Kybernetika

Pelin Çetin; Sahin Emrah Amrahov

A New overlapping community detection algorithm based on similarity of neighbors in complex networks

Kybernetika, Vol. 58 (2022), No. 2, 277–300

Persistent URL: http://dml.cz/dmlcz/150468

Terms of use:

© Institute of Information Theory and Automation AS CR, 2022

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.

This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* http://dml.cz

KYBERNETIKA — VOLUME *58* (2022), NUMBER 2, PAGES 277–300

A NEW OVERLAPPING COMMUNITY DETECTION ALGORITHM BASED ON SIMILARITY OF NEIGHBORS IN COMPLEX NETWORKS

Pelin Çetin and Şahin Emrah Amrahov

Community detection algorithms help us improve the management of complex networks and provide a clean sight of them. We can encounter complex networks in various fields such as social media, bioinformatics, recommendation systems, and search engines. As the definition of the community changes based on the problem considered, there is no algorithm that works universally for all kinds of data and network structures. Communities can be disjointed such that each member is in at most one community or overlapping such that every member is in at least one community. In this study, we examine the problem of finding overlapping communities in complex networks and propose a new algorithm based on the similarity of neighbors. This algorithm runs in $O(m\,lg\,m)$ running time in the complex network containing m number of relationships. To compare our algorithm with existing ones, we select the most successful four algorithms from the Community Detection library (CDlib) by eliminating the algorithms that require prior knowledge, are unstable, and are time-consuming. We evaluate the successes of the proposed algorithm and the selected algorithms using various known metrics such as modularity, F-score, and Normalized Mutual Information. In addition, we adapt the coverage metric defined for disjoint communities to overlapping communities and also make comparisons with this metric. We also test all of the algorithms on small graphs of real communities. The experimental results show that the proposed algorithm is successful in finding overlapping communities.

Keywords: overlapping community detection, complex networks, graph approach, similarity approach, community metrics

Classification: 94C15

1. INTRODUCTION

Nowadays, people have to cope with the growing data in every field with the growth of technology usage. Even just using the Internet, we produce numerous data every minute. Also, we can store big data and access it from miles away. As a result of these, data management is getting harder, and there is a need for categorization. Grouping data increases manageability and helps to understand the main structure. It allows sharing information only with the related ones in social networks, making better suggestions

DOI: 10.14736/kyb-2022-2-0277

in recommendation systems, finding related topics in search engines, finding similar proteins in drug discovery, etc.

In the literature, both clustering and community detection algorithms are used to group objects. Community detection is a special case of clustering, which is peculiar to networks. While clustering uses attributes to group similar objects, community detection uses relations between the objects. Since community detection algorithms use structural properties of the networks, they produce more general solutions for many domains.

There are two main approaches for community detection; mathematical and heuristic. Mathematical methods define the community and detect these structures. Heuristic methods assign objects to a community with the most similar one. Since defining a community is hard because of having different structures and strict rules, the heuristic approach is more common in the literature.

It is not always possible to assign each object to a community that is the most appropriate. The reason for that, sometimes, there can be more than one suitable community for the object. For example, a person can be in both school and business communities on a social network. In a product network, a product can have more than one category. A shaving machine can be in both beard and hair categories, or a robot cleaner can be in both vacuum cleaner and mop class, etc. In search engines, a word can have more than one meaning. Because of this, overlapping community detection algorithms are proposed. These algorithms can help us see multiple solutions together and hidden structures in the networks.

In this study, we propose an algorithm that can be used to solve different problems from various domains and help to see hidden structures. We create six small test networks divided into natural communities and see if the algorithms find them correctly. We measure the modularity, normalized mutual information, and F-Scores of our algorithm and compare them with the results of some well-known algorithms. However, how well these metrics measure the success of community detection algorithms is a controversial issue, as many authors [4, 7, 39, 45] have identified. In this study, we adapt the coverage metric used for disjoint communities to overlapping communities. We test both the proposed algorithm on benchmark and synthetic datasets with a small and large number of vertices. Our contributions are:

- We proposed a new overlapping community detection algorithm that can detect different community structures, where most algorithms like PercoMVC and LPANNI do not work. In addition, it is suitable to work on various networks such as protein-protein, social, books, and so on.
- The proposed algorithm does not require prior knowledge such as the number of communities, directions of relations, weights of connections, etc.
- We show that our algorithm works better than the existing community detection algorithms by using evaluation metrics and test networks.

The rest of the paper is organized as follows: In the next section, we summarize the studies on overlapping community detection. In section (3), we give information about evaluation metrics. The details of the proposed algorithm are given in section (4). We compare the results in section (5). Finally, in section (6), we discuss our results and define the conclusion of the study respectively.

2. RELATED WORK

The overlapping community detection problem has been studied for many years, and various methods have been applied. As far as we know, the first overlapping community detection algorithm is Clique Percolation Method (CPM) which was proposed by Palla et al. [43] in 2005. CPM uses k-cliques on the network and merges them if they share k-1 nodes. It works on connected networks. Then, working on weighted networks [12, 18], faster [44] or improved [52] versions are proposed. The main disadvantage of the CPM method is that it has a strict definition of the community. It is possible to use more flexible structures such as k-clans, k-clubs, k-plex, and k-core instead of the clique. However, k must be determined beforehand, and this method is not suitable for sparse networks [11].

In 2007, Zhang et al. used Non-negative Matrix Factorization (NMF) to detect overlapping communities [57]. This method uses a local modularity function to determine the number of communities. Then, Bayesian Non-negative Matrix Factorization (BNMF) is proposed [46] that improves the performance of NMF computationally. In another study, the performance of BNMF is improved by using three factors [58]. One of the latest studies uses this method on signed networks [24]. Another one uses this method and combines structural and semantic features to propose a hybrid method [47]. The other one uses this method with Bayesian Affiliation Preference [13]. In general, matrix factorization methods are used in sparse networks. Since networks have complex structures and detecting communities is time-consuming, this method is not commonly used in the literature.

The Label Propagation Algorithm (LPA) is another algorithm that is commonly used in the literature. The Speaker-Listener Label Propagation Algorithm (SLPA) [55] is an extended version of the label propagation algorithm that allows each node to have multiple labels. SL3PA is an improved and multi-thread version of SLPA. In another version of LPA [35], labels are updated in ascending order of node importance, and a historical label selection strategy is used. Another one [59] uses a node importance contribution matrix for similarities between nodes to detect overlapping communities. The DEMON algorithm [10] also uses LPA. This algorithm lets each node choose their communities by voting and combines the votes to find overlapping communities. In the last years, LPA for highly mixed networks [54], LPA-X (LPA with X-means algorithm) [16], multi-threading LPA [38], LPA with the belonging function [5], LPA with density peak clustering algorithm [31], LPA with node importance [28], and LPA with deepwalk [56] is proposed. However, LPA suffers from instability. Since it uses randomness, it could produce different communities in each run.

Variants of crisp community detection algorithms such as Newman-Girvan and Label Propagation are also used in overlapping community detection. Gregory [19] proposes CONGA, an overlapping version of the Newman-Girvan algorithm. He splits nodes or removes edges according to betweenness values. Then, he uses local betweenness and proposes CONGO, the optimized version of CONGA [20]. He also proposes an extension of the Label Propagation algorithm (COPRA) and determines the belonging coefficients of nodes by using the average coefficients of all neighbors in this version [21]. In another combination of Label Propagation and Girvan-Newman algorithms, Joghan et al. [26] determine the weights of edges by using edge-betweenness, then weighted LPA is applied.

However, calculating edge betweenness is time-consuming and is not very suitable for complex networks.

In 2009, Lancichinetti et al. [29] proposed the LFM algorithm to find both overlapping communities and hierarchical structures. They use random seeding and expand the communities by using local optimization of a fitness function. Lee et al. [30] used maximal cliques instead of random seed selection and proposed Greedy Clique Expansion (GCE) algorithm. Choumane et al. [9] find core nodes using the highest edge neighborhood overlap value, then find the communities by expanding them. Aghaalizadeh et al. [1] also use the seeding technique. They select important nodes and expand them using similarity until all nodes belong to a community. Agrawal and Patel al. [2] propose a hybrid method. They group nodes by directly connected, indirectly connected, and disconnected to find structural similarity. Then they use context similarity. After combining these two similarities, they applied the k-medoids algorithm. In our studies, we observe that applying an expanding technique with a similarity metric is effective. However, using the seeding technique requires the determination of the number of seeds. Because of that, if the number of the communities is unknown, using the seeding technique can cause to detection of more communities than desired.

In 2010, Ahn et al. [3] used links instead of nodes to detect overlapping communities. They use hierarchical clustering with Jaccard similarity between links. Ma et al. [37] also propose a loop edge deletion algorithm and structural clustering method. Lim [33] et al. transform node graph to link graph and use structural properties of a graph with link similarity in the LinkSCAN algorithm. Kim et al. [27] proposed the LinkBlackHole algorithm that uses LinkSCAN's ability to detect overlapping communities and BlackHole's robustness to mixing between communities. Jin et al. [25] developed a model which takes into consideration both node and link communities. The parameters of the model are learned by using an NMF, and the communities are determined by using greedy optimization. The number of communities is determined by using consensus clustering. Then, Sun et al. [49] propose the linked-based LPA. Lierde et al. [32] extended the normalized cut method to detect overlapping communities. They used a spectral clustering algorithm to solve the cut minimization problem. Their algorithm can use the prior knowledge of the number of the communities or get this data from node centrality. They also proposed a hierarchical version of the algorithm for an unknown number of clusters. Gupta and Kumar [23] use granular information of links and the rough set theory for overlapping community detection. However, link-based methods additionally require a graph transformation step to generate node communities from the edge communities.

There are also statistical approaches such as the Bayesian and Stochastic Block Model. Gopalan and Blei [17] developed a Bayesian-based model which assumes that there are k communities, and each node has an association with these communities by probability. They use community indicators that point to one of the k communities. For each pair of nodes, if indicators point to the same community, then the algorithm connects the nodes with high probability. Zhou et al. [59] propose a Mixed-Membership Stochastic Block Model and take into account the influence of the nodes. However, these methods require high computational time.

The PageRank algorithm is also used for community detection problems. Wang et al.

[53] use the PageRank algorithm to evaluate nodes and use the location data of nodes to detect overlapping communities. They represent communities by peak nodes, internal nodes by slope nodes, and overlapping nodes by valley nodes. Whang et al. [51] select seeds and then expand them based on the Page Rank clustering score. All neighborhoods are taken into consideration by modifying seeds to represent all neighborhoods of themselves. Gao et al. [15] improve the accuracy of the PageRank algorithm by keeping the walker away from the irrelevant communities. In one of the latest studies that use random walk, Guo et al. propose an overlapping community detection algorithm based on network representation learning. They use a node-centrality-based walk strategy and design different community-aware random walk strategies for high-degree and low-degree nodes [22]. One disadvantage of this method is that they are slow.

Having these approaches, expansion methods are produced. Wang et al. [50] use an approach that starts from central nodes and finds communities by expanding them. First of all, they calculate structural similarities. Then, the central nodes are detected by sorting these similarities in descending order. Their algorithm looks at all neighbors of each central node and determines to add a node to the community of the central node by calculating density. When all neighbors are evaluated, it terminates. They use local search for expansion. Long [34] uses core nodes and applies an expanding procedure. Firstly, he computes the edge density for each node. Secondly, skeleton edges are found by using threshold parameters, then core and margin nodes are detected. Thirdly, overlapping nodes are selected among core nodes and replicated. Then the belonging degrees of margin nodes are calculated, and they are merged with the core nodes having the highest belonging coefficient. Finally, the replicated nodes which belong to the same community are merged. However, a threshold value is required in this method, and determining the threshold value is another problem. Similarly, Ma et al. [36] proposed an algorithm that first finds the most influential nodes and then detects overlapping communities using the seeding technique.

In this study, we also apply an expansion method. Unlike the existing methods, we do not have a central node selection stage. Instead of evaluating neighbor nodes, we sort all edges in the network and merge the nodes connected by an edge. Additionally, we determine the merging process by using our criteria. In our method, we use similarity scores, and the proposed method does not require a threshold value. The details of the proposed algorithm are given in section (4).

3. KNOWN AND PROPOSED EVALUATION METRICS

There are internal and external metrics to measure the success of the community detection algorithms. If there is no ground-truth data, then internal metrics are used. Some of the internal metrics are modularity, conductance, and coverage. External metrics such as F-Score, Normalized Mutual Information (NMI), and purity are used when the real communities are known.

3.1. F-score

F-score is calculated by using precision and recall values. Precision is the fraction of the true positives that are predicted positive. Recall is the rate of the true positives to the

actual positive values. F-score given Formula (1) provides a balanced result between precision and recall.

$$F\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{1}$$

3.2. Normalized mutual information (NMI)

Normalized Mutual Information is the amount of information that can be extracted by knowing another one. If $I(X,Y)$ denotes the mutual information of jointly random variables X and Y, $H(X)$ and $H(Y)$ are the entropies, then NMI is calculated by Formula (2).

$$NMI(X,Y) = \frac{H(X) + H(Y) - H(X,Y)}{\frac{H(X)+H(Y)}{2}}. \tag{2}$$

There are also extensions of NMI in literature. One of the most known of them is proposed by Lancichinetti and Fortuno [29]. They modified the NMI into Formula (3) to work with overlapping communities.

$$NMI_{LFK}(X,Y) = 1 - \frac{1}{2}\left(\frac{H(X|Y)}{H(X)} + \frac{H(Y|X)}{H(Y)}\right). \tag{3}$$

McDaid et al. [40] indicate that the normalization factor in NMI_{LFK} causes overestimating of the similarity of two clusters. Because of that, they used their normalization and modified NMI as in Formula (4).

$$NMI_{MGH}(X,Y) = \frac{I(X|Y)}{max(H(X),H(Y))}. \tag{4}$$

We use F-Score and NMI_{MGH} in this study if we have real communities. We use modularity and coverage metrics for evaluation when the real communities are unknown.

3.3. Modularity

Newman and Girvan [42] introduced modularity in 2004. This metric is based on the idea that a random graph is not likely to have a community structure. The difference between detected partitions and a random graph with the same number of vertices and degrees is measured with this metric. In Formula (5), A is the adjacency matrix of the graph, which stores the actual links between vertices, and m is the number of edges. The expected connection between nodes i and j with degree k_i and k_j is calculated by the multiplication of probabilities $p_i = k_i/2m$ and $p_j = k_j/2m$. $\delta(C_i, C_j)$ is a function that produces 1 if community C_i of vertex i and community C_j of vertex j are the same, 0 otherwise. However, this modularity is not suitable for overlapping communities.

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(C_i, C_j). \tag{5}$$

Nepusz et al. [41] used a similarity measure s_{ij} which is the sum of the multiplication of membership degrees instead of $\delta(C_i, C_j)$ function in the original modularity function. In Formula (6), $s_{ij} = \sum_{c=1}^{k} a_{ic}a_{jc}$ where a_{ic} is the belonging degree of vertex i to community c, and k is the number of communities.

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] s_{ij}. \tag{6}$$

Shen et al. [48] defined a generalized version of the modularity, which works for both disjoint and overlapping communities. They used $1/(O_i O_j)$ instead of $\delta(C_i, C_j)$. O_i in Formula (7) is the number of communities of vertex i.

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \frac{1}{O_i O_j}. \tag{7}$$

Chen et al. [8] proposed another generalized version of modularity, Formula (8). They defined a function $f(a_{ic}, a_{jc})$ for the membership of vertices i and j to community c. This function can be equal to multiplication, average, maximum, and so on of the belonging degrees.

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] f(a_{ic}, a_{jc}). \tag{8}$$

Modularity is a useful metric for the networks where ground-truth data is unknown. In this study, we use Shen Modularity.

3.4. Coverage

Coverage is the ratio of the number of intra-community edges to the total number of edges in a graph [14]. Intra-community edges are defined as the edges which connect two vertices in the same community. This definition of the coverage metric is used for disjoint communities. In the case of overlapping communities, intra-community edges are not defined. The reason is that two neighbors can be both in the same community and in different communities at the same time. We slightly change the definition of intra-community edges and adapt the coverage metric to overlapping communities.

Let a graph $G = (V, E)$ be given. Suppose that after a community detection algorithm is applied, this graph is split into $k >= 1$ disjoint or overlapping communities $C_1, C_2, \ldots C_k$. The formula of the overlapping version of the coverage is given in Formula (9). For an edge $(u, v) \in E$, if there is a community C_j such that $u \in C_j$ and $v \in C_j$, we call the edge (u, v) an intra-community edge. The set of all intra-community edges in G is denoted by E_{in}.

$$Coverage = \begin{cases} 0, & \text{if } k = 1 \\ \frac{|E_{in}|}{|E|}, & \text{if } k > 1 \end{cases} \tag{9}$$

Some algorithms can put all vertices into one community and generate only one community for some input graphs. In order to prevent this situation, we define case $k = 1$ as an exception in Formula (9).

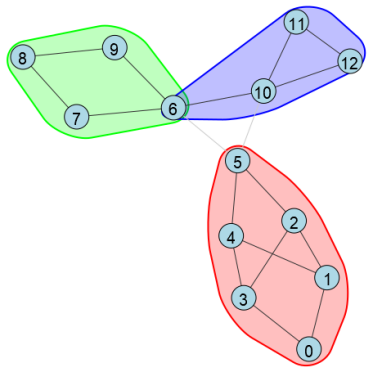


Fig. 1: Example of the overlapping version of the coverage.

We give an example in Figure 1 for this version of coverage. In Figure 1, all edges except edges (5, 6) and (5, 10) are intra-community edges. For example, (6, 10) is an intra-community edge because vertices 6 and 10 both belong to the blue community. For this example, $|E_{in}| = 16$, and $|E| = 18$. The value of the coverage is found 0.89 by using Formula (9).

4. THE PROPOSED ALGORITHM

In this section, we explain our new algorithm for detecting overlapping communities. The input of the proposed algorithm is an undirected and unweighted simple graph $G = (V, E)$. For example, an undirected graph can represent relationships between people in a social network. The vertices of this graph represent people, and the edges represent the relationships between them. The output of the algorithm is a list of overlapping communities C. For a social network, communities mean sub-groups (sub-graphs) formed by people who are more in contact with each other.

The proposed algorithm consists of three stages, and they are explained as follows. The first stage is the preparation and initialization stage. At this stage, we add each vertex to its neighbors' list. Then, we calculate the similarity of all adjacent vertices connected by an edge in our input graph using cosine similarity. Adding each vertex to its neighbors' list allows us to increase the success of cosine similarity since it makes the similarity score of two vertices more than zero if there is no common neighbor between two vertices, but they are connected. Then, we assign the calculated similarities as weights to the edges, which connect the vertices. That is how we transform an initially unweighted graph G into a weighted graph. We create a community consisting of only itself for each vertex and obtain the initial list of communities C. We complete the first stage by sorting edges of the graph G in descending order of weights.

The second stage is the generation of the list of communities C. In this stage, we perform the following operations starting from the first edge in the ordered list until all edges are passed. If both of the vertices connected by the current edge are in single-

element communities consisting of only themselves, we merge these two communities into a single one. Otherwise, we check if there is a community where these vertices coexist. If such a community exists, we go to the next edge. If there is no such community, we calculate which of these vertices would be more advantageous to put to the other's community according to the criteria we explain below. For edge (u, v), if putting vertex u into community C_v of vertex v is more advantageous according to our criterion, we expand C_v by adding u. Additionally, if there is community $\{u\}$ in C, we delete it from C. We emphasize that we add only the vertex itself, not all vertices in its community. In this way, we evaluate each vertex one by one. Once all edges are passed, the second stage completes, and we have a list of communities C.

The third stage of the algorithm is the stage of improving the community list C. In this stage, we sort the communities in C according to the number of their elements in descending order. Then, starting from the second community in the list, we do the following operations until all communities are traversed. We look at the intersection of the current community and a previous community by order. If the number of elements in the intersection is more than half of the number of elements of the current community, we merge these two communities. As a special case, if there are only two elements in the current community and one of them is in the intersection, we also apply the merging process.

Now let us explain the criteria that we used for evaluation and how we get the more advantageous situation based on it. We define the criterion for an edge (u, v) by Formula (10), where n_u denotes the set of neighbors of vertex u, and C_v represents a community that contains vertex v. Since the $CN(u, v)$ value shows the number of u's neighbors in a community of vertex v, we express the criterion with CN, which are the initials of the expression "Common Neighbors".

$$CN(u, v) = \max_{C_v \in C} |C_v \cap n_u| - \tag{10}$$

We evaluate each possible combination with our criteria. First, we determine the highest value of $CN(u, v)$ and community C_v^*, which provides that value according to Formula (10). Next, we calculate the value of $CN(v, u)$ for edge (v, u). Let C_u^* be the community that provides this highest value. If $CN(u,v) > CN(v,u)$, then adding vertex u to community C_v^* is more advantageous. If $CN(u,v) < CN(v,u)$, it is more advantageous to add vertex v to community C_u^*. In the case of $CN(u,v) = CN(v,u)$, we look at the degrees of u and v and put the vertex with a lower degree into the other's community. In the case of equality of degrees, we add vertex v to community C_u^*.

Let us show how the proposed algorithm finds the communities for the graph in Figure 2. First, for all 18 edges, we calculate their weights by using cosine similarity. Then, we evaluate these 18 edges in descending order of weights starting from the initial situation that each vertex forms a separate community. The results after each step are given in Figure 3.

In the beginning, the edge with the highest degree is $(11, 12)$, and vertices 11 and 12 both belong to communities with only one element. Because of that, the algorithm merges them in step 1. In step 2, we evaluate the second edge with the next highest weight $(10, 11)$ using our criteria. Since vertex 10 has more neighbors in the community

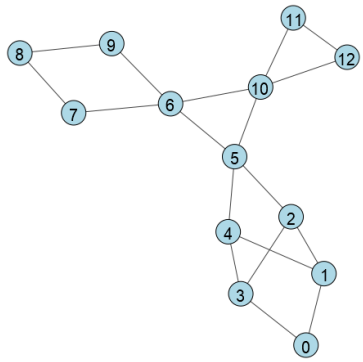


Fig. 2: An example graph.

of vertex 11, vertex 10 is added to the community of vertex 11. In step 3, since vertices connected by edge $(10, 12)$ are already in the same community, no changes are made. In step 5, since $CN(u, v)$ is equal to $CN(v, u)$, and the degrees of these vertices are equal, vertex 9 is added to the community of vertex 8. In step 10, $CN(u, v)$ and $CN(v, u)$ are also equal, but the degree of vertex 0 is higher than the degree of vertex 3. Therefore, vertex 3 is added to the community of vertex 0.

Step	Edge (u, v)	Weight	CN(u, v)	CN(v, u)	Degrees of u and v	Communities
0	-	-	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}, {11}, {12}}
1	(11, 12)	1.00	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}, {11, 12}}
2	(10, 11)	0.78	2	1	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10, 11, 12}}
3	(10, 12)	0.78	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10, 11, 12}}
4	(7, 8)	0.67	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7, 8}, {9}, {10, 11, 12}}
5	(8, 9)	0.67	1	1	equal	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7, 8, 9}, {10, 11, 12}}
6	(5, 6)	0.60	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5, 6}, {7, 8, 9}, {10, 11, 12}}
7	(5, 10)	0.60	1	2	-	{{0}, {1}, {2}, {3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
8	(6, 10)	0.60	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
9	(0, 1)	0.58	-	-	-	{{0, 1}, {2}, {3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
10	(0, 3)	0.58	1	1	deg(v) is higher	{{0, 1}, {2}, {0, 3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
11	(6, 7)	0.52	2	1	-	{{0, 1}, {2}, {0, 3}, {4}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
12	(6, 9)	0.52	-	-	-	{{0, 1}, {2}, {0, 3}, {4}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
13	(1, 2)	0.50	1	1	equal	{{0, 1, 2}, {0, 3}, {4}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
14	(1, 4)	0.50	1	1	equal	{{0, 1, 2, 4}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
15	(2, 3)	0.50	1	3	-	{{0, 1, 2, 3, 4}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
16	(3, 4)	0.50	-	-	-	{{0, 1, 2, 3, 4}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
17	(2, 5)	0.45	1	2	-	{{0, 1, 2, 3, 4, 5}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
18	(4, 5)	0.45	-	-	-	{{0, 1, 2, 3, 4, 5}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}

Fig. 3: Steps of the second stage of the proposed algorithm.

After step 18, list C is obtained. When we sort this list in descending order according to the numbers of elements in the communities, we get $C = \{\{0, 1, 2, 3, 4, 5\}, \{6, 7, 8, 9\}, \{5, 6, 10\}, \{10, 11, 12\}, \{0, 3\}\}$. We start with the second community $\{6, 7, 8, 9\}$. Community $\{0, 1, 3, 4, 5\}$ is the only community before $\{6, 7, 8, 9\}$. Since the intersection of these communities is an empty set, we go to the next community without making any changes. The next community is $\{5, 6, 10\}$. The intersection of $\{5, 6, 10\}$ and the previ-

ous community $\{6, 7, 8, 9\}$ has only one element 6. Since this number is less than half of the number of elements in community $\{5, 6, 10\}$ (i. e., $1 < 3/2$), there is no merging process. The intersection of $\{5, 6, 10\}$ and community $\{1, 2, 3, 4, 5\}$ has only one element (5), and the merging condition is not met. The next community $\{10, 11, 12\}$ has an intersection with only $\{5, 6, 10\}$ from the previous ones. Since this intersection has one element, the merging condition is also not met. Finally, the intersections of community $\{0, 3\}$ and previous communities are examined. Only $\{0, 1, 2, 3, 4, 5, 6\}$ has an intersection with this community, and this intersection has 2 elements. Accordingly, these two communities are merged. As a result, the communities found by the algorithm for the analyzed graph are $C = \{\{0, 1, 2, 3, 4, 5\}, \{6, 7, 8, 9\}, \{5, 6, 10\}, \{10, 11, 12\}\}$, given in Figure 4.

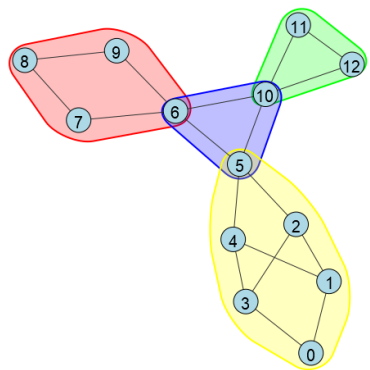


Fig. 4: Communities detected by the proposed algorithm.

We remind the following notations to describe the pseudocode of our Algorithm 1. $G = (V, E)$ is an undirected and unweighted simple graph. Since it is simple, there are no parallel edges (i. e., there can be at most one edge between any two vertices) and no loops (i. e., the vertices are not connected to themselves). n_u is the set of neighbors of vertex u, $n_u = \{v \in V | (u, v) \in E\}$. N_u is the set formed by adding u itself to n_u, $N_u = \{u\} \cup n_u$. The degree of vertex u is $deg(u) = |n_u|$. C_v represents a community which contains vertex v. S_v is a set of all communities that contains vertex v. $C = \{C_1, C_2, ...C_k\}$ is the set of detected communities.

We assume that input graph $G = (V, E)$ has n number of vertices, m number of edges ($|V| = n, |E| = m$), and after the second stage, the proposed algorithm generates k number of communities. Here, k is a small number compared with n. In the first stage of the algorithm, we have two loops. The first loop, which calculates the cosine similarities, runs in $O(m)$, and the second loop runs in $O(n)$. Sorting of the edges is performed in $O(m \, lg \, m)$. Therefore, the complexity of the first stage of the proposed algorithm is $O(n + m + m \, lg \, m) = O(m \, lg \, m)$. In the second stage, we have a loop that runs in $O(m)$. The third stage runs in $O(k^2)$. The total complexity of the proposed algorithm is $O(m \, lg \, m + k^2)$. We can accept that k is equal to $O(n)$. Finally, the complexity of the algorithm is $O(m \, lg \, m)$ for dense graphs.

Algorithm 1: detectCommunities(G).

Input : $G = (V, E)$ is an unweighted simple graph
Output : C is the list of communities

/* Stage 1: Initialization */
for *each* $(u, v) \in E$ **do**
  // calculate the similarity for each pair of adjacent vertices
  $w(u, v) = |N_u \cap N_v| / \sqrt{|N_u||N_v|}$
end
$C = \phi$
for *each* $v \in V$ **do**
  // add single vertex's community to the communities list
  $C = C \cup \{v\}$
end
sort edges $(u, v) \in E$ according to $w(u, v)$ in descending order
/* Stage 2: Generating communities */
for *each edge* $(u, v) \in E$ **do**
  // if the vertices u and v have communities consisting only of themselves
  if $C_u = \{\{u\}\}$ *and* $C_v = \{\{v\}\}$ **then**
    // add $\{u, v\}$ to the community set and remove $\{u\}$ and $\{v\}$
    $C \leftarrow C \cup \{u, v\} - \{u\} - \{v\}$
  end
  // if there is no community where u and v coexist
  else if $S_u \cap S_v = \phi$ **then**
    $CN(u, v) = \max_{C_v \in C} |C_v \cap n_u|$
    $C_v^* = arg\,max\ CN(u, v)$
    $CN(v, u) = \max_{C_u \in C} |C_u \cap n_v|$
    $C_u^* = arg\,max\ CN(v, u)$
    if $CN(u, v) > CN(v, u)$ *or* $(CN(u, v) = CN(v, u)$ *and* $deg(u) < deg(v))$ **then**
      $C \leftarrow C - C_v^*$
      $C_v^* \leftarrow C_v^* \cup \{u\}$
      $C \leftarrow C \cup C_v^*$
      if $\{u\} \in C$ **then**
        $C \leftarrow C - \{u\}$
      end
    end
    else
      $C \leftarrow C - C_u^*$
      $C_u^* \leftarrow C_u^* \cup \{v\}$
      $C \leftarrow C \cup C_u^*$
      if $\{v\} \in C$ **then**
        $C \leftarrow C - \{v\}$
      end
    end
  end
end
/* Stage 3: Merging communities */
// $C = \{C_1, C_2, ...C_k\}$
$k \leftarrow length(C)$
sort the set C in according to the numbers of elements in the communities C_i in descending order
$l \leftarrow 1$
for $i = 2$ *to* k **do**
  for $j = l$ *downto* 1 **do**
    if $|C_i \cap C_j| > |C_i|/2$ *or* $(|C_i| = 2$ *and* $|C_i \cap C_j| = 1)$ **then**
      $C \leftarrow C - C_j - C_i$
      $C_i \leftarrow C_i \cup C_j$
      $C \leftarrow C \cup C_i$
    end
  end
  renumber communities in list C with indices less than or equal to i and assign the number of the renumbered communities to l
end
return C

5. EXPERIMENTAL RESULTS

We evaluated 34 static overlapping community detection algorithms from the Community Detection library (CDlib) [6] to compare our algorithm with existing ones. Firstly for fairness of comparison, we eliminate the algorithms that require prior knowledge. Then, we run all the remaining static overlapping community detection algorithms. Secondly, we eliminate the time-consuming and unstable algorithms giving different results on each run by reviewing their results. Finally, we choose Core Expansion, Ego Networks, LPANNI, and PercoMVC algorithms which are the most successful in our small dataset. This dataset consists of three types of community structures with the following properties:

- **Clique:** every member of a community knows each other,
- **Star:** only the leader of a community knows the members, and there is no connection between the members,
- **Ring:** members of communities are connected to each other with a circular chain

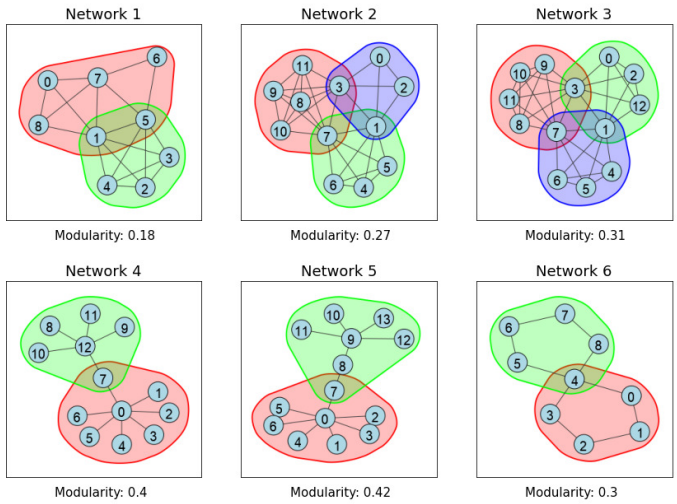


Fig. 5: Defined networks and the communities detected by proposed algorithm.

We create six networks (Figure 5) that contain clique, star, and ring communities. For example, networks 1, 2, and 3 include different sizes of cliques. Network 4 and 5 consists of different sizes of stars. Network 6 consists of equal size of rings. When we look at these networks, we can see the natural communities. For example, they are {{0, 1, 7, 8}, {1, 2, 3, 4, 5}, {5, 6, 7}} for network 1 and {{0, 1, 2, 3}, {1, 4, 5, 6, 7}, {3, 7, 8, 9, 10, 11}} for network 2. The proposed algorithm detects all natural communities for all networks except network 1 in Figure 5. The algorithm finds communities {{0, 1, 5, 6, 7, 8}, {1, 2, 3, 4, 5}} for network 1.

We run the other algorithms on these small networks. The results are given in Figure A.1. Core Expansion is the algorithm that gives the closest results to ours. However, it can not assign each vertex to a community and is not very successful in finding the overlapping vertices in the last three networks. On the other hand, PercoMVC is another successful algorithm on the first three networks. But this algorithm fails to detect star and ring-shaped structures. When we look at the results of Ego Networks, we can say that they are also true, but it produces lots of communities that can be occurred with a variant of combinations of the vertices. Because of that, it is hard to see the major communities, and it is needed to filter results. LPANNI could not detect communities in network 2 and 3. On other networks, it detects crisp communities.

We also compare the modularity scores of these algorithms in Table 1. It is known that the result between 0.3 and 0.7 for modularity indicates reasonable community structures. The proposed algorithm gets acceptable modularity scores in the five networks. Although the communities are detected as expected, we get a 0.27 modularity score on network 2. On other networks except for network 1, we get modularity scores between this interval. As a result, we could not get a higher modularity score than 0.42, although finding the communities perfectly. In addition, Ego Networks detects every possible community structure, but it gets the lowest modularity scores. Also, it is not possible to calculate modularity for every network for the Core Expansion algorithm.

	Core Expansion		Ego Networks		LPANNI		PercoMVC		Proposed	
	Q	Coverage	Q	Coverage	Q	Coverage	Q	Coverage	Q	Coverage
Network 1	0	0.84	0.03	1	0.19	0.74	0.19	0.95	0.18	1
Network 2	0.27	1	0.05	1	0	0	0.27	1	0.27	1
Network 3	0.31	1	0.06	1	0	0	0.31	1	0.31	1
Network 4	0.39	0.92	0.04	1	0.41	0.92	0	0	0.40	1
Network 5	0.41	0.92	0.05	1	0.41	0.92	0	0	0.42	1
Network 6	0.26	0.90	0.07	1	0.12	0.80	0	0	0.30	1

Tab. 1: Scores of the algorithms.

For the networks in Figure 5, we also compare the results of the adapted coverage in Table 1. Since PercoMVC does not detect a community on networks 4, 5, and 6, the adapted coverage metric gives 0 scores. Similarly, adapted coverage for the results of LPANNI algorithm on networks 2 and 3 are 0. According to these scores, all detected communities except from found by LPANNI on networks 1 and 6 are acceptable and get a higher score than 0.8. For a given vertex, the adapted coverage metric measures how social (cohesive) its neighbors are in the sense of being together in the same community. On the other hand, the proposed algorithm tries that similar members will be maximally socialized together. Therefore, our algorithm gives high scores in the adapted coverage metric.

Then, we test our algorithm on synthetic networks. We start with a small dataset to observe the results and use the LFR benchmark with the following parameters: 15 for the number of vertices, 3 for the average degree, 5 for the maximum degree, 0.1 as a mixing parameter, 3 for overlapping vertices, and 2 for the membership of overlapping vertices. We give the results in Figure B.1 and Figure C.1. Since PercoMVC does not work, we

could not give results of it. The results of the small LFR benchmark datasets indicate that the proposed algorithm has high modularity, NMI, and F-Score. Additionally, it has low runtime.

We also test our algorithm on some of the most known real datasets. Zachary's Karate Club dataset is one of the primary networks in community detection. The actual and predicted communities are given Figures 6 and 7.

Node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Predicted	1,2	1	1,3	1	2	2	2	1	1,3	1	2	1	1	1	3	3	2	1	3	1	3	1	3	3	3	3	3	1,3	1,3	3	1,3	1,3	3	1,3
Merging 1 and 2	1	1	1,2	1	1	1	1	1	1,2	1	1	1	1	1	2	2	1	1	2	1	2	1	2	2	2	2	2	1,2	1,2	2	1,2	1,2	2	1,2
Real	1	1	1	1	1	1	1	1	2	2	1	1	1	1	2	2	1	1	2	1	2	1	2	2	2	2	2	2	2	2	2	2	2	2

Fig. 6: Communities of Karate Club with the label of nodes.

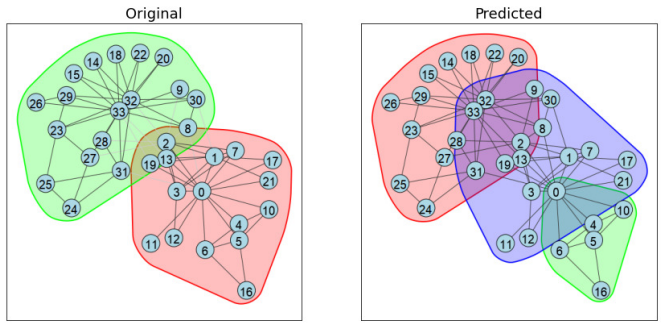


Fig. 7: Communities of Karate Club.

We found three communities and got NMI 0.29, F-Score 0.52, and modularity 0.25 (Table 2). When we examine the results in Figure 7, we detect two major communities; blue and red ones. The green one in predicted communities is a hidden community. Vertices 4, 5, 6, 10, and 16 are connected to the red community with vertex 0. Except for vertex 0, they do not have any connection with the rest of the network. If we ignore this hidden structure and accept this as a part of the red community like in Figure 7, we correctly group all vertices out of vertex 9. According to our algorithm, vertex 9 belongs to the red community in the real dataset. Our algorithm identifies 2, 8, 27, 28, 30, 31, and 33 as overlapping vertices.

	Real			Proposed			
	k	Q	Coverage	k	Q	NMI	F-Sc
Karate Club	2	0.37	0.87	3	0.25	0.29	0.52
Les Miserables	7	0.56	0.76	7	0.39	0.46	0.45
Politic Books	3	0.41	0.84	4	0.43	0.35	0.22
Dolphins	2	0.37	0.96	5	0.34	0.30	0.34

Tab. 2: Results of real datasets.

Similarly, the results of our algorithm on Dolphins, Les Miserables, and Politic Books

are given in Table 2 and Figure 8. It is seen that predicted communities are very close to real ones, and they are more detailed than real ones.

Core Expansion

	k	Q	NMI	F-Score	Coverage
karate	2	0	0.36	0.67	0.54
lesmis	13	0	0.40	0.67	0.89
polbooks	16	0	0.13	0.08	0.76
dolphins	12	0	0.13	0.15	0.65

LPANNI

	k	Q	NMI	F-Score	Coverage
karate	3	0.40	0.63	0.60	0.82
lesmis	10	0.52	0.57	0.61	0.87
polbooks	6	0.50	0.34	0.14	0.91
dolphins	6	0.51	0.33	0.26	0.81

Ego Network

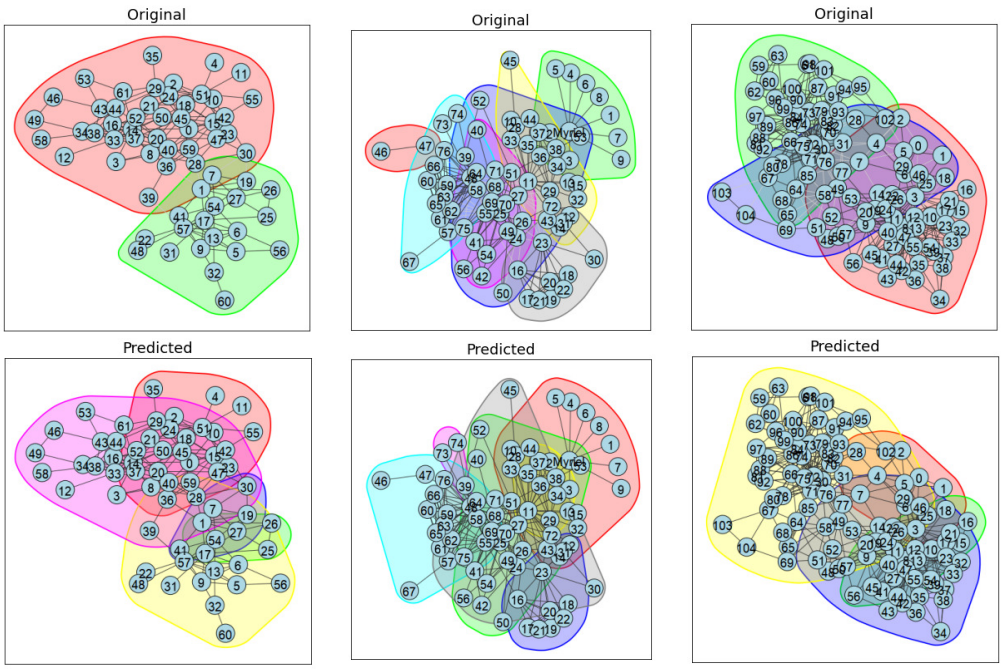
	k	Q	NMI	F-Score	Coverage
karate	34	0.04	0.14	0.05	1
lesmis	63	0.07	0.25	0.16	1
polbooks	105	0.04	0.10	0.02	1
dolphins	62	0.05	0.10	0.02	1

PercoMVC

	k	Q	NMI	F-Score	Coverage
karate	3	0	0.38	0.69	0.87
lesmis	4	0	0.40	0.32	0.93
polbooks	6	0	0.29	0.34	0.84
dolphins	4	0	0.18	0.27	0.57

Tab. 3: Results of algorithms on real datasets.

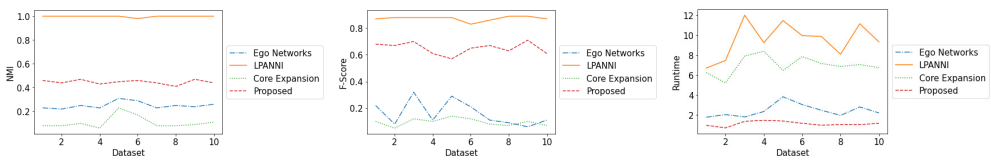
We also compare the modularity, NMI, F-Score, and coverage values with the other algorithms for real datasets in Table 3. The problem with the Core Expansion and PercoMVC is that they can not group every vertex, and Ego Networks finds too many communities. Although LPANNI is successful on Karate Club and Les Miserables datasets, it gets lower NMI and F-Scores on Politic Books and Dolphins datasets. On the other hand, the proposed algorithm is better for these datasets.



(a) Dolphins. (b) Lesmis. (c) Polbooks.

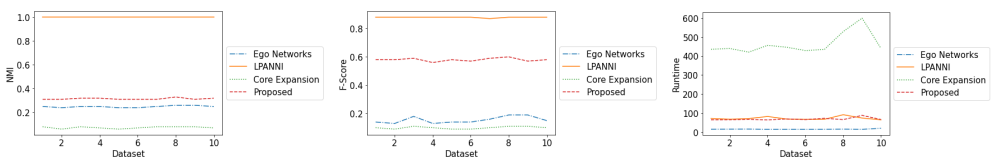
Fig. 8: Results of the proposed algorithm on real datasets.

Finally, we use the LFR benchmark to produce larger networks. We use the following parameters: 1000 for the number of vertices, 30 for the average degree, 100 for the maximum degree, 0.1 as the mixing parameter, 100 for the number of overlapping vertices, and 2 for the membership of overlapping vertices. We generate 10 networks. NMI and F-Score results of each algorithm with runtime results are given in Figure 9. Since PercoMVC algorithm uses Clique Percolation Method, it is not possible to get a solution for every network produced by the LFR benchmark. For this reason, we could not give the results of PercoMVC. LPANNI has the highest NMI and F-Score but also has the longest runtime. The proposed algorithm gives the best results according to the runtime. On the other hand, when we increase the number of vertices and the overlapping vertices ten times in the benchmark, Ego Networks gives the longest runtime. The proposed algorithm is still better than the two algorithms (Figure 10).



(a) NMI (b) F-Score (c) Runtime (sec)

Fig. 9: Results of algorithms on LFR benchmark with 1000 vertices.



(a) NMI (b) F-Score (c) Runtime (sec)

Fig. 10: Results of algorithms on LFR benchmark with 10000 vertices.

6. CONCLUSIONS

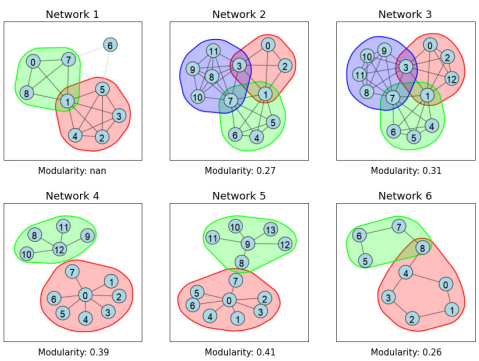
In this study, we propose an overlapping community detection algorithm that can detect the communities in star, clique, and ring structures. We test our algorithm in both real and synthetic datasets. We use modularity, NMI, F-score, and an overlapping version of coverage as evaluation criteria. Our algorithm successfully finds the hidden structures and overlapping vertices without a need for prior knowledge. It can also work on different networks, even unweighted and undirected.

The experiments confirm no algorithm gives the best results for all dataset. Because of that, every algorithm in this area is vulnerable. For example, although LPANNI has the highest scores in LFR benchmarks, it is not successful in detecting the communities that we define. On the other hand, although PercoMVC perfectly defines communities for the first three networks that we defined, it could not work on LFR benchmarks.

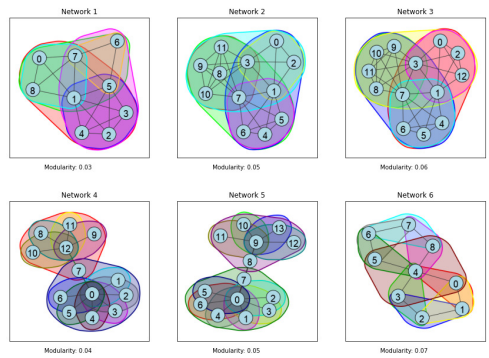
However, it is better than Core Expansion on real datasets. Despite Core Expansion being the worst algorithm on the LFR benchmark, it is successful on our small datasets. Although Ego Networks finds every possible community, it has a long runtime, and it is hard to see major communities. The proposed algorithm works well in all these datasets and among all 34 overlapping community detection algorithms. However, there is still a need for improvement of the algorithms.

In the future, we plan to work to improve the algorithm to detect overlapping and disjoint communities at the same time and parallelize the algorithm to improve the running time of our algorithm. We also plan to show different levels of communities as in the hierarchical methods.

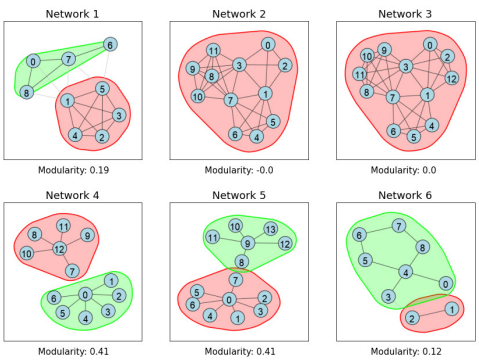
A. APPENDIX: RESULTS OF THE ALGORITHMS ON OUR SMALL DATASET



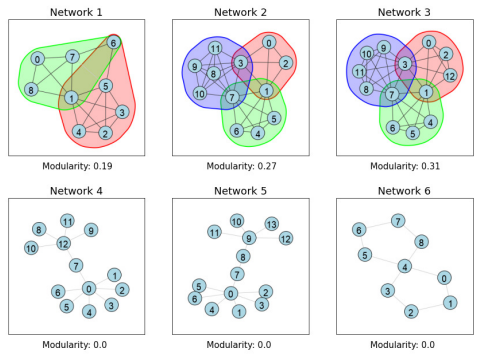
(a) Core Expansion



(b) Ego Networks



(c) LPANNI



(d) PercoMVC

Fig. A.1: Results of the algorithms on our small dataset.

B. APPENDIX: RESULTS OF LFM BENCHMARK

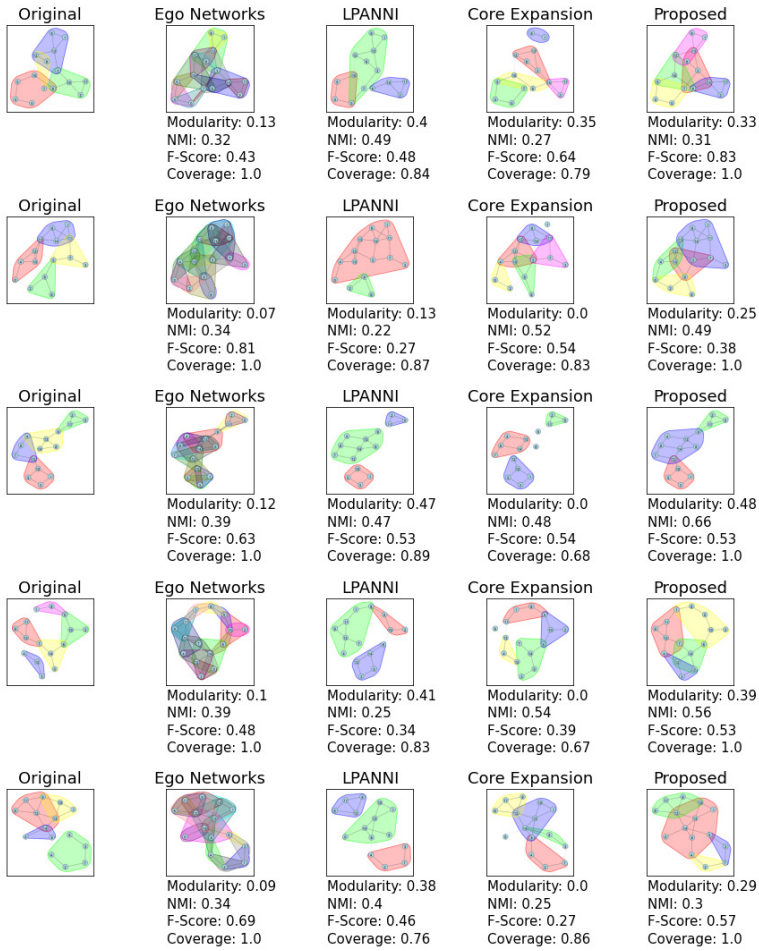
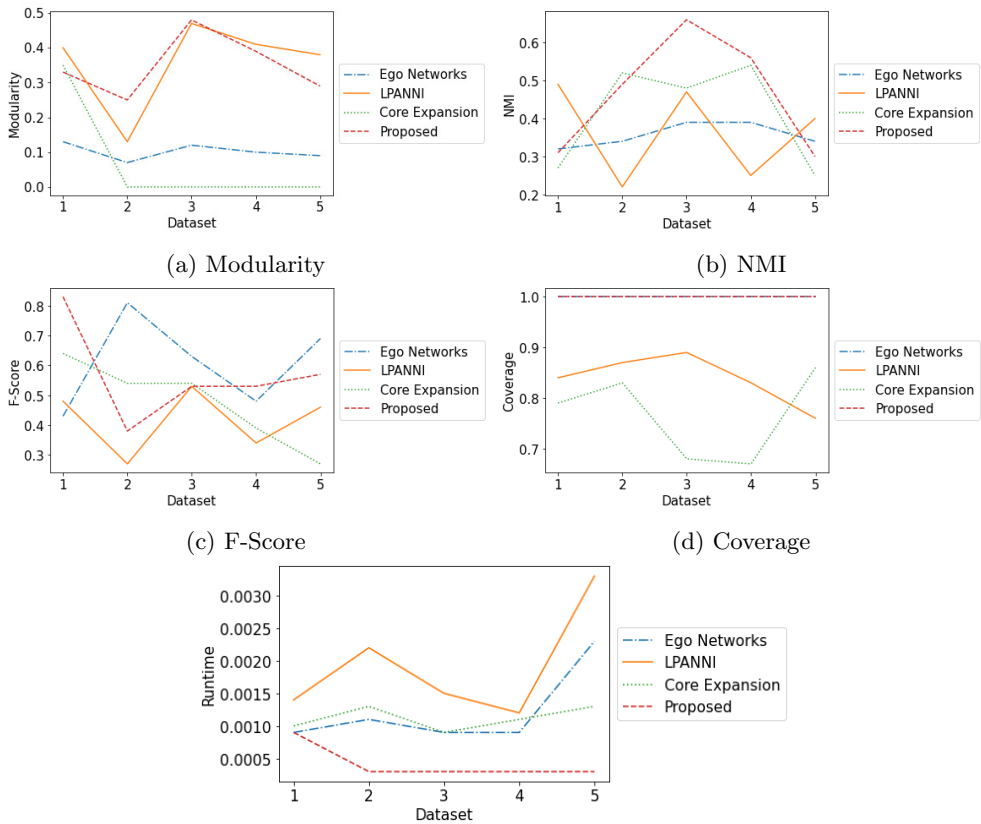


Fig. B.1: Results of LFM benchmark

C. APPENDIX: RESULTS OF ALGORITHM ON SMALL LFR BENCHMARK



(a) Modularity

(b) NMI

(c) F-Score

(d) Coverage

(e) Runtime

Fig. C.1: Results of the algorithms on small LFR benchmark.

ACKNOWLEDGEMENT

We would like to thank Editor in Chief Prof. Sergej Čelikovský and the anonymous reviewers for taking the time and effort necessary to review the study. We sincerely appreciate all valuable comments and suggestions, which improved the quality of our paper.

(Received February 27, 2022)

REFERENCES

[1] S. Aghaalizadeh, S. T. Afshord, A. Bouyer, and B. Anari: A three-stage algorithm for local community detection based on the high node importance ranking in social networks. Phys. A *563* (2021), 125420. DOI:10.1016/j.physa.2020.125420

[2] S. Agrawal and A. Patel: SAG cluster: An unsupervised graph clustering based on collaborative similarity for community detection in complex networks. Phys. A *563* (2021), 125459. DOI:10.1016/j.physa.2020.125459

[3] Y. Y. Ahn, J. P. Bagrow, and S. Lehmann: Link communities reveal multiscale complexity in networks. Nature *466* (2010), 761–764. DOI:10.1038/nature09182

[4] M. Arab and M. Hasheminezhad: Limitations of quality metrics for community detection and evaluation. In: 3th International Conference on Web Research (ICWR), Tehran 2017.

[5] J. P. Attal, M. Malek, and M. Zolghadri: Overlapping community detection using core label propagation algorithm and belonging functions. Appl. Intell. *51* (2021), 8067–8087. DOI:10.1007/s10489-021-02250-4

[6] CDlib: Community Discovery Library. `https://cdlib.readthedocs.io/en/latest/`, 2022.

[7] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly: Metrics for community analysis: A survey. ACM Computing Surveys *50* (2018), 1–37. DOI:10.1145/3139222

[8] D. Chen, M. Shang, Z. Lv, and Y. Fu: Detecting overlapping communities of weighted networks via a local algorithm. Phys. A *389* (2010), 4177–4187. DOI:10.1016/j.physa.2010.05.046

[9] A. Choumane, A. Awada, and A. Harkous: Core expansion: a new community detection algorithm based on neighborhood overlap. Soc. Netw. Anal. Min. *10* (2020), 30. DOI:10.1007/s13278-020-00647-6

[10] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi: DEMON: A local-first discovery method for overlapping communities. In: Proc. 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing 2012.

[11] Ş. Emrah Amrahov, and B. Tugrul: A community detection algorithm on graph data. In: 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya 2018.

[12] I. Farkas, D. Ábel, G. Palla, and T. Vicsek: Weighted network modules. New J. Phys. *9* (2007), 180. DOI:10.1177/1461444807072228

[13] L. Feng, Q. Zhao, and C. Zhou: Incorporating affiliation preference into overlapping community detection. Phys. A *563* (2021), 125429. DOI:10.1016/j.physa.2020.125429

[14] S. Fortunato: Community detection in graphs. Phys. Rep. *486* (2009), 75–174. DOI:10.1016/j.physrep.2009.11.002

[15] Y. Gao, X. Yu, and H. Zhang: Overlapping community detection by constrained personalized pagerank. Expert Syst. Appl. *173* (2021), 114682. DOI:10.1016/j.eswa.2021.114682

[16] J. Ge, H. Sun, C. Xue, L. He, X. Jia, H. He, and J. Chen: LPX: Overlapping community detection based on X-means and label propagation algorithm in attributed networks. Comput. Intell. *37* (2021), 484–510. DOI:10.1111/coin.12420

[17] P. K. Gopalan and D. M. Blei: Efficient discovery of overlapping communities in massive networks. Proc. Natl. Acad. Sci. USA *110* (2013), 14534–14539. DOI:10.1073/pnas.1221839110

[18] S. Goswami and A. K. Das: Determining maximum cliques for community detection in weighted sparse networks. Knowl. Inf. Syst. *64* (2022), 289–324. DOI:10.1007/s10115-021-01631-y

[19] S. Gregory: An algorithm to find overlapping community structure in networks. In: Proc. 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw 2007.

[20] S. Gregory: A fast algorithm to find overlapping communities in networks. In: Machine Learning and Knowledge Discovery in Databases, Berlin 2008.

[21] S. Gregory: Finding overlapping communities in networks by label propagation. New J. Phys. *12* (2010), 103018. DOI:10.1088/1367-2630/12/10/103018

[22] K. Guo, Q. Wang, J. Lin, L. Wu, W. Guo, and K.-M. Chao: Network representation learning based on community-aware and adaptive random walk for overlapping community detection. Appl. Intell. *52* (2022), 9919–9937. DOI:10.1007/s10489-021-02999-8

[23] S. Gupta and P. Kumar: An overlapping community detection algorithm based on rough clustering of links. Data Knowl. Engrg. *125* (2020), 101777. DOI:10.1016/j.datak.2019.101777

[24] C. He, H. Liu, Y. Tang, S. Liu, X. Fei, Q. Cheng, and H. Li: Similarity preserving overlapping community detection in signed networks. Future Gener. Comput. Syst. *116* (2021), 275–290. DOI:10.1016/j.future.2020.10.034

[25] D. Jin, B. Gabrys, and J. Dang: Combined node and link partitions method for finding overlapping communities in complex networks. Sci. Rep. *5* (2015), 8600. DOI:10.1038/srep08600

[26] H. S. Joghan, A. Bagheri, and M. Azad: Weighted label propagation based on local edge betweenness. J. Supercomput. *75* (2019), 8094–8114. DOI:10.1007/s11227-019-02978-4

[27] J. Kim, S. Lim, J. G. Lee, and B. S. Lee: LinkBlackHole*: Robust overlapping community detection using link embedding. IEEE Trans. Knowl. Data Engrg. *31* (2019), 2138–2150. DOI:10.1109/TKDE.2018.2873750

[28] I. B. E. Kouni, W. Karoui, and L. B. Romdhane: Node importance based label propagation algorithm for overlapping community detection in networks. Expert Syst. Appl. *162* (2020), 113020. DOI:10.1016/j.eswa.2019.113020

[29] A. Lancichinetti, S. Fortunato, and J. Kertesz: Detecting the overlapping and hierarchical community structure in complex networks. New J. Phys. *11* (2009), 033015. DOI:10.1088/1367-2630/11/3/033015

[30] C. Lee, F. Reid, A. McDaid, and N. Hurley: Detecting highly overlapping community structure by greedy clique expansion. `https://arxiv.org/abs/1002.1827`, 2010.

[31] C. Li, H. Chen, T. Li, and X. Yang: A stable community detection approach for complex network based on density peak clustering and label propagation. Appl. Intell. *52* (2021), 1188–1208. DOI:10.1007/s10489-021-02287-5

[32] H. V. Lierde, G.S. Member, and T. W. S. Chow: Scalable spectral clustering for overlapping community detection in large-scale networks. IEEE Trans. Knowl. Data. Engrg. *32* (2020), 754–767. DOI:10.1109/TKDE.2019.2892096

[33] S. Lim, S. Ryu, S. Kwon, K. Jung, and J. G. Lee: LinkSCAN: Overlapping community detection using the link-space transformation. In: IEEE 30th International Conference on Data Engineering, Chicago 2014.

[34] H. Long: Overlapping community detection with least replicas in complex networks. Inform. Sci. *453* (2018), 216–226. DOI:10.1016/j.ins.2018.03.063

[35] H. Lu, Z. Zhang, Z. Qu, and Y. Kang: LPANNI: Overlapping community detection using label propagation in large-scale complex networks. IEEE Trans. Knowl. Data. Engrg. *31* (2019), 1736–1749. DOI:10.1109/TKDE.2018.2866424

[36] T. Ma, Q. Liu, J. Cao, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan: LGIEM: Global and local node influence based community detection. Future Gener. Comput. Syst. *105* (2020), 533–546. DOI:10.1016/j.future.2019.12.022

[37] T. Ma, Y. Wang, M. Tang, J. Cao, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan: LED: A fast overlapping communities detection algorithm based on structural clustering. Neurocomputing *207* (2016), 488–500. DOI:10.1016/j.neucom.2016.05.020

[38] A. Mahabadi, and M. Hosseini: SLPA-based parallel overlapping community detection approach in large complex social networks. Multimed. Tools. Appl. *80* (2021), 6567–6598. DOI:10.1007/s11042-020-09993-1

[39] A. D. McCarthy, T. Chen, R. Rudinger, and D. W. Matula: Metrics matter in community detection. In: Complex Networks and Their Applications VIII (H. Cherifi, S. Gaito, J. Mendes, E. Moro, L. Rocha, eds.), Stud. Comput. Intell., *881*, Springer 2019, pp. 164–175. DOI:10.1007/978-3-030-36687-2_14

[40] A. McDaid, D. Greene, and N. Hurley: Normalized mutual information to evaluate overlapping community finding algorithms. `https://arxiv.org/abs/1110.2515`, ArXiv 2011.

[41] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó: Fuzzy communities and the concept of bridgeness in complex networks. Phys. Rev. E *77* (2008), 1–12. DOI:10.1103/physreve.77.016107

[42] M. E. J. Newman, and M. Girvan: Finding and evaluating community structure in networks. Phys. Rev. E *69* (2004), 1–15. DOI:10.1103/physreve.69.026113

[43] G. Palla, I. Derényi, I. Farkas, and T. Vicsek: Uncovering the overlapping community structure of complex networks in nature and society. Nature *435* (2005), 814–818. DOI:10.1038/nature03607

[44] B. Pattabiraman, M. A. Patwary, A. H. Gebremedhin, W. K. Liao, and A. Choudhary: Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. Internet Math. *11* (2015), 421–448. DOI:10.1080/15427951.2014.986778

[45] H. S. Pattanayak, H. K. Verma, and A. L. Sangal: Community detection metrics and algorithms in social networks. In: First International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar 2018, pp. 483–489. DOI:10.1109/icsccc.2018.8703215

[46] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon: Overlapping community detection using Bayesian non-negative matrix factorization. Phys. Rev. E *83* (2011), 066114. DOI:10.1103/physreve.83.066114

[47] M. Qin, and K. Lei: Dual-channel hybrid community detection in attributed networks. Inform. Sci. *551* (2021), 146–167. DOI:10.1016/j.ins.2020.11.010

[48] H. Shen, X. Cheng, K. Cai, and M-B. Hu: Detect overlapping and hierarchical community structure in networks. Phys. A *388* (2009), 1706–1712. DOI:10.1016/j.physa.2008.12.021

[49] Z. Sun, B. Wang, J. Sheng, Z. Yu, and J. Shao: Overlapping community detection based on information dynamics. IEEE Access *6* (2018), 70919–70934. DOI:10.1109/ACCESS.2018.2879648

[50] X. Wang, G. Liu, and J. Li: Overlapping community detection based on structural centrality in complex networks. IEEE Access *5* (2017), 25258–25269. DOI:10.1109/ACCESS.2017.2769484

[51] Z. X. Wang, Z. C. Li, X. F. Ding, and J. H. Tang: Overlapping community detection based on node location analysis. Knowl. Based Syst. *105* (2016), 225–235. DOI:10.1016/j.knosys.2016.05.024

[52] X. Wen, W. N. Chen, Y. Lin, T. Gu, H. Zhang, Y. Li, Y. Yin, and J. Zhang: A maximal clique based multiobjective evolutionary algorithm for overlapping community detection. IEEE Trans. Evol. Comput. *21* (2017), 363–377. DOI:10.1109/TEVC.2016.2605501

[53] J. J. Whang, F. David, and I. S. Dhillon: Overlapping community detection using neighborhood-inflated seed expansion. IEEE Trans. Knowl. Data. Engrg. *28* (2016), 1272–1284. DOI:10.1109/TKDE.2016.2518687

[54] Q. Wu, R. Chen, L. Wang, and K. Guo: A label propagation algorithm for community detection on high-mixed networks. Concurr. Comput. Pract. Exp. *33* (2021). DOI:10.1002/cpe.6141

[55] J. Xie, B. K. Szymanski, and X. Liu: SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: IEEE 11th International Conference on Data Mining Workshops, Vancouver 2011, pp. 344–349. DOI:10.1109/icdmw.2011.154

[56] H. Yu, R. Ma, J. Chao, and F. Zhang: An overlapping community detection approach based on DeepWalk and improved label propagation. IEEE Trans. Comput. Soc. Syst. (2022), 1–11. DOI:10.1109/tcss.2022.3152579

[57] S. Zhang, R. S. Wang, and X. S. Zhang: Uncovering fuzzy community structure in complex networks. Phys. Rev. E *76* (2007), 046103. DOI:10.1103/physreve.76.046103

[58] Y. Zhang and D. Y. Yeung: Overlapping community detection via bounded nonnegative matrix tri-factorization. In: Proc. 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing 2012, pp. 606–614. DOI:10.1145/2339530.2339629

[59] Q. Zhou, S. Cai, and Y. Zhang: Detecting overlapping community structure with node influence. IEEE Access *7* (2019), 171223–171234. DOI:10.1109/ACCESS.2019.2955161

Pelin Çetin, Ankara University, Department of Computer Engineering, Ankara, 06830. Turkey.

e-mail: pelincetin.cs@gmail.com

Sahin Emrah Amrahov, Ankara University, Department of Computer Engineering, Ankara, 06830. Turkey.

e-mail: emrah@eng.ankara.edu.tr