitioning breaks communities," in *ASONAM*, 2011, pp. 102–109.

[9] A. Lancichinetti, F. Radicchi, J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PLoS ONE*, vol. 6, no. 4, p. e18961, 2011.

[10] A. McDaid and N. Hurley, "Detecting highly overlapping communities with model-based overlapping seed expansion," in *ASONAM*, 2010, pp. 112–119.

[11] B. Rees and K. Gallagher, "Overlapping community detection by collective friendship group inference," in *ASONAM*, 2010, pp. 375–379.

[12] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E*, vol. 80, p. 016118, Jul 2009.

[13] A. Lancichinetti, S. Fortunato, and J. Kertsz, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, p. 033015, 2009.

[14] D. R. White and F. Harary, "The cohesiveness of blocks in social networks: Node connectivity and conditional density," *Sociological Methodology*, vol. 31, no. 1, pp. 305–359, 2001.

[15] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *J. ACM*, vol. 35, no. 4, pp. 921–940, 1988.

[16] G. Orman and V. Labatut, "A comparison of community detection algorithms on artificial networks," in *DS*, 2009, vol. 5808, pp. 242–256.

[17] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence," *Random Structures and Algorithms*, vol. 6, no. 2/3, pp. 161–179, 1995.

[18] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977.

[19] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks." in *NIPS*, 2012, pp. 548–556.