Overlapping community detection algorithms using Modularity and the cosine

Duy Hieu DO and Thi Ha Duong PHAN
Institute of Mathematics
Vietnam Academy of Science and Technology

Email: ddhieu@math.ac.vn (Do Duy Hieu), phanhaduong@math.ac.vn (Phan Thi Ha Duong).

Abstract

The issue of network community detection has been extensively studied across many fields. Most community detection methods assume that nodes belong to only one community. However, in many cases, nodes can belong to multiple communities simultaneously. To address this, several research studies on overlapping community detection often involve a two-step approach. The first step uses existing algorithms to identify known communities, and the second involves identifying overlaps. This paper presents two overlapping network community detection algorithms that build on this two-step approach, using the extended modularity and cosine function. To demonstrate the feasibility and effectiveness of these algorithms, we conducted experiments using real data.

1 Introduction

In recent years, many studies have focused on network systems such as social networks, biological networks, and technological networks [28, 4]. Researchers have been interested in studying local properties of networks, including clustering [42], degree distribution [3, 1], and correlations [31, 29]. They have also investigated large-scale properties such as path length [42], coverage [8, 6], and hierarchy [37, 7]. Among these properties, structured communities have received the most attention, such as in [14, 15, 21, 38].

The network community problem has been studied extensively in many fields, and most community detection methods assume that nodes belong to only one community. However, in many cases, nodes can participate in multiple communities simultaneously, making the problem more challenging. For example, in a network of scientists where nodes represent individuals and edges represent collaborations, a scientist may belong to multiple communities if they collaborate with researchers from different fields. This situation has motivated researchers to develop methods to identify overlapping communities automatically. Some authors have made significant efforts to characterize communities with overlapping nodes, as evidenced by recent papers such as [33, 34, 22, 20]. Although studying separate community structures is widely studied, developing methods to handle overlapping communities is necessary for a complete understanding of network structures.

1.1 Overlapping community detection algorithms

We are interested in community detection involving a two-step approach. The first step uses existing algorithms to identify known communities, and the second consists in identifying overlaps, such as two of the following algorithms.

The first Algorithm we were interested in is the one in [35]. The authors propose a simple approach to identify overlapping communities in a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ represents n nodes and $E \subseteq V \times V$ represents m edges. A cover is defined as a family of subsets of nodes:

$$C = \{C_1; C_2; ...; C_k\}.$$

Each subset C_u is called a cluster or community, and the objective of community detection is to find a cover C_j that best describes the community structure of the graph, such that nodes within clusters are more densely connected than to nodes in other clusters.

The authors associate an affiliation matrix $F_{C_j} \in \mathbb{R}^{|V|\times|C|}$ with the cover C_j, where F_{vc} represents the degree of affiliation of vertex v with community $C_j \in C$. The following calculation obtains the matrix of belonging coefficients for the final cover.

$$F_{uC_j} = \begin{cases} 1 & if \ \frac{\sum_{(u,v)\in E} \chi_{vC_j}}{d_u} \geq \theta, \\ 0 & otherwise. \end{cases} \tag{1.1}$$

where χ_{vC_j} is an indicator variable that takes the value 1 when the edge v of the original graph G belongs to cluster C_j and 0 otherwise, and θ is a threshold parameter, and d_u is the degree of vertex u in the graph G. Thus, the belonging coefficient F_{uC_j} of node u to cluster C_j is proportional to the number of adjacent edges belonging to the cluster C_j. As the value of θ increases, the degree of overlapping between the communities also increases. For the convenience of presentation, we temporarily call this **Parameterized Overlap Algorithm** (or **Paramet. Overlap** for short).

The second Algorithm we are interested in is the one in [10]. First of all, the authors defined Modularity Q_0 for the overlapping community as follows:

$$Q_0 = \frac{1}{2m} \sum_{C_j \in C} \sum_{u,v \in C_j} \alpha_{uC_j} \alpha_{vC_j} \left(A_{uv} - \frac{d_u d_v}{2m} \right),$$

where m is the number of edges, d_u is the degree of u, and

$$\alpha_{uC_j} = \frac{d_{uC_j}}{\sum_{C_j \in C} d_{uC_j}}, \tag{1.2}$$

with $d_{uC_j} = \Sigma_{v \in C_j} A_{uv}$. Obviously, the definition in Eq. 1.2 satisfies the following conditions [30]:

$$0 \leq \alpha_{uC_j} \leq 1, \quad \forall C_j \in C, u \in V$$

and

$$\sum_{C_j \in C} \alpha_{uC_j} = 1$$

Furthermore, if node u belongs to only one community C_j, α_{uC_j} equals 1; if node u does not belong to the community C_j, $\alpha_{uC_j} = 0$. At is to say, Eq. ef pt6 is consistent with the definition of Modularity for a non-overlapping community in [27].

After that, the authors proposed the following overlapping community detection algorithm. For a community C_j and a node u, the belonging degree $B(u, C_j)$ between u and C_j is defined as

$$B(u, C_j) = \frac{\sum_{v \in C_j} A_{uv}}{d_u}.$$

Find all neighbors N_{C_j} of the initial community C_j. If $\exists u \in N_{C_j}, B(u, C_j) > B^U$ (in the paper [10], the authors took $B^U = 0.5$, which means a node u is tight enough with community C_j if the belonging degree is larger than 0.5), add all these type nodes into community C_j directly. If $\forall u \in N_{C_j}, B(u, C_j) < B^L$ (in the paper [10], the authors took $B^L = 0.4$ that means a node u is loose enough with community C_j if the belonging degree is less than 0.4, and the node needs checking further using the Modularity Q_0 for a belonging degree between 0.4 and 0.5), stop expanding the community; otherwise, add the nodes with $B^L \leq B(u, C_j) \leq B^U$ into the community if the Modularity Q_0 becomes larger after adding. The detailed processes are listed as follows (for the convenience of presentation, we temporarily call this **Module Overlap Algorithm**).

- Find all neighbors N_{C_j} of community C_j, and calculate the belonging degree $B(u, C_j)$ for each neighbor u.
- Find all nodes with $B(u, C_j) > B^U$ and $B^L \leq B(u, C_j) \leq B^U$, denoted by $N_u = \{u \mid B(u, C_j) > B^U\}$ and $N_{lu} = \{u \mid B^L \leq B(u, C_j) \leq B^U\}$, respectively.
- If $|N_u| > 0$, add all nodes of N_u into the community and obtain a larger partial community, also denoted by C_j, and return to step(i).
- If $|N_{lu}| > 0$, for each node u in N_{lu}, add u into the community if the Modularity Q_0 becomes larger after adding it, obtain a larger partial community, also denoted by C_j.
- If $|N_u| = 0$ and $|N_{lu}| = 0$, stop expanding and mine a community ultimately.

1.2 Random walk on graphs

Let $G = (V, E)$ be a directed graph with n vertices and m edges, define A the adjacency matrix of G. For $i = 1, 2, ..., n$, we define d_i^{out} the out-degree of vertex i, and d_i^{in} the in-degree of vertex i.

It is worth noting that undirected graphs can be seen as a special case of directed graphs, where the adjacency matrix is symmetric and the out-degree and in-degree of each vertex are equal.

A random walk on a graph is a process that starts at a given vertex and moves to another vertex at each time step, the next vertex in the walk is chosen uniformly at random from among the neighbors of the current vertex.

For conciseness, in the following, we will use the out-degree of a vertex as its degree, denoted by d_i unless stated otherwise. We define a diagonal matrix D using the vertex out-degrees and let $P = D^{-1}A$. The matrix $P = [p_{ij}]_{i,j=\overline{1,n}}$ represents the transition probabilities of a Markov chain associated with a random walk on graph G. At each vertex i, the random walk can move to vertex j with probability $p_{ij} = a_{ij}/d_i$ if $(i, j) \in E$. Then P^t represents the transition probabilities of this random walk after t steps.

We assume that G is strongly connected, meaning there is a directed path from any vertex i to every other vertex j. We consider an aperiodic random walk X_k finite graph with n vertices. According to the convergence theorem for finite Markov chains [**?**], the associated transition matrix P satisfies $\lim_{k\to\infty} P = P_\infty$, where $(P_\infty)_{ij} = \phi_j$, the j th component of the unique stationary distribution $\phi = (\phi_1, \phi_2, ..., \phi_n)$.

For an undirected graph G where $a_{ij} = a_{ji}$, we have $d_i = d_i^{out} = d_i^{in}$, and $\phi_i = d_i/2m$ for all vertex i.

1.3 Our contribution

In this paper, each algorithm consists of two steps. In the first step, we use existing community detection algorithms, such as the Hitting times Walktrap algorithm [11], Walktrap algorithm [34], or the Louvain algorithm [37], to find disjointed communities for the network. In the second step, we determine whether a vertex belongs to a community or not by using modularity or cosine functions.

- In Section 2, we introduce a concept called Theta-Modularity, an extension of regular Modularity. To do this, we look at the number of edges between the vertex and that community. Specifically, the criteria for a vertex to belong to a community is that the number of edges between them must be sufficiently large and dependent on the total degree of the vertices in that community.
- In Section 3, we use the method to coordinate the graph's vertices. Then we can identify each vertex as a vector and find the cluster centers that are the average coordinates of the vertices in that cluster. Finally, we will propose an overlapping algorithm based on the idea that vertices of the same cluster will create a small angle with the cluster center; in other words, the cosine of that vertex and the cluster center are more significant than a constant θ.

Finally, to assess the effectiveness and rationality of our algorithms, we will conduct experiments to compare and evaluate the clustering results of our algorithms with other algorithms and the clustering generated by the graph generation model. Specifically, in Subsection 5.2.2, we will compare two of our algorithms for undirected graphs, namely the Parameterized Modularity Overlap Algorithm and the Cosine Overlap Algorithm, with two other algorithms, namely the Parameterized Overlap Algorithm [35] and the Module Overlap Algorithm [10]. We will compare these algorithms based on Modularity and the clustering generated by the graph generation model. Additionally, we will perform experiments on real datasets and compare the algorithms based on Modularity. In Subsection 5.2.3, we will apply our directed graph algorithms to randomly generated graphs and compare the results with the clustering generated by the graph generation model.

2 Overlapping community detection using modularity

In this section, we will use the modularity function to determine whether a vertex belongs to a community or not. The classic modularity function evaluates the difference between the real number of edges between two vertices and the expected number of edges between them. Our new approach is to introduce a parameter to make the evaluation more flexible. Specifically,

we will use a threshold for the expected number of edges. This new parameterized modularity function will be applied to undirected graphs in section 2.1, where the expected number is calculated based on the degree of the vertices. It will also be used in section 2.2 for directed graphs, where the expected number is calculated based on the in-degree and out-degree of the vertices. A breakthrough in section 2.3 is that the parameterized modularity function for directed graphs will be defined not based on the vertex degrees but on the stationary distribution of a random walk on the graph. This provides a more advanced approach to determining community membership.

2.1 Overlapping community detection for undirected graphs

The expected number of edges falling between two vertices u and v in the configuration model is equal to $d_u d_v/2m$, where d_u is the degree of vertex u and m is the total number of edges in the observed network. The actual number of edges observed to fall between the same two vertices is equal to the element A_{uv} of the adjacency matrix A, so that The actual-minus-expected edge count for the vertex pair is $A_{uv} - d_u d_v/2m$. Suppose $C = \{C_1, C_2, ..., C_k\}$ is a cover of the vertices of the undirected graph G, the modularity Q (as defined in [27]) is then equal to

$$Q = \frac{1}{2m} \sum_{u,v \in V} \left(A_{uv} - \frac{d_u d_v}{2m} \right) \delta\left(C_u, C_v\right) = \frac{1}{2m} \sum_{C_j \in C} \sum_{u,v \in C_j} \left(A_{uv} - \frac{d_u d_v}{2m} \right). \tag{2.1}$$

where $\delta\left(C_u, C_v\right)$ is 1 if u and v are in the same community, and 0 otherwise.

This modularity illustrates the criteria that u and v belong to the same cluster if the real number of edges between them is greater than the expected number of edges between them. However, we find many practical problems when dividing clusters by data; depending on the goal, the required criteria is more flexible. Therefore, we propose a new modularity with theta coefficient as follows.

$$Q(\theta) = \frac{1}{2m} \sum_{C_j \in C} \sum_{u,v \in C_j} \left(A_{uv} - \theta \frac{d_u d_v}{2m} \right). \tag{2.2}$$

Our modularity means that two vertices u and v belong to the same cluster if the number of edges between u and v is more significant than θ-times the expected number of edges between them.

We observe that the modularity of clustering of graph G can be expressed as the sum of the modularity of each cluster, as shown in formula 2.2. This means that if we add a vertex to a cluster, only the modularity value of that specific cluster will be affected. As a result, for every community, C_j and vertex $u \in V(G) \setminus C_j$, the modularity of cluster C_j will change by an amount when we add vertex u to it, and the following formula can calculate it.

$$\Delta Q(\theta)_{u,C_j} = \frac{1}{2m} \sum_{w \in C_j} \left(A_{uw} - \theta \frac{d_u d_w}{2m} \right). \tag{2.3}$$

From the above comment, we propose that vertex u will be added to the community C_j if then $\Delta Q(\theta)_{u,C_j}$ is positive:

$$\frac{1}{2m} \sum_{w \in C_j} \left(A_{uw} - \theta \frac{d_u d_w}{2m} \right) > 0. \tag{2.4}$$

It implies that

$$\sum_{w \in C_j} \frac{A_{uw}}{d_u} > \theta \sum_{w \in C_j} \frac{d_w}{d}. \tag{2.5}$$

Remark 2.1 *From (1.1) and (2.5), we have remarked that vertex u belongs to the community C_j if the number of edges between u and C_j is large enough. However, in the equation (1.1), the number of edges between u and C_j must always be greater than a fixed θ constant. It is independent of the properties of the community C_j. Our method is more reasonable because it depends not only on the θ coefficient but also on the characteristics of the community C_j. Specifically, it depends on the sum of the degrees of the vertices in the community C_j.*

From there, we propose the overlap detection algorithm described in **Algorithm** 1. We will call this the **Parameterized Modularity Overlap Algorithm** for undirected graphs (or **Paramet. Modul.** for short).

Algorithm 1: Parameterized Modularity Overlap Algorithm

Input: An undirected graph G and a threshold value θ
Output: Clusters of vertices with overlapping communities
Apply the Louvain algorithm [40] to obtain the initial clustering of G into communities $C_1, C_2, ..., C_k$;
repeat
  foreach *vertex* u **do**
    foreach *community* C_j *adjacent to* u *and not containing* u **do**
      if $\sum_{w \in C_j} \frac{A_{uw}}{d_u} > \theta \sum_{w \in C_j} \frac{d_w}{d}$ **then**
        Add u to C_j;
      end
    end
  end
until *No communities* C_j *meet the condition*;

In the Algorithm 1, we have to traverse all the vertices, and for each vertex, we consider all the adjacent communities with it. So the computational complexity of this Algorithm will be $O(kn)$, where n is the number of vertices of the graph, and k is the number of communities.

2.2 Overlapping community detection for directed graphs using modularity

In [2, 26], the authors had given the in/out-degree sequence of directed graph, in which the probability to have an edge from vertex v to vertex u is determined by $d_u^{in} d_v^{out}$, where d_u^{in} and d_v^{out} are the in- and out-degrees of the vertices. Suppose $C = \{C_1, C_2, ..., C_k\}$ is a cover of G, the modularity Q is defined as.

$$Q^d = \frac{1}{m} \sum_{C_j \in C} \sum_{u,v \in C_j} \left(A_{uv} - \frac{d_u^{in} d_v^{out}}{m} \right), \tag{2.6}$$

where A_{uv} is defined conventionally to be 1 if there is an edge from v to u and zero otherwise. Note that indeed edges $j \to i$ make larger contributions to this expression if d_u^{in} and/or d_v^{out} is small.

Similar to the case of undirected graphs, we also define Theta-Modularity as follows:

$$Q^d(\theta) = \frac{1}{m} \sum_{C_j \in C} \sum_{u,v \in C_j} \left(A_{uv} - \theta \frac{d_u^{in} d_v^{out}}{m} \right). \tag{2.7}$$

Then for each community C_j, we will consider the vertices $u \in V(G) \setminus C_j$. We notice that if a vertex u added to the community C_j, the modulus of the cluster C_j changes by an amount of

$$\Delta Q^d(\theta)_{u,C_j} = \frac{1}{m} \sum_{w \in C_j} \left(A_{uw} - \theta \frac{d_u^{in} d_w^{out}}{m} \right) + \frac{1}{m} \sum_{w \in C_j} \left(A_{wu} - \theta \frac{d_w^{in} d_u^{out}}{m} \right). \tag{2.8}$$

From the above comment, we propose that vertex u belongs to the community C_j, is $\Delta Q^d(\theta)_{u,C_j}$ is positive:

$$\frac{1}{m} \sum_{w \in C_j} \left(A_{uw} - \theta \frac{d_u^{in} d_w^{out}}{m} \right) + \frac{1}{m} \sum_{w \in C_j} \left(A_{wu} - \theta \frac{d_w^{in} d_u^{out}}{m} \right) > 0. \tag{2.9}$$

equivalent to

$$\sum_{w \in C_j} (A_{uw} + A_{wu}) > \theta \sum_{w \in C_j} \left(\frac{d_u^{in} d_w^{out}}{m} + \frac{d_w^{in} d_u^{out}}{m} \right), \tag{2.10}$$

Remark 2.2 *From (2.10), we have remarked that vertex u belongs to the community C_j if the total number of edges from u to C_j and the number of edges from C_j to u is large enough.*

From there, we propose the overlap detection algorithm described in **Algorithm** 2. We will call this the **Directed Parameterized d-Modularity Overlap Algorithm** for directed graphs (or **Di-Paramet. d-Modul.** for short)

Algorithm 2: Directed Parameterized d-Modularity Overlap Algorithm

Input: A directed graph G and a threshold value θ
Output: Clusters of vertices with overlapping communities
Apply Louvain algorithm [12] to obtain the initial clustering of G into communities $C_1, C_2, ..., C_k$;
repeat
  foreach *vertex u* **do**
    foreach *community C_j adjacent to u and not containing u* **do**
      if $\sum_{w \in C_j} (A_{uw} + A_{wu}) > \theta \sum_{w \in C_j} \left(\frac{d_u^{in} d_w^{out}}{m} + \frac{d_w^{in} d_u^{out}}{m} \right)$ **then**
        Add u to C_j;
      end
    end
  end
until *No communities C_j meet the condition*;

Similar to the Algorithm 1, the Algorithm 2 also has a computational complexity of $O(kn)$, where n is the number of vertices of the graph, and k is the number of communities.

2.3 Overlapping community detection for directed graphs using the stationary distribution

Many modularities have been proposed for directed graphs, such as the Modularity in Formula 2.6. In many cases, those modularity proposals will lose the essential properties of directed graphs. Therefore, we [11] proposed a definition of modularity for directed graphs based on random walks and stationary distribution, which is a natural extension of modularity on undirected graphs. In detail, for a cover $C = \{C_1, C_2, ..., C_k\}$ of the directed graph G, our proposed modularity is the following.

$$Q^{sd} = \sum_{uv} \left(P_{vu}\phi_v - \phi_u\phi_v\right) \delta_{C_uC_v}, \tag{2.11}$$

where P_{vu} is the transition probability of the random walk process from v-th vertex to u-th vertex and $\phi = (\phi_1, \phi_2, ..., \phi_n)$ is stationary distribution stationary.

We also have the θ modularity as follows.

$$Q^{sd}(\theta) = \sum_{uv} \left(P_{vu}\phi_v - \theta\phi_u\phi_v\right) \delta_{C_uC_v}. \tag{2.12}$$

and then

$$Q^{sd}(\theta) = \sum_{C_j \in C} \sum_{u,v \in C_j} \left(P_{vu}\phi_v - \theta\phi_u\phi_v\right). \tag{2.13}$$

Then for each community C_j, we will consider the vertices $u \in V(G) \setminus C_j$: ì u is added to the community C_j the modularity of the cluster C_j changes by an amount of

$$\Delta Q^{sd}(\theta)_{u,C_j} = \sum_{w \in C_j} \left(\phi_u P_{uw} - \theta\phi_u\phi_w\right) + \sum_{w \in C_j} \left(\phi_w P_{wu} - \theta\phi_u\phi_w\right). \tag{2.14}$$

We also propose that vertex u belongs to community C_j if $\Delta Q^{sd}(\theta)_{u,C_j}$ is positive. Therefore, if u belongs to the community C_j, then we have

$$\sum_{w \in C_j} \left(\phi_u P_{uw} - \theta\phi_u\phi_w\right) + \sum_{w \in C_j} \left(\phi_w P_{wu} - \theta\phi_u\phi_w\right) > 0, \tag{2.15}$$

equivalent to

$$\sum_{w \in C_j} \left(\phi_u P_{uw} + \phi_w P_{wu}\right) > 2\theta \sum_{w \in C_j} \phi_u\phi_w. \tag{2.16}$$

Remark 2.3 *The formula (2.16) means that vertex u belongs to the community C_j if the sum of the probabilities from vertex u to the community C_j and the probabilities from community C_j to vertex u is large enough.*

From there, we also propose the **Algorithm** 3. We will call this the **Directed Parameterized sd-Modularity Overlap Algorithm** for directed graphs (or **Di-Paramet. sd-Modul.** for short).

Algorithm 3: Directed Parameterized sd-Modularity Overlap Algorithm

Input: A directed graph G and a threshold value θ
Output: Clusters of vertices with overlapping communities
Apply Louvain algorithm [12] to obtain the initial clustering of G into communities $C_1, C_2, ..., C_k$;
repeat
  foreach *vertex* u **do**
    foreach *community* C_j *adjacent to* u *and not containing* u **do**
      if $\sum_{w \in C_j} (\phi_u P_{uw} + \phi_w P_{wu}) > 2\theta \sum_{w \in C_j} \phi_u \phi_w$ **then**
        Add u to C_j;
      end
    end
  end
until *No communities* C_j *meet the condition*;

Algorithm 3 requires computing the stationary distribution and using two loops similar to Algorithm one and Algorithm 2. Various efficient computation algorithms exist to calculate the stationary distribution, such as the one presented in [9], which has a computational complexity of $\tilde{O}(m^{3/4}n + mn^{2/3})$, where the $\tilde{O}(n)$ notation suppresses polylogarithmic factors in n. Therefore, the total computational complexity of this Algorithm is the sum of $O(kn)$ and $\tilde{O}(m^{3/4}n+mn^{2/3})$, which equals $\tilde{O}(m^{3/4}n + mn^{2/3})$.

Remark 2.4 *The equation (2.4) for undirected graphs can be rewritten as follows.*

$$\frac{1}{2m} \sum_{w \in C_j} \left(A_{uw} - \theta \frac{d_u d_w}{2m} \right) + \frac{1}{2m} \sum_{w \in C_j} \left(A_{wu} - \theta \frac{d_u d_w}{2m} \right) > 0. \tag{2.17}$$

and because $A_{uw} = d_u P_{uw}$, $d = 2m$ $\phi_u = \frac{d_u}{d}$, *then from (2.17), we get*

$$\sum_{w \in C_j} \left(\frac{d_u}{d} P_{uw} - \theta \frac{d_u}{d} \frac{d_w}{d} \right) + \sum_{w \in C_j} \left(\frac{d_w}{d} P_{wu} - \theta \frac{d_u}{d} \frac{d_w}{d} \right) > 0, \tag{2.18}$$

which implies that

$$\sum_{w \in C_j} (\phi_u P_{uw} + \phi_w P_{wu}) > 2\theta \sum_{w \in C_j} \phi_u \phi_w. \tag{2.19}$$

From (2.16) and (2.19), we observe that the condition for a vertex u to belong to a community C_j that we propose in a directed graphs is a natural extension of that in undirected graphs through the random walk and stationary distribution.

3 Overlapping community detection using the cosine

In many algorithms, researchers have represented the vertices of a graph as vectors in space. Numerous studies have utilized the similarity measure between vertices, which is determined by the angle between these vectors (as seen in [39]). Specifically, it has been observed that two vertices belong to the same community when the angle formed by their respective vectors is small. Consequently, the cosine of the angle between them is approximately 1.

Building upon this concept, we propose the following algorithms: a vertex u belongs to a community C if the cosine of the angle between the vector of u and the vector of the center of C is approximately 1. In this case, the vector of the center of C is calculated by taking the average of the coordinates of all vertices within the community. Since the vectors corresponding to the vertices within the same community form small angles with the cluster center, they will also create small angles among themselves. Based on this principle, we introduce overlapping community detection algorithms in the subsequent subsections.

3.1 Overlapping community detection for undirected graphs

In [36], the authors noticed that: two vertices u and v, that are closed each other, tend to "see" all the other vertices in the same way, that means

$$P^t_{uw} \simeq P^t_{vw}, \text{ for all } w. \tag{3.1}$$

Then they defined the distance between them:

$$R_{uv}(t) := \sqrt{\sum_{w=1}^{n} \frac{(P^t_{uw} - P^t_{vw})^2}{d(k)}} = \|D^{-1/2}P^t_{u\bullet} - D^{-1/2}P^t_{v\bullet}\|. \tag{3.2}$$

Inspiring from this idea, we correspond each vertex u to the vector $D^{-1/2}P^t_{u\bullet}$,

$$Coord(u) := D^{-1/2}P^t_{u\bullet} = \left\{ \frac{P^t_{u1}}{d_1^{1/2}}, \frac{P^t_{u2}}{d_2^{1/2}}, ..., \frac{P^t_{un}}{d_n^{1/2}} \right\}. \tag{3.3}$$

From the equation 3.1, we can also observe that if two vertices u and v belong to the same community, the angle formed by the two vectors $Coord(u)$ and $Coord(v)$ will be pretty small. In other words

$$cosin(Coord(u), Coord(v)) \simeq 1. \tag{3.4}$$

Because the lengths of vectors are comparable, and using the cosine function provides an explicit evaluation by comparing to 1, we will use Equation 3.4 to determine whether two vertices are in the same cluster or not. From this comment, we propose the following **Algorithm** 4. We will call this the **Cosine Overlap Algorithm** for undirected graphs.

Algorithm 4: Cosine Overlap Algorithm

Input: Undirected graph G, parameter θ

Output: Clusters $C_1, C_2, ..., C_k$

Apply Louvain algorithm [40] to obtain initial clusters $C_1, C_2, ..., C_k$ of G; **foreach** *cluster* C_j **do**

Calculate the cluster center $Center_j$ as follows:

$$Center_j = \frac{1}{|C_j|} \sum_{u \in C_j} \text{Coord}(u), \tag{3.5}$$

where $\text{Coord}(u) = D^{-1/2} P^t_{u\bullet}$ is the coordinate vector of node u defined as

$$\text{Coord}(u) = \left\{ \frac{P^t_{u1}}{d_1^{1/2}}, \frac{P^t_{u2}}{d_2^{1/2}}, ..., \frac{P^t_{un}}{d_n^{1/2}} \right\}. \tag{3.6}$$

end

foreach *cluster* C_j **do**

foreach *vertex* u **do**

if $cosin(Coord(u), Center_j) > \theta$ **then**

Add vertex u to cluster C_j;

end

end

end

We will select the parameter θ based on the network's structure and the desired level of overlap. For networks with a well-defined structure, we can choose θ between 0.7 and 0.8. Otherwise, if the network lacks a clear community structure, we should select a value of θ lower than 0.7.

The computational complexity of the Louvain algorithm is $O(n \log n)$. Determining the coordinates of the vertices has a complexity of $O(n^3)$, while finding the center has a complexity of $O(n)$. The cosine calculation step has a complexity of $O(kn^2)$. Therefore, the overall computational complexity of this algorithm is $O(n^3)$.

3.2 Overlapping community detection for directed graphs

For a strongly connected digraph G, let $\Phi^{1/2} = diag[\sqrt{\phi_u}]$. Yanhua and Z. L. Zhang [41] defined the normalized digraph Laplacian matrix (Diplacian for short) $\Gamma = [\Gamma_{uv}]$ for the graph G as follows.

Definition 3.1 *([41, Definition 3.2]) The Diplacian* Γ *is defined as*

$$\Gamma = \Phi^{1/2}(I - P)\Phi^{-1/2}. \tag{3.7}$$

We perform singular value decomposition on the normalized Laplace matrix $\Gamma = U\Sigma V^T$, where $V = [V_1, V_2, ..., V_n]$. Next, we take $V_k = [V_1, V_2, \ldots, V_k]$. For each vertex u in the graph G, we [11] have coordinated as follows:

$$Coord(u) = \left(\phi_u^{-1/2} V_1(u), \ldots, \phi_u^{-1/2} V_k(u), \phi_u^{-1/2} U_1(u), \ldots, \phi_u^{-1/2} U_k(u) \right). \tag{3.8}$$

The same as the Algorithm in the case of undirected graphs, we can also observe that if two vertices u and v belong to the same community, the angle formed by the two vectors $Coord(u)$ and $Coord(v)$ will be pretty small. Therefore

$$cosin(Coord(u), Coord(v)) \simeq 1.$$

From this comment, we also propose the **Algorithm** 5 for directed graphs. We will call this the **Directed Cosine Overlap Algorithm** for undirected graphs (or **Di-Cosine Overlap Algorithm** for short).

Algorithm 5: Directed Cosine Overlap Algorithm

Input: Directed graph G, parameter θ
Output: Clusters $C_1, C_2, ..., C_k$
Apply our NL-PCA algorithm [11] to obtain initial clusters $C_1, C_2, ..., C_k$ of G;
foreach *cluster* C_j **do**
  Calculate the cluster center $Center_j$ as follows:

$$Center_j = \frac{1}{|C_j|} \sum_{u \in C_j} \text{Coord}(u), \tag{3.9}$$

  where the coordinate vector of node u is defined as

$$Coord(u) = \left(\phi_u^{-1/2} V_1(u), \ldots, \phi_u^{-1/2} V_k(u), \phi_u^{-1/2} U_1(u), \ldots, \phi_u^{-1/2} U_k(u)\right). \tag{3.10}$$

end
foreach *cluster* C_j **do**
  foreach *vertex* u **do**
    if $cosin(Coord(u), Center_j) > \theta$ **then**
      Add vertex u to cluster C_j;
    end
  end
end

We will also choose the parameter θ as in the Algorithm for undirected graphs. The computational complexity of the NL-PCA algorithm is $O(n^3)$. We have already found the coordinates of the vertices in NL-PCA algorithm, so there is no need to recalculate. The step to find the center of complexity is $O(kn)$. So, the computational complexity of this algorithm is $O(n^3)$.

4 Examples

We inlustrate our algorithms for explicit graphs as follows: the Parameterized Modularity Overlap Algorithm for undirected graph in Figure 1, the Cosine Overlap Algorithm for undirected graph in Figure 2, the Parameterized Modularity Overlap Algorithm for directed graphs in Figure 3.

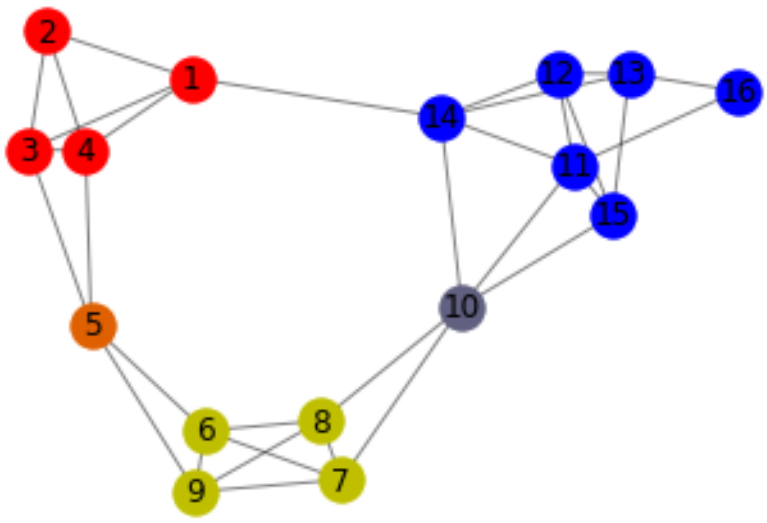

Figure 1: Applying the Parameterized Modularity Overlap Algorithm for undirected graphs with $\theta = 1$, we obtained three communities are $C_1 = \{1, 2, 3, 4, 5\}$, $C_2 = \{5, 6, 7, 8, 9, 10\}$, and $C_3 = \{10, 11, 12, 13, 14, 15, 16\}$. Vertex 5 belongs to communities C_1 and C_2. Vertex 10 belongs to both communities C_2 and C_3.

5 Experiments

We will evaluate the effectiveness and rationality of our algorithms by conducting experiments to compare and consider the clustering results of our algorithms with other algorithms and the clustering generated by the graph generation model. Specifically: In Subsection 5.2.2, we will compare two of our algorithms for undirected graphs, namely the Parameterized Modularity Overlap Algorithm (proposed in Subsection 2.1) and the Cosine Overlap Algorithm (proposed in Subsection 3.1), with two other algorithms, namely the Parameterized Overlap Algorithm [35] and the Module Overlap Algorithm [10]. We will compare these algorithms based on Modularity and the clustering generated by the graph generation model. Additionally, we will perform experiments on real datasets and compare the algorithms based on Modularity. Subsection 5.2.3, we will apply our directed graph algorithms to randomly generated graphs and compare the results with the clustering generated by the graph generation model.

5.1 Evaluating metrics

Modularity for undirected graphs with overlapping communities:

Chen et al. [10] provide the generalized modularity-based belonging function for calculating modularity in undirected graphs. The following equation represents this function:

$$Q = \frac{1}{2m} \sum_{C_j \in C} \sum_{u,v \in C_j} \left(A_{uv} - \frac{d_u d_v}{2m} \right) f\left(\alpha_{uC_j}, \alpha_{vC_j}\right). \tag{5.1}$$

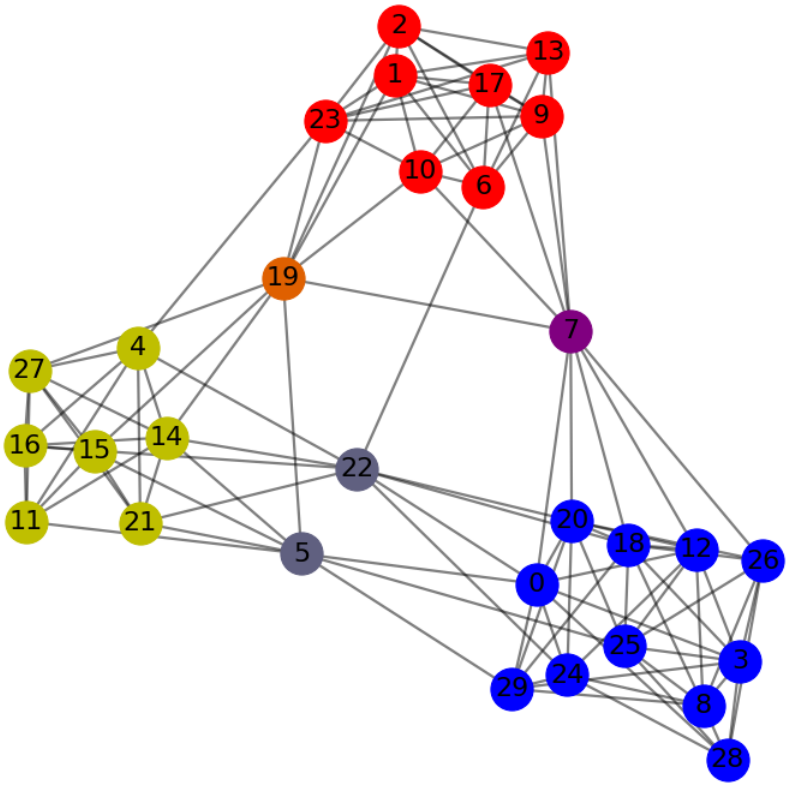

Figure 2: Applying the Cosine Overlap Algorithm with $\theta = 0.6$, we obtained three communities $C1 = \{4, 5, 11, 14, 15, 16, 21, 22, 27, 19\}$, $C2 = \{1, 2, 6, 7, 9, 10, 13, 17, 19, 23\}$ and $C3 = \{0, 3, 8, 12, 18, 20, 24, 25, 26, 28, 29, 5, 7, 22\}$. Vertex 19 belongs to communities C_1 and C_2. Vertex 7 belongs to both communities C_2 and C_3, and vertices 5, 22 belongs to both communities C_1 and C_3.

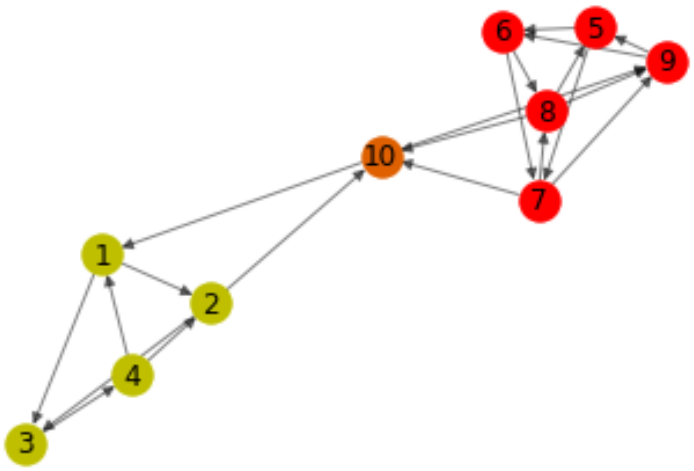

Figure 3: Applying the Parameterized Modularity Overlap Algorithm for directed graphs with $\theta = 1$, we obtained two communities $C_1 = \{1, 2, 3, 4, 10\}$ and $C_2 = \{5, 6, 7, 8, 9, 10\}$. Vertex 10 belongs to communities C_1 and C_2.

where α_{uC_j} is defined as in the Equation (1.2). The belonging coefficient function $f\left(\alpha_{uC_j}, \alpha_{vC_j}\right)$ can be the product or average of $\alpha_{uC_j}, \alpha_{vC_j}$. If it is average, it becomes the following equation.

$$Q = \frac{1}{2m} \sum_{C_j \in C} \sum_{u,v \in V} \left(A_{uv} - \frac{d_u d_v}{2m} \right) \left(\alpha_{uC_j} + \alpha_{uC_j} \right) /2. \tag{5.2}$$

In this part of the experiment, we will use this modularity to evaluate the clustering quality.

Modularity for directed graphs with overlapping communities:

To evaluate the effectiveness of overlapping community detection algorithms for directed graphs, we will use the modularity [30] given by the following formula.

$$Q_{ov} = \frac{1}{m} \sum_{C_j \in C} \sum_{u,v \in V} \left[\beta_{l(u,v),C_j} A_{uv} - \frac{\beta^{out}_{l(u,v),C_j} d^{\text{out}}_u \ \beta^{in}_{l(u,v),C_j} d^{in}_v}{m} \right], \tag{5.3}$$

where

$$\beta_{l(u,v),C_j} = \mathcal{F}\left(\alpha_{u,C_j}, \alpha_{v,C_j}\right), \tag{5.4}$$

and

$$\beta^{out}_{l(u,v),C_j} = \frac{\sum_{v \in V} \mathcal{F}(\alpha_{uC_j}, \alpha_{uC_j})}{|V|}, \ \ \beta^{in}_{l(u,v),c} = \frac{\sum_{u \in V} \mathcal{F}(\alpha_{uC_j}, \alpha_{uC_j})}{|V|}. \tag{5.5}$$

with definition of $\mathcal{F}(\alpha_{uC_j}, \alpha_{uC_j})$ is somewhat arbitrary. It is possible, for example, to define it as the product of the belonging coefficients of the nodes involved or as $\max\left(\alpha_{uC_j}, \alpha_{uC_j}\right)$. They don't make any choice about a particular form for $\mathcal{F}$. We will choose $\mathcal{F}$, same as in the case of an undirected graph, i.e., $\mathcal{F}(\alpha_{uC_j}, \alpha_{uC_j}) = \left(\alpha_{uC_j} + \alpha_{uC_j}\right)/2$.

5.2 The random graph model and experiments on random graphs

5.2.1 The random graph model

Evaluating a community detection algorithm is difficult because one needs some test graphs whose community structure is already known. A classical approach is to use randomly generated graphs with labeled communities. Here we will use this approach and generate the graphs as follows.

LFR benchmark graphs: This random graph generator model creates community-structured graphs with overlapping vertices. Andrea Lancichinetti and Santo Fortunato proposed it in [24]. This model generates graphs with many of the same properties as real networks. To create the graphs, we need the following parameters:

- N: number of nodes.
- k: average degree.
- maxk: maximum degree.
- mu: mixing parameter.
- t_1: minus exponent for the degree sequence.
- t_2: minus exponent for the community size distribution.
- minc: minimum for the community sizes.
- maxc: maximum for the community sizes.
- on: number of overlapping nodes.
- om: number of memberships of the overlapping nodes.
- C: average clustering coefficient.

When applying, people often use the parameters $t_1 = 2$, $t_2 = 1$, and $mu = 0.05$. So in this paper, we also default the values of these parameters as such. This random graph generation model can generate both undirected and directed graphs.

5.2.2 Experiments on random graphs for undirected graphs

In this part of the experiment, each table results from experiments on ten randomly generated graphs using the LFR benchmark graphs mode. We will conduct experiments on all four algorithms for the ten graphs generated. For each Algorithm, we will perform 20 experiments corresponding to 20 different parameters and select the clustering result with the highest Modularity among those experiments. Specifically, the parameters for each Algorithm will be as follows:

- Parameterized Modularity Overlap Algorithm: we will use the coefficient $\theta = 1 + 0.1t$ with $t \in \{1, 2, \ldots, 20\}$.
- Cosine Overlap Algorithm: we will use the coefficient $\theta = 0.2 + 0.035t$ with $t \in \{1, 2, \ldots, 20\}$.

- Parameterized Overlap Algorithm: we will use the coefficient $\theta = 0.2 + 0.015t$ with $t \in \{1, 2, \ldots, 20\}$.
- Module Overlap Algorithm: we will set the coefficient B^U to 0.5 and use the coefficient $B^L = 0.2 + 0.015t$ with $t \in \{1, 2, \ldots, 20\}$.

To compare the algorithms' results, we will take the result with the maximum Modularity value (the Modularity we use is the Formula 5.2). Then draw a chart with the vertical axis being the value of maximum Modularity and the horizontal axis being the number of experiments.

Experiment 1 for undirected graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(200, 250)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(5, 7)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(7, 15)$.
- The number of vertices in the overlap we take is a uniform probability in the interval $(50, 70)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 3)$.
- The size of the smallest community we take is a uniform probability in the interval $(20, 25)$.
- The size of the largest community we take is a uniform probability in the interval $(25, 30)$.

We have illustrated the results obtained from this experiment as shown in Figure 4. In the scenario where a graph consists of approximately 200 to 250 vertices, with an overlap of 50 to 70 vertices, our two algorithms have been shown to yield the maximum modularity value in most cases. Among these algorithms, the Cosine Overlap Algorithm has the best results.

This experiment will also investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 1 in Table 1.

Experiment 2 for undirected graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(300, 350)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(6, 8)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(8, 16)$.

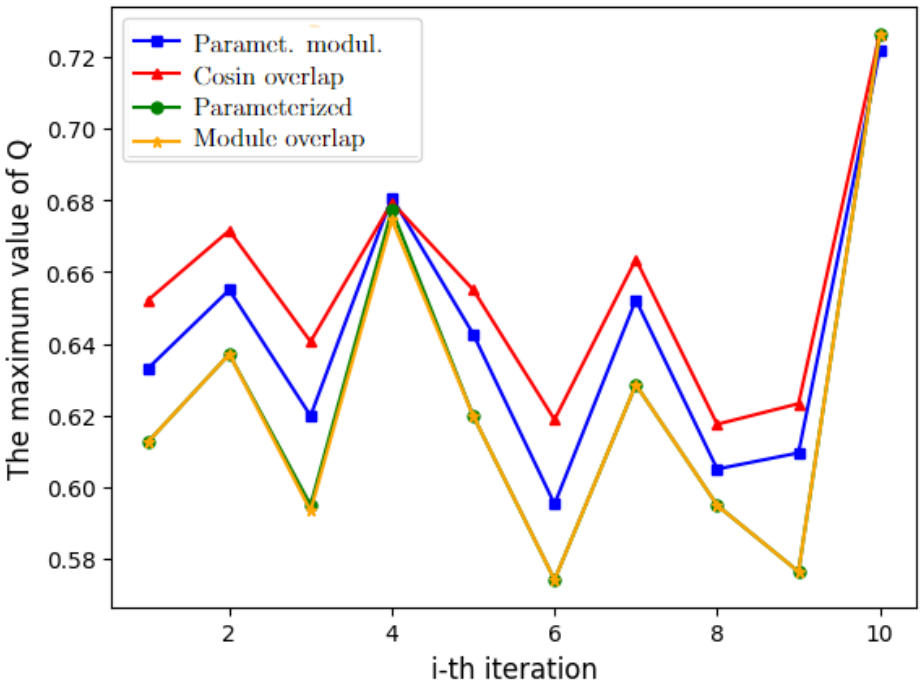

Figure 4: This chart illustrates the results of experiment 1 for undirected graphs; we experimented on ten randomly generated graphs using the LFR benchmark graphs model with the number of vertices in the interval $(200, 250)$, the largest community size in the interval $(25, 30)$, the smallest community size at least in the interval $(20, 25)$, the number of overlapping vertices in the interval $(50, 70)$.

	Original label	Paramet. Modul.	Cosine Overlap	Paramet. Overlap	Module Overlap
Exp. 1	$\mathcal{NO}$: 64	$\mathcal{NO}$: 67	$\mathcal{NO}$: 72	$\mathcal{NO}$: 19	$\mathcal{NO}$: 19
	N_3 : 64	N_2 : 51, N_3 : 15, N_4 : 1	N_2 : 36 N_3 : 29 N_4 : 6 N_5 : 1	N_2 : 18 N_3 : 1	N_2 : 18 N_3 : 1
Exp. 2	$\mathcal{NO}$: 53	$\mathcal{NO}$: 61	$\mathcal{NO}$: 56	$\mathcal{NO}$: 25	$\mathcal{NO}$: 25
	N_3 : 53	N_2 : 49 N_3 : 12	N_2 : 17 N_3 : 35 N_4 : 4	N_2 : 25	N_2 : 25
Exp. 3	$\mathcal{NO}$: 56	$\mathcal{NO}$: 92	$\mathcal{NO}$: 65	$\mathcal{NO}$: 18	$\mathcal{NO}$: 18
	N_3 : 56	N_2 : 72 N_3 : 16 N_4 : 4	N_2 : 29 N_3 : 32 N_4 : 4	N_2 : 18	N_2 : 16
Exp. 4	$\mathcal{NO}$: 53	$\mathcal{NO}$: 54	$\mathcal{NO}$: 53	$\mathcal{NO}$: 48	$\mathcal{NO}$: 41
	N_2 : 53	N_2 : 54	N_2 : 53	N_2 : 48	N_2 : 41
Exp. 5	$\mathcal{NO}$: 63	$\mathcal{NO}$: 78	$\mathcal{NO}$: 72	$\mathcal{NO}$: 26	$\mathcal{NO}$: 26
	N_3 : 63	N_2 : 51 N_3 : 25 N_4 : 2	N_2 : 25 N_3 : 42 N_4 : 3	N_2 : 25 N_3 : 1	N_2 : 25 N_3 : 1
Exp. 6	$\mathcal{NO}$: 65	$\mathcal{NO}$: 72	$\mathcal{NO}$: 73	$\mathcal{NO}$: 20	$\mathcal{NO}$: 20
	N_3 : 65	N_2 : 54 N_3 : 18	N_2 : 25 N_3 : 42 N_4 : 6	N_2 : 20	N_2 : 20
Exp. 7	$\mathcal{NO}$: 50	$\mathcal{NO}$: 47	$\mathcal{NO}$: 51	$\mathcal{NO}$: 16	$\mathcal{NO}$: 16
	N_3 : 50	N_2 : 28 N_3 : 19	N_2 : 22 N_3 : 27 N_4 : 2	N_2 : 15 N_3 : 1	N_2 : 15 N_3 : 1
Exp. 8	$\mathcal{NO}$: 65	$\mathcal{NO}$: 75	$\mathcal{NO}$: 69	$\mathcal{NO}$: 31	$\mathcal{NO}$: 31
	N_3 : 65	N_2 : 61 N_3 : 14	N_2 : 29 N_3 : 37 N_4 : 3	N_2 : 31	N_2 : 31
Exp. 9	$\mathcal{NO}$: 64	$\mathcal{NO}$: 65	$\mathcal{NO}$: 66	$\mathcal{NO}$: 21	$\mathcal{NO}$: 21
	N_3 : 64	N_2 : 44 N_3 : 21	N_2 : 29 N_3 : 35 N_4 : 1 N_5 : 1	N_2 : 21	N_2 : 21
Exp. 10	$\mathcal{NO}$: 52	$\mathcal{NO}$: 36	$\mathcal{NO}$: 47	$\mathcal{NO}$: 45	$\mathcal{NO}$: 45
	N_2 : 52	N_2 : 36	N_2 : 47	N_2 : 45	N_2 : 45

Table 1: This table illustrates the results of experiment 1 for undirected graphs; We experimented on ten randomly generated graphs using the LFR benchmark graphs model with the number of vertices in the interval (200, 250) and overlapping vertices number in the interval (60, 80). Our Cosine Overlap Algorithm produces the most accurate results closest to the original label, and our Parameterized Modularity Overlap Algorithm performs well. Especially when each overlap vertex belongs to more than two communities, in these cases, our two algorithms demonstrate much higher efficiency than the other two algorithms.

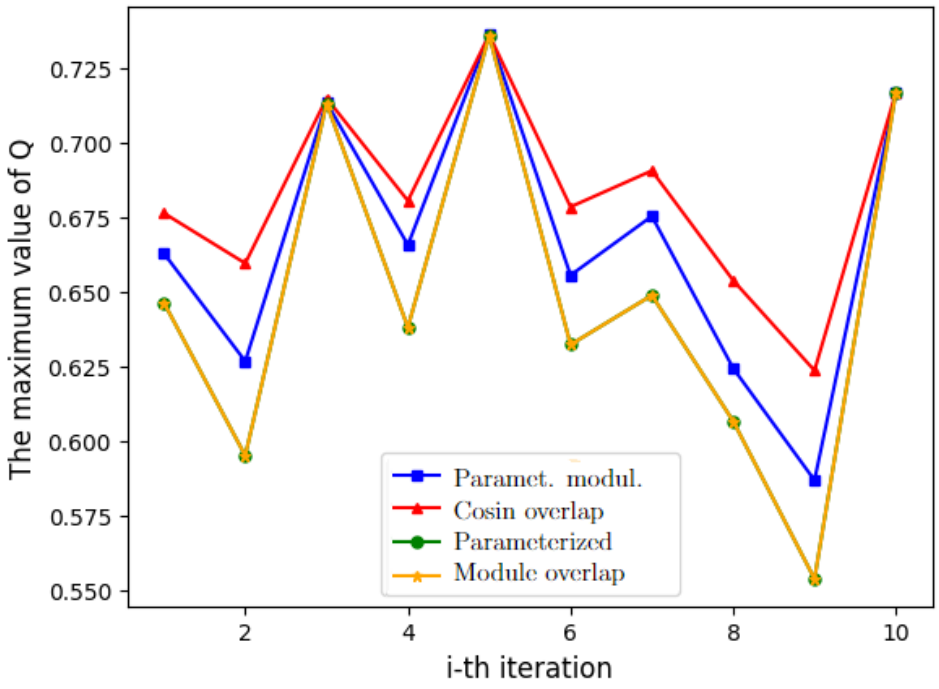

Figure 5: This chart illustrates the results of experiment 2 for undirected graphs; we experimented on ten randomly generated graphs using the LFR benchmark graphs model with the number of vertices in the interval $(300, 350)$, the largest community size in the interval $(35, 40)$, the smallest community size at least in the interval $(30, 35)$, the number of overlapping vertices in the interval $(60, 80)$.

- The number of vertices in the overlap we take is a uniform probability in the interval $(60, 80)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 5)$.
- The size of the smallest community we take is a uniform probability in the interval $(30, 35)$.
- The size of the largest community we take is a uniform probability in the interval $(35, 40)$.

We have illustrated the results obtained from this experiment as shown in Figure 5. In the scenario where a graph consists of approximately 300 to 350 vertices, with an overlap of 60 to 80 vertices, our two algorithms still have been shown to yield the maximum modularity value in most cases. Among these algorithms, the Cosine Overlap Algorithm has the best results.

This experiment will also investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 2 in Table 2.

	Original label	Paramet. Modul.	Cosine Overlap	Paramet. Overlap	Module Overlap
Exp. 1	$\mathcal{NO}$: 75	$\mathcal{NO}$: 103	$\mathcal{NO}$: 77	$\mathcal{NO}$: 35	$\mathcal{NO}$: 35
	N_3 : 75	N_2 : 76 N_3 : 26 N_4 : 1	N_2 : 30 N_3 : 5	N_2 : 30 N_3 : 5	N_3 : 57 N_2 : 17 N_4 : 3
Exp. 2	$\mathcal{NO}$: 79	$\mathcal{NO}$: 112	$\mathcal{NO}$: 80	$\mathcal{NO}$: 3	$\mathcal{NO}$: 3
	N_4 : 79	N_2 : 75 N_3 : 30 N_4 : 6 N_5 : 1	N_4 : 39 N_3 : 17 N_5 : 15 N_2 : 4 N_6 : 2 N_7 : 2 N_8 : 1	N_2 : 3	N_2 : 3
Exp. 3	$\mathcal{NO}$: 77	$\mathcal{NO}$: 78	$\mathcal{NO}$: 77	$\mathcal{NO}$: 65	$\mathcal{NO}$: 65
	N_2 : 77	N_2 : 76 N_3 : 2	N_2 : 76 N_3 : 1	N_2 : 65	N_2 : 65
Exp. 4	$\mathcal{NO}$: 68	$\mathcal{NO}$: 60	$\mathcal{NO}$: 71	$\mathcal{NO}$: 29	$\mathcal{NO}$: 29
	N_3 : 68	N_2 : 33 N_3 : 27	N_3 : 60 N_4 : 6 N_2 : 5	N_2 : 26 N_3 : 3	N_2 : 26 N_3 : 3
Exp. 5	$\mathcal{NO}$: 74	$\mathcal{NO}$: 74	$\mathcal{NO}$: 76	$\mathcal{NO}$: 72	$\mathcal{NO}$: 72
	N_2 : 74	N_2 : 74	N_2 : 76	N_2 : 72	N_2 : 72
Exp. 6	$\mathcal{NO}$: 76	$\mathcal{NO}$: 95	$\mathcal{NO}$: 78	$\mathcal{NO}$: 25	$\mathcal{NO}$: 25
	N_3 : 76	N_2 : 72 N_3 : 22 N_4 : 1	N_3 : 49 N_2 : 21 N_4 : 7 N_5 : 1	N_2 : 24 N_3 : 1	N_2 : 24 N_3 : 1
Exp. 7	$\mathcal{NO}$: 72	$\mathcal{NO}$: 87	$\mathcal{NO}$: 72	$\mathcal{NO}$: 40	$\mathcal{NO}$: 40
	N_3 : 72	N_2 : 33 N_3 : 7	N_3 : 65 N_2 : 5 N_4 : 2	N_2 : 33 N_3 : 7	N_2 : 33 N_3 : 7
Exp. 8	$\mathcal{NO}$: 67	$\mathcal{NO}$: 41	$\mathcal{NO}$: 70	$\mathcal{NO}$: 4	$\mathcal{NO}$: 4
	N_4 : 67	N_2 : 32 N_3 : 7 N_4 : 2	N_4 : 31 N_3 : 29 N_2 : 6 N_5 : 4	N_2 : 4	N_2 : 4
Exp. 9	$\mathcal{NO}$: 79	$\mathcal{NO}$: 101	$\mathcal{NO}$: 81	$\mathcal{NO}$: 3	$\mathcal{NO}$: 3
	N_4 : 79	N_2 : 75 N_3 : 23 N_4 : 3	N_3 : 39 N_4 : 28 N_2 : 10 N_5 : 4	N_2 : 3	N_2 : 3
Exp. 10	$\mathcal{NO}$: 62	$\mathcal{NO}$: 63	$\mathcal{NO}$: 62	$\mathcal{NO}$: 62	$\mathcal{NO}$: 63
	N_2 : 62	N_2 : 63	N_2 : 62	N_2 : 63	N_2 : 62

Table 2: This table illustrates the results of experiment 2 for undirected graphs; Even when we increase the number of vertices to 300 to 350, the results remain consistent with experiment 2. Our Cosine-overlap algorithm continues to provide the most precise results closest to the original label, while our Parameterized Modularity Overlap Algorithm also performs well. This is particularly true when each vertex is assigned to more than two communities, as our two algorithms demonstrate significantly higher efficiency than the other two algorithms.

Experiment 3 for undirected graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(400, 500)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(6, 8)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(8, 16)$.
- The number of vertices in the overlap we take is a uniform probability in the interval $(60, 80)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 5)$.
- The size of the smallest community we take is a uniform probability in the interval $(30, 35)$.
- The size of the largest community we take is a uniform probability in the interval $(35, 40)$.

Despite increasing the number of peaks in the graphs to approximately 400 to 500, we could replicate the same results from the previous two experiments. We present these results in Figure 6.

This experiment will also investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 3 in Table 3.

Experiment 4 for undirected graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(700, 900)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(6, 10)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(10, 20)$.
- The number of vertices in the overlap we take is a uniform probability in the interval $(60, 100)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 5)$.
- The size of the smallest community we take is a uniform probability in the interval $(40, 50)$.
- The size of the largest community we take is a uniform probability in the interval $(50, 70)$.

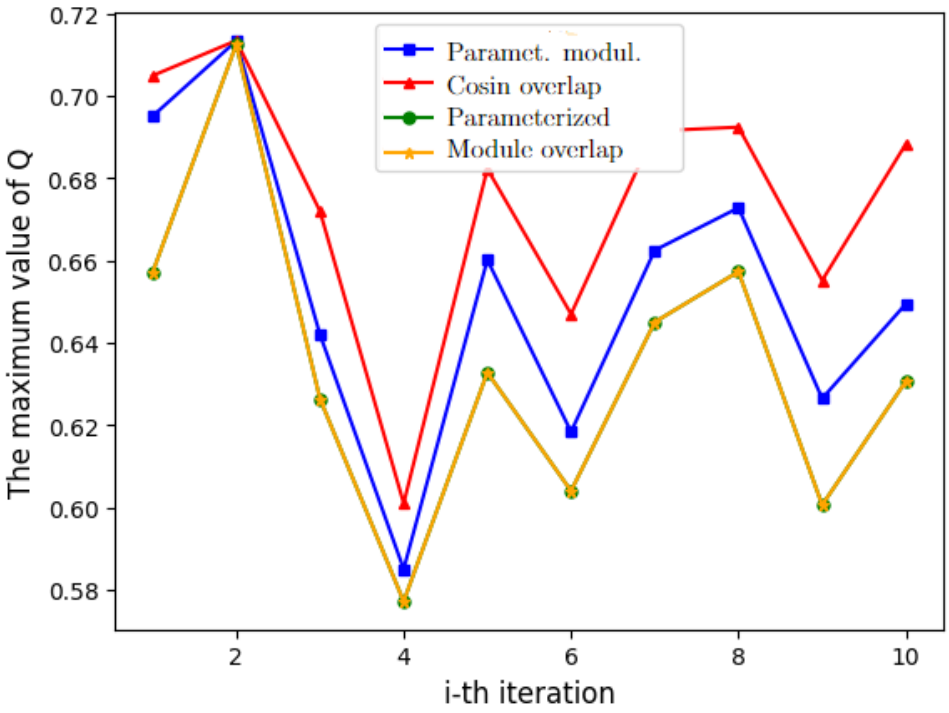

Figure 6: This chart illustrates the results of experiment 3 for undirected graphs; we experimented on ten randomly generated graphs using the LFR benchmark graphs model with the number of vertices in the interval $(400, 500)$, the largest community size in the interval $(35, 40)$, the smallest community size at least in the interval $(30, 35)$, the number of overlapping vertices in the interval $(60, 80)$.

	Original label	Paramet. Modul.	Cosine Overlap	Paramet. Overlap	Module Overlap
Exp. 1	$\mathcal{NO}$: 65	$\mathcal{NO}$: 69	$\mathcal{NO}$: 66	$\mathcal{NO}$: 26	$\mathcal{NO}$: 26
	N_3 : 65	N_3 : 45 N_2 : 24	N_3 : 58 N_2 : 5 N_4 : 3	N_2 : 24 N_3 : 2	N_2 : 24 N_3 : 2
Exp. 2	$\mathcal{NO}$: 61	$\mathcal{NO}$: 56	$\mathcal{NO}$: 55	$\mathcal{NO}$: 53	$\mathcal{NO}$: 53
	N_2 : 61	N_2 : 56	N_2 : 55	N_2 : 53	N_2 : 53
Exp. 3	$\mathcal{NO}$: 63	$\mathcal{NO}$: 69	$\mathcal{NO}$: 71	$\mathcal{NO}$: 3	$\mathcal{NO}$: 3
	N_4 : 63	N_2 : 60 N_3 : 8 N_4 : 1	N_3 : 33 N_4 : 23 N_2 : 10 N_5 : 5	N_2 : 3	N_2 : 3
Exp. 4	$\mathcal{NO}$: 75	$\mathcal{NO}$: 63	$\mathcal{NO}$: 68	$\mathcal{NO}$: 9	$\mathcal{NO}$: 9
	N_5 : 75	N_2 : 59 N_3 : 4	N_2 : 33 N_3 : 26 N_4 : 8 N_5 : 1	N_2 : 9	N_2 : 9
Exp. 5	$\mathcal{NO}$: 79	$\mathcal{NO}$: 75	$\mathcal{NO}$: 90	$\mathcal{NO}$: 24	$\mathcal{NO}$: 24
	N_3 : 75	N_2 : 52 N_3 : 23	N_3 : 59 N_2 : 26 N_4 : 5	N_2 : 23 N_3 : 1	N_2 : 23 N_3 : 1
Exp. 6	$\mathcal{NO}$: 68	$\mathcal{NO}$: 58	$\mathcal{NO}$: 73	$\mathcal{NO}$: 1	$\mathcal{NO}$: 1
	N_5 : 68	N_2 : 47 N_3 : 11	N_4 : 25 N_3 : 23 N_5 : 12 N_2 : 12 N_6 : 1	N_2 : 1	N_2 : 1
Exp. 7	$\mathcal{NO}$: 63	$\mathcal{NO}$: 59	$\mathcal{NO}$: 66	$\mathcal{NO}$: 2	$\mathcal{NO}$: 2
	N_5 : 63	N_2 : 50 N_3 : 8 N_4 : 1	N_4 : 25 N_3 : 19 N_5 : 13 N_2 : 8 N_6 : 1	N_2 : 2	N_2 : 2
Exp. 8	$\mathcal{NO}$: 62	$\mathcal{NO}$: 73	$\mathcal{NO}$: 70	$\mathcal{NO}$: 22	$\mathcal{NO}$: 22
	N_3 : 62	N_2 : 57 N_3 : 15 N_4 : 1	N_3 : 46 N_2 : 12 N_4 : 11 N_5 : 1	N_2 : 20 N_3 : 2	N_2 : 20 N_3 : 2
Exp. 9	$\mathcal{NO}$: 66	$\mathcal{NO}$: 69	$\mathcal{NO}$: 67	$\mathcal{NO}$: 2	$\mathcal{NO}$: 2
	N_4 : 66	N_2 : 52 N_3 : 17	N_3 : 32 N_4 : 24 N_2 : 6 N_5 : 5	N_2 : 2	N_2 : 2
Exp. 10	$\mathcal{NO}$: 67	$\mathcal{NO}$: 94	$\mathcal{NO}$: 72	$\mathcal{NO}$: 1	$\mathcal{NO}$: 1
	N_5 : 67	N_2 : 66 N_3 : 22 N_4 : 4 N_5 : 2	N_4 : 29 N_3 : 16 N_5 : 16 N_2 : 9 N_6 : 2	N_2 : 1	N_2 : 1

Table 3: This table illustrates the results of experiment 3 for undirected graphs; Our two undirected graph algorithms continued to achieve optimal results even when the number of vertices was increased from 400 to 500. Especially the Cosine Overlap Algorithm still gives the best results.

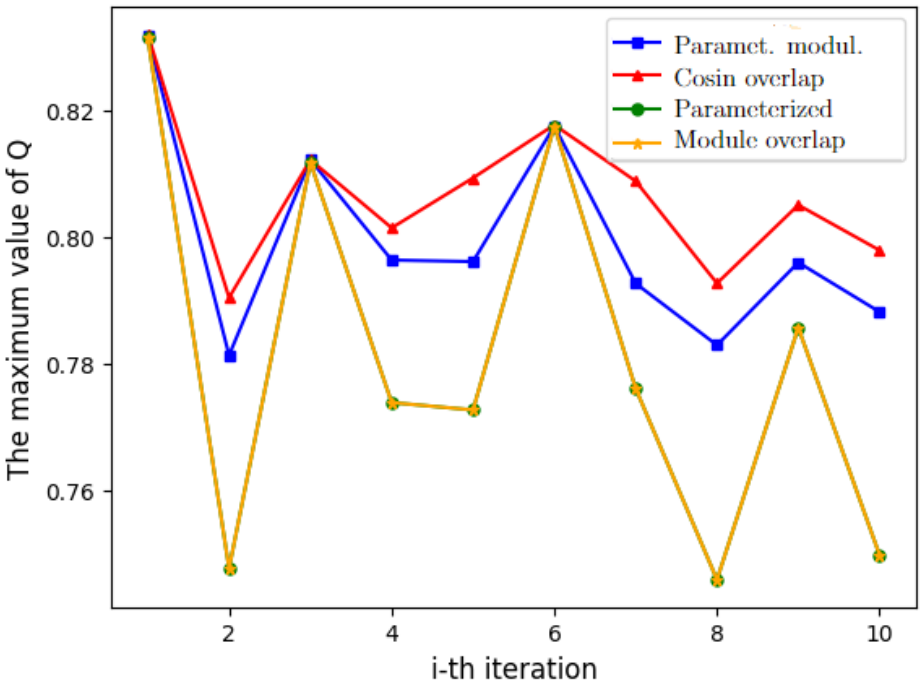

Figure 7: This chart illustrates the results of experiment 4 for undirected graphs; we experimented on ten randomly generated graphs using the LFR benchmark graphs model with the number of vertices in the interval $(700, 900)$, the largest community size in the interval $(50, 70)$, the smallest community size at least in the interval $(40, 50)$, the number of overlapping vertices in the interval $(60, 100)$.

Despite increasing the number of peaks in the graphs to approximately 700 to 900, we still could replicate the same results from the previous two experiments. We present these results in Figure 7.

This experiment will also investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 4 in Table 4.

5.2.3 Experiments on random graphs for directed graphs

In the following experiments, we will utilize our three algorithms (Directed Parameterized d-Modularity Overlap Algorithm, Directed Parameterized sd-Modularity Overlap Algorithm, and Directed Cosine Overlap Algorithm) for directed graphs to cluster random generation graphs. Subsequently, we will compare the index obtained from our algorithms with the graph generation index. The parameters for each Algorithm will be as follows:

- Directed Parameterized d-Modularity Overlap Algorithm: we will use the coefficient $\theta = 1 + 0.1t$ with $t \in \{1, 2, \ldots, 20\}$.
- Directed Parameterized sd-Modularity Overlap Algorithm: we will use the coefficient $\theta = 1 + 0.1t$ with $t \in \{1, 2, \ldots, 20\}$.
- Directed Cosine Overlap Algorithm: we will use the coefficient $\theta = 0.2 + 0.035t$ with $t \in \{1, 2, \ldots, 20\}$.

We will select the clustering result for each algorithm corresponding to the parameter value that obtained the highest modularity value (the Modularity we use is the Formula 5.3).

In the upcoming experiments, we will adopt the following notations: $\mathcal{NO}$ represents the total number of vertices in the overlap of a graph, and N_ℓ represents the number of vertices in the ℓ communities, where $\ell = 1, 2, 3....$

Experiment 1 for directed graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(50, 70)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(3, 5)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(5, 7)$.
- The number of vertices in the overlap we take is a uniform probability in the interval $(5, 7)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 3)$.
- The size of the smallest community we take is a uniform probability in the interval $(10, 15)$.
- The size of the largest community we take is a uniform probability in the interval $(15, 20)$.

	Original label	Paramet. Modul.	Cosine Overlap	Paramet. Overlap	Module Overlap
Exp. 1	$\mathcal{NO}$: 88	$\mathcal{NO}$: 90	$\mathcal{NO}$: 90	$\mathcal{NO}$: 85	$\mathcal{NO}$: 85
	N_2 : 88	N_2 : 90	N_2 : 89 N_3 : 1	N_2 : 85	N_2 : 85
Exp. 2	$\mathcal{NO}$: 74	$\mathcal{NO}$: 116	$\mathcal{NO}$: 74	$\mathcal{NO}$: 2	$\mathcal{NO}$: 2
	N_4 : 74	N_2 : 49 N_4 : 40 N_3 : 27	N_4 : 67 N_3 : 6 N_5 : 1 N_8 : 1	N_2 : 2	N_2 : 2
Exp. 3	$\mathcal{NO}$: 88	$\mathcal{NO}$: 88	$\mathcal{NO}$: 87	$\mathcal{NO}$: 87	$\mathcal{NO}$: 87
	N_2 : 88	N_2 : 88	N_2 : 88	N_2 : 87	N_2 : 87
Exp. 4	$\mathcal{NO}$: 84	$\mathcal{NO}$: 85	$\mathcal{NO}$: 85	$\mathcal{NO}$: 28	$\mathcal{NO}$: 28
	N_3 : 84	N_3 : 50 N_2 : 35	N_3 : 76 N_2 : 6 N_4 : 3	N_2 : 27 N_3 : 1	N_2 : 27 N_3 : 1
Exp. 5	$\mathcal{NO}$: 90	$\mathcal{NO}$: 123	$\mathcal{NO}$: 89	$\mathcal{NO}$: 12	$\mathcal{NO}$: 12
	N_4 : 90	N_2 : 68 N_3 : 43 N_4 : 12	N_4 : 55 N_3 : 27 N_2 : 7	N_2 : 12	N_2 : 12
Exp. 6	$\mathcal{NO}$: 77	$\mathcal{NO}$: 77	$\mathcal{NO}$: 77	$\mathcal{NO}$: 75	$\mathcal{NO}$: 75
	N_2 : 77	N_2 : 77	N_2 : 77	N_2 : 75	N_2 : 75
Exp. 7	$\mathcal{NO}$: 71	$\mathcal{NO}$: 142	$\mathcal{NO}$: 71	$\mathcal{NO}$: 4	$\mathcal{NO}$: 4
	N_4 : 71	N_2 : 102 N_3 : 34 N_4 : 6	N_4 : 46 N_3 : 20 N_2 : 3 N_5 : 2	N_2 : 4	N_2 : 4
Exp. 8	$\mathcal{NO}$: 86	$\mathcal{NO}$: 99	$\mathcal{NO}$: 87	$\mathcal{NO}$: 0	$\mathcal{NO}$: 0
	N_4 : 86	N_3 : 36 N_4 : 34 N_2 : 29	N_4 : 78 N_3 : 7 N_2 : 1 N_5 : 1		
Exp. 9	$\mathcal{NO}$: 74	$\mathcal{NO}$: 82	$\mathcal{NO}$: 76	$\mathcal{NO}$: 45	$\mathcal{NO}$: 45
	N_3 : 74	N_2 : 53 N_3 : 29	N_3 : 58 N_2 : 16 N_4 : 2	N_2 : 40 N_3 : 5	N_2 : 40 N_3 : 5
Exp. 10	$\mathcal{NO}$: 84	$\mathcal{NO}$: 101	$\mathcal{NO}$: 85	$\mathcal{NO}$: 3	$\mathcal{NO}$: 3
	N_4 : 84	N_4 : 36 N_3 : 35 N_2 : 30	N_4 : 80 N_3 : 4 N_2 : 1	N_2 : 3	N_2 : 3

Table 4: This table illustrates the results of experiment 4 for undirected graphs; Our two undirected graph algorithms continued to achieve optimal results even when the number of vertices was increased from 700 to 900, and the Cosine Overlap Algorithm still gives the best results.

	Original label	Di-Paramet. d-Modul.	Di-Cosine	Di-Paramet. sd-Modul.
Exp. 1	$\mathcal{NO} : 7$	$\mathcal{NO} : 7$	$\mathcal{NO} : 7$	$\mathcal{NO} : 5$
	$N_3 : 7$	$N_2 : 6\ N_3 : 1$	$N_2 : 4\ N_3 : 3$	$N_3 : 3\ N_2 : 2$
Exp. 2	$\mathcal{NO} : 7$	$\mathcal{NO} : 5$	$\mathcal{NO} : 8$	$\mathcal{NO} : 7$
	$N_3 : 7$	$N_2 : 3\ N_3 : 2$	$N_2 : 6\ N_3 : 2$	$N_2 : 5\ N_3 : 2$
Exp. 3	$\mathcal{NO} : 5$	$\mathcal{NO} : 6$	$\mathcal{NO} : 4$	$\mathcal{NO} : 5$
	$N_3 : 5$	$N_3 : 4\ N_2 : 2$	$N_2 : 3\ N_3 : 1$	$N_3 : 4\ N_2 : 1$
Exp. 4	$\mathcal{NO} : 6$	$\mathcal{NO} : 6$	$\mathcal{NO} : 6$	$\mathcal{NO} : 6$
	$N_2 : 6$	$N_2 : 6$	$N_2 : 6$	$N_2 : 6$
Exp. 5	$\mathcal{NO} : 5$	$\mathcal{NO} : 5$	$\mathcal{NO} : 5$	$\mathcal{NO} : 7$
	$N_2 : 5$	$N_2 : 5$	$N_2 : 5$	$N_2 : 7$
Exp. 6	$\mathcal{NO} : 6$	$\mathcal{NO} : 4$	$\mathcal{NO} : 6$	$\mathcal{NO} : 7$
	$N_3 : 6$	$N_2 : 4$	$N_2 : 4\ N_3 : 2$	$N_2 : 4\ N_3 : 3$
Exp. 7	$\mathcal{NO} : 7$	$\mathcal{NO} : 8$	$\mathcal{NO} : 7$	$\mathcal{NO} : 8$
	$N_3 : 7$	$N_2 : 8$	$N_2 : 7$	$N_2 : 7\ N_3 : 1$
Exp. 8	$\mathcal{NO} : 5$	$\mathcal{NO} : 6$	$\mathcal{NO} : 5$	$\mathcal{NO} : 6$
	$N_3 : 5$	$N_2 : 5\ N_3 : 1$	$N_2 : 4\ N_3 : 1$	$N_2 : 5\ N_3 : 1$
Exp. 9	$\mathcal{NO} : 6$	$\mathcal{NO} : 5$	$\mathcal{NO} : 7$	$\mathcal{NO} : 4$
	$N_3 : 6$	$N_2 : 4\ N_3 : 1$	$N_2 : 4\ N_3 : 3$	$N_2 : 2\ N_3 : 2$
Exp. 10	$\mathcal{NO} : 5$	$\mathcal{NO} : 8$	$\mathcal{NO} : 5$	$\mathcal{NO} : 8$
	$N_3 : 5$	$N_2 : 6\ N_3 : 2$	$N_2 : 3\ N_3 : 2$	$N_2 : 7\ N_3 : 1$

Table 5: Experimental 1 for directed graphs: Our Directed Cosine Overlap Algorithm yields the best results for graphs with a small number of vertices, ranging from 50 to 70 families, and overlapping vertices consisting of 5 to 7 nodes. Nonetheless, Our two different algorithms also exhibit a high level of accuracy.

This experiment will investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 1 in Table 5.

Experiment 2 for directed graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(100, 150)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(5, 7)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(7, 10)$.
- The number of vertices in the overlap we take is a uniform probability in the interval $(10, 20)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 4)$.

	Original label	Di-Paramet. d-Modul.	Di-Cosine	Di-Paramet. sd-Modul.
Exp. 1	$\mathcal{NO}:14$	$\mathcal{NO}:13$	$\mathcal{NO}:14$	$\mathcal{NO}:16$
	$N_3:14$	$N_2:12\ N_3:1$	$N_2:11\ N_3:3$	$N_2:16$
Exp. 2	$\mathcal{NO}:15$	$\mathcal{NO}:11$	$\mathcal{NO}:14$	$\mathcal{NO}:12$
	$N_2:15$	$N_2:11$	$N_2:14$	$N_2:12$
Exp. 3	$\mathcal{NO}:17$	$\mathcal{NO}:16$	$\mathcal{NO}:17$	$\mathcal{NO}:15$
	$N_3:17$	$N_2:14\ N_3:2$	$N_3:9\ N_2:8$	$N_2:12\ N_3:3$
Exp. 4	$\mathcal{NO}:19$	$\mathcal{NO}:6$	$\mathcal{NO}:6$	$\mathcal{NO}:6$
	$N_3:19$	$N_2:17\ N_3:1$	$N_3:9\ N_2:7$	$N_2:17\ N_3:2$
Exp. 5	$\mathcal{NO}:14$	$\mathcal{NO}:12$	$\mathcal{NO}:16$	$\mathcal{NO}:16$
	$N_2:5$	$N_2:12$	$N_2:16$	$N_2:16$
Exp. 6	$\mathcal{NO}:12$	$\mathcal{NO}:10$	$\mathcal{NO}:12$	$\mathcal{NO}:9$
	$N_2:12$	$N_2:10$	$N_2:12$	$N_2:9$
Exp. 7	$\mathcal{NO}:10$	$\mathcal{NO}:10$	$\mathcal{NO}:10$	$\mathcal{NO}:9$
	$N_2:10$	$N_2:10$	$N_2:10$	$N_2:9$
Exp. 8	$\mathcal{NO}:13$	$\mathcal{NO}:12$	$\mathcal{NO}:13$	$\mathcal{NO}:12$
	$N_3:13$	$N_2:10\ N_3:2$	$N_3:9\ N_2:4$	$N_3:6\ N_2:6$
Exp. 9	$\mathcal{NO}:15$	$\mathcal{NO}:13$	$\mathcal{NO}:14$	$\mathcal{NO}:13$
	$N_3:15$	$N_2:9\ N_3:4$	$N_3:9\ N_2:5$	$N_2:10\ N_3:3$
Exp. 10	$\mathcal{NO}:12$	$\mathcal{NO}:11$	$\mathcal{NO}:11$	$\mathcal{NO}:14$
	$N_2:12$	$N_2:11$	$N_2:11$	$N_2:14$

Table 6: Experimental 2 for directed graphs: Our observations reveal that even when the number of vertices in the graph increases from 100 to 150 and the number of overlapping vertices increases from 10 to 20 nodes, the Cosine Overlap Algorithm yields the best results. Furthermore, our other two algorithms also deliver highly accurate outcomes.

- The size of the smallest community we take is a uniform probability in the interval $(30, 40)$.
- The size of the largest community we take is a uniform probability in the interval $(40, 50)$.

This experiment will investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 2 in Table 6.

Experiment 3 for directed graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(200, 250)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(5, 7)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(7, 10)$.

- The number of vertices in the overlap we take is a uniform probability in the interval $(10, 25)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 4)$.
- The size of the smallest community we take is a uniform probability in the interval $(40, 50)$.
- The size of the largest community we take is a uniform probability in the interval $(50, 60)$.

This experiment will investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 3 in Table 7.

Experiment 4 for directed graphs:

We will experiment on ten randomly generated graphs using the LFR benchmark graphs mode with the following parameters:

- The number of vertices we take is a uniform probability in the interval $(300, 500)$.
- The mean degree of each vertex we take is a uniform probability in the interval $(7, 10)$.
- The maximum degree of the vertices we take is a uniform probability in the interval $(10, 15)$.
- The number of vertices in the overlap we take is a uniform probability in the interval $(25, 40)$.
- Each vertex in the overlap will be in the k community; we take k with a uniform probability in the interval $(2, 4)$.
- The size of the smallest community we take is a uniform probability in the interval $(50, 70)$.
- The size of the largest community we take is a uniform probability in the interval $(70, 80)$.

This experiment will investigate the number of overlapping vertices identified by the algorithms and compare it with those generated by the graph generation method. That will enable us to evaluate the efficiency of our Algorithm. We present the results of Experiment 4 in Table 8.

5.3 Real data and experiments on real data

5.3.1 Real data

In this paper, we will perform experiments on the following famous real data.

Zachary's karate club:

Wayne W. Zachary studied a social network of a karate club over three years from 1970 to 1972, as a paper in [43]. The network represents 34 members, recording the connections between pairs of members who had interactions beyond the club's premises. After being utilized by Michelle Girvan and Mark Newman in 2002 [16], the network became a widely-used example of community structure in networks.

	Original label	Di-Paramet. d-Modul.	Di-Cosine	Di-Paramet. sd-Modul.
Exp. 1	$\mathcal{NO}$: 23	$\mathcal{NO}$: 21	$\mathcal{NO}$: 24	$\mathcal{NO}$: 20
	N_2 : 23	N_2 : 21	N_2 : 22 N_3 : 2	N_2 : 20
Exp. 2	$\mathcal{NO}$: 19	$\mathcal{NO}$: 18	$\mathcal{NO}$: 16	$\mathcal{NO}$: 18
	N_2 : 19	N_2 : 18	N_2 : 16	N_2 : 18
Exp. 3	$\mathcal{NO}$: 13	$\mathcal{NO}$: 15	$\mathcal{NO}$: 13	$\mathcal{NO}$: 16
	N_4 : 13	N_3 : 11 N_2 : 4	N_3 : 9 N_2 : 3 N_4 : 1	N_3 : 12 N_2 : 4
Exp. 4	$\mathcal{NO}$: 17	$\mathcal{NO}$: 19	$\mathcal{NO}$: 17	$\mathcal{NO}$: 20
	N_4 : 17	N_3 : 10 N_2 : 9	N_3 : 10 N_2 : 7	N_3 : 10 N_2 : 10 N_2 : 6
Exp. 5	$\mathcal{NO}$: 16	$\mathcal{NO}$: 15	$\mathcal{NO}$: 19	$\mathcal{NO}$: 21
	N_4 : 16	N_3 : 7 N_2 : 6 N_4 : 2	N_2 : 9 N_3 : 7 N_4 : 3	N_2 : 14 N_3 : 5 N_4 : 2
Exp. 6	$\mathcal{NO}$: 24	$\mathcal{NO}$: 22	$\mathcal{NO}$: 22	$\mathcal{NO}$: 28
	N_3 : 24	N_2 : 12 N_3 : 10	N_2 : 17 N_3 : 5	N_2 : 17 N_3 : 11
Exp. 7	$\mathcal{NO}$: 16	$\mathcal{NO}$: 16	$\mathcal{NO}$: 16	$\mathcal{NO}$: 16
	N_2 : 16	N_2 : 16	N_2 : 16	N_2 : 16
Exp. 8	$\mathcal{NO}$: 23	$\mathcal{NO}$: 21	$\mathcal{NO}$: 23	$\mathcal{NO}$: 24
	N_4 : 23	N_2 : 14 N_3 : 5 N_4 : 2	N_2 : 10 N_3 : 10 N_4 : 3	N_2 : 16 N_3 : 8
Exp. 9	$\mathcal{NO}$: 25	$\mathcal{NO}$: 27	$\mathcal{NO}$: 27	$\mathcal{NO}$: 33
	N_4 : 25	N_3 : 12 N_2 : 10 N_4 : 5	N_2 : 12 N_3 : 11 N_4 : 4	N_2 : 15 N_3 : 11 N_4 : 7
Exp. 10	$\mathcal{NO}$: 24	$\mathcal{NO}$: 17	$\mathcal{NO}$: 26	$\mathcal{NO}$: 20
	N_3 : 24	N_2 : 15 N_3 : 2	N_2 : 16 N_3 : 10	N_2 : 18 N_3 : 2

Table 7: Experimental 3 for directed graphs: Continuing our investigation by gradually increasing the number of vertices in the graph to a range of 200-250 nodes and overlapping vertices from 10 to 25, we consistently obtain the same conclusion. The Cosine Overlap Algorithm remains the most effective, while our other two algorithms also exhibit high precision.

	Original label	Di-Paramet. d-Modul.	Di-Cosine	Di-Paramet. sd-Modul.
Exp. 1	$\mathcal{NO}$: 27	$\mathcal{NO}$: 30	$\mathcal{NO}$: 27	$\mathcal{NO}$: 29
	N_3 : 27	N_3 : 26 N_2 : 4	N_3 : 23 N_2 : 4	N_3 : 25 N_2 : 4
Exp. 2	$\mathcal{NO}$: 39	$\mathcal{NO}$: 41	$\mathcal{NO}$: 38	$\mathcal{NO}$: 42
	N_4 : 39	N_3 : 23 N_4 : 11 N_2 : 7	N_3 : 21 N_2 : 12 N_4 : 5	N_3 : 22 N_4 : 12 N_2 : 8
Exp. 3	$\mathcal{NO}$: 32	$\mathcal{NO}$: 32	$\mathcal{NO}$: 32	$\mathcal{NO}$: 34
	N_2 : 32	N_2 : 32	N_2 : 32	N_2 : 34
Exp. 4	$\mathcal{NO}$: 38	$\mathcal{NO}$: 37	$\mathcal{NO}$: 37	$\mathcal{NO}$: 37
	N_3 : 38	N_2 : 22 N_3 : 15	N_3 : 23 N_2 : 14	N_2 : 21 N_3 : 16
Exp. 5	$\mathcal{NO}$: 28	$\mathcal{NO}$: 28	$\mathcal{NO}$: 28	$\mathcal{NO}$: 30
	N_3 : 28	N_3 : 19 N_2 : 9	N_3 : 16 N_2 : 11 N_4 : 1	N_3 : 20 N_2 : 10
Exp. 6	$\mathcal{NO}$: 37	$\mathcal{NO}$: 37	$\mathcal{NO}$: 36	$\mathcal{NO}$: 45
	N_2 : 37	N_2 : 37	N_2 : 35 N_3 : 1	N_2 : 44 N_3 : 1
Exp. 7	$\mathcal{NO}$: 30	$\mathcal{NO}$: 31	$\mathcal{NO}$: 30	$\mathcal{NO}$: 30
	N_2 : 30	N_2 : 31	N_2 : 30	N_2 : 30
Exp. 8	$\mathcal{NO}$: 26	$\mathcal{NO}$: 25	$\mathcal{NO}$: 26	$\mathcal{NO}$: 27
	N_3 : 26	N_3 : 14 N_2 : 11	N_3 : 14 N_2 : 12	N_2 : 14 N_3 : 13
Exp. 9	$\mathcal{NO}$: 35	$\mathcal{NO}$: 38	$\mathcal{NO}$: 32	$\mathcal{NO}$: 39
	N_2 : 35	N_2 : 38	N_2 : 31 N_3 : 1	N_2 : 39
Exp. 10	$\mathcal{NO}$: 28	$\mathcal{NO}$: 30	$\mathcal{NO}$: 28	$\mathcal{NO}$: 29
	N_3 : 28	N_3 : 20 N_2 : 10	N_3 : 19 N_2 : 9	N_3 : 23 N_2 : 6

Table 8: Experimental 4 for directed graphs: Ultimately, we still obtain consistent outcomes when we increase the number of vertices in the graph from 300 to 500 and the number of overlapping vertices from 25 to 40. Our Cosine Overlap Algorithm remains the most effective, while the other two produce precise results.

Dolphin's associations:

The dataset used in this study was obtained from [25]. It describes the connections between 62 dolphins living in Doubtful Sound, New Zealand, where the links between pairs of dolphins represent statistically significant frequent associations. This network can be naturally divided into two distinct groups.

College football:

The college football network analyzed in [16] has been adopted as a standard benchmark for community detection. This network depicts the games played by Division I colleges during the regular season in the autumn of 2000, where each node represents a football team and each edge represents a regular season game. With 115 nodes and 616 edges, the network can be partitioned into 12 communities based on athletic conferences, with each community containing 8 to 12 teams.

Jazz network:

We obtained the data from The Red Hot Jazz Archive digital database [44]. Their analysis includes 198 bands that performed between 1912 and 1940, with most bands playing in the 1920s. The database lists the musicians who played in each band. Still, it needs to differentiate which musicians played at different times, making it impossible to study the temporal evolution of the collaboration network. The 1275 other names of musicians are distributed among the bands.

Metabolic network:

According to [19], a metabolic network represents the comprehensive collection of metabolic and physical processes that dictate a cell's physiological and biochemical characteristics. Therefore, these networks consist of metabolic reactions, pathways, and the regulatory interactions that direct these reactions.

Email network:

As stated in [17], the email network was created by analyzing email exchanges among approximately 1700 employees within a medium-sized university. Email networks are a reliable and non-intrusive way to depict how information flows within human organizations. The study revealed that the network organized itself into a state where the distribution of community sizes was self-similar.

In addition, we will also perform experiments on the following real data: Hamster households, hamster friendships, DNC co-recipient, and Asoiaf. These data we can see in [23].

5.3.2 Experiments on real data

We will conduct experiments on all four algorithms for each real network. We will perform 20 experiments with 20 different parameters for each Algorithm and select the clustering result with the highest Modularity among those experiments. Specifically, the parameters for each Algorithm will be set as follows:

Graph, $G = (\|V\|, \|E\|)$	Paramet. Modul. overlap	Cosine Overlap	Paramet. Overlap	Module Overlap
Dolphin's associations [25], $G = (62, 159)$	0.531895337674	0.53094150322	0.528294799784	0.53094150322
Zachary's karate club [43], $G = (34, 78)$	0.430041913215	0.419789612097	0.426799802761	0.42679980276
Metabolic network [19], $G = (453, 2025)$	0.453017231486	0.437497244045	0.4501564863	0.444779475254
College football [16], $G = (115, 613)$	0.610320160024	0.607710626963	0.605620296867	0.60440722891
Jazz network [18], $G = (198, 2742)$	0.4551599877295	0.448597518592	0.447319862619	0.444517929222
Email network [17], $G = (1133, 5451)$	0.590076882399	0.580598607432	0.591096281957	0.578600803260
Hamster households [23], $G = (921, 4032)$	0.38044958047	0.365814675655	0.378205207170	0.3774382660141
Hamsters friendships [23], $G = (1858, 12534)$	0.469201717136	0.4512991517167	0.453775622073	0.4547541325587
DNC corecipient [23], $G = (906, 10429)$	0.443415207455	0.44206605884	0.441983027638	0.443116266090
Asoiaf [23], $G = (796, 2823)$	0.606235436820	0.60978808427	0.611979108748	0.609218205843

Table 9: Table showing experimental results obtained from real data

- Parameterized Modularity Overlap Algorithm: we will use the coefficient $\theta = 1 + 0.1t$ with $t \in \{1, 2, \ldots, 20\}$.
- Cosine Overlap Algorithm: we will use the coefficient $\theta = 0.2 + 0.035t$ with $t \in \{1, 2, \ldots, 20\}$.
- Parameterized Overlap Algorithm: we will use the coefficient $\theta = 0.2 + 0.015t$ with $t \in \{1, 2, \ldots, 20\}$.
- Module Overlap Algorithm: we will set the coefficient B^U to 0.5 and use the coefficient $B^L = 0.2 + 0.015t$ with $t \in \{1, 2, \ldots, 20\}$.

5.4 Conclusion of the experiments

The above results show that our two algorithms are efficient in almost all experiments.

- Most experiments give better results for the Parameterized Modularity Overlap Algorithm than the Parameterized Overlap Algorithm and the Module Overlap Algorithm. Moreover, this Algorithm has the advantage of low computational complexity.
- We built The Cosine Overlap Algorithm by observing the network's relationship between random walks and community structure. Although the computational complexity will be greater than the Parameterized Modularity Overlap Algorithm, all experiments on the Cosine algorithm randomization graph are for the best results. Furthermore, the Algorithm makes a lot of sense in theory and is an interesting algorithm that deserves attention.
- Our algorithms for undirected graphs perform better than the other two algorithms, especially when each overlapping vertex belongs to more than two communities.
- Our directed graph algorithms demonstrate remarkable efficiency compared to the indices generated based on the graph generation method. Especially the Directed Cosine Overlap Algorithm, in most cases, gets the best results.

6 Conclusion and further work

In this paper, we have proposed two algorithms for overlapping community detection for undirected and directed graphs; our algorithms go through 2 steps. In step 1, we separate community detection using the algorithms we know, such as the Hitting times Walktrap algorithm, NL-PCA algorithm[11], Walktrap algorithm [36] or Louvain algorithm [40]. In step 2, we look for overlapping communities. Specifically, we have proposed the following two algorithms.

- The Parameterized Modularity Overlap Algorithm uses the idea that vertex u belongs to the community C_j if the sum of the probabilities from vertex u to the community C_j and the probabilities from community C_j to vertex u is large enough.
- In the Cosine Overlap Algorithm, we first coordinate the vertices of the graph, then find the centers of the clusters and use the idea that the vertex u belongs to the cluster C_j if the angle between the vector corresponds to the vertex u and the center of the cluster C_j is small.

In the future, we will continue to study the problem of finding overlapping communities based on two vertices belonging to a community through other criteria, such as using cut, distance, and cosine.

Acknowledgments

This research was supported by the International Center for Research and Postgraduate Training in Mathematics, Project code: NVCC01.02/23-24.

References

[1] L. A. N. Amaral, A. Scala, M. Barth'el'emy, and H. E. Stanley, Classes of small-world networks. Proc. Natl. Acad. Sci. USA 97, 11149–11152 (2000).

[2] A. Arenas, J. Duch, A. Fernandez, and S. Gomez. Size reduction of complex networks preserving Modularity. New Journal of Physics, 9:176, 2007.

[3] A. L. Barabasi and R. Albert, Emergence of scaling in random networks. Science 286, 509–512 (1999).

[4] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.- U. Hwang, Complex networks: Structure and dynamics.

[5] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN Syst. 30 (1-7) (1998) 107–117

[6] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts, Network robustness and fragility: Percolation on random graphs. Phys. Rev. Lett. 85, 5468–5471 (2000).

[7] A. Clauset, C. Moore, and M. E. J. Newman, Hierarchical structure and the prediction of missing links in networks. Nature 453, 98–101 (2008).

[8] R. Cohen, K. Erez, D. ben-Avraham, and S. Havlin, Re- silence of the Internet to random breakdowns. Phys. Rev. Lett. 85, 4626–4628 (2000).

[9] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Sidford, A. Vladu, Faster Algorithms for Computing the Stationary Distribution, Simulating Random Walks, and More, Annual IEEE Symposium on Foundations of Computer Science, 2016, Page(s):583 - 592.

[10] D. Chen, M. Shang, Z. Lu, and Y. Fu, Detecting Overlapping Communities of Weighted Networks via a Local Algorithm, Physica A: Statistical Mechanics and its Applications, vol. 389, no. 19, pp. 4177–4187, (2010).

[11] D. T. Dat, D. D. Hieu and P. T. H. Duong, Community detection in directed graphs using stationary distribution and hitting times methods, Social Network Analysis and Mining volume 13, Article number: 80 (2023).

[12] N. Dugué and A. Perez, Directed Louvain: maximizing modularity in directed networks. [Research Report] Université d'Orléans. 2015.

[13] B. S. Everitt, S. Landau, and M. Leese. Cluster Analysis. Hodder Arnold, London, 4th edition, 2001.

[14] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of web communities. Computer, 35(3):66-71, 2002.

[15] S. Fortunato, "Community detection in graphs," Physics Reports, vol. 486, pp. 75–174, 2010.

[16] M. Girvan, M.E.J Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 (2002) 7821–7826

[17] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. Phys. Rev. E, 68(6):065103, 2003.

[18] Gleiser, P. M. and Danon, L. Community structure in jazz. Adv. Complex. Syst. 6, 565–573 (2003).

[19] Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N. and Barabási, A.-L. The large-scale organization of metabolic networks. Nature 407, 651–654 (2000)

[20] S. Harenberg,G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, N. Samatova. Community detection in large-scale networks: a survey and empirical evaluation. WIREs Comput Stat 6(6):426–439, 2014.

[21] J. Kleinberg and S. Lawrence. The structure of the web. Science, 294(5548) : 1849 − 1850, 2001

[22] M. Jebabli, H. Cherifi,C. Cherifi, A. Hamouda. Community detection algorithm evaluation with ground-truth data. Physica A Stat Mech Appl 492:651–706.2018.

[23] J. Kunegis. KONECT – The Koblenz Network Collection. In Proc. Int. Conf. on World Wide Web Companion, pages 1343–1350, 2013.

[24] A. Lancichinetti and S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Phys. Rev. E 80, 016118 – Published 31 July 2009.

[25] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations, Behav. Ecol. Sociobiol. 54 (2003) 396–405.

[26] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. Physical Review Letter, 100:118703, 2008.

[27] M.E.J. Newman and M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026–113.

[28] M. E. J. Newman, The structure and function of complex networks. SIAM Review 45, 167–256 (2003).

[29] M. E. J. Newman, Assortative mixing in networks. Phys. Rev. Lett. 89, 208701 (2002).

[30] V. Nicosia, G. Mangioni, V. Carchiolo, M. Malgeri, Extending the definition of Modularity to directed graphs with overlapping communities, J. Stat. Mech. (2009) P03024.

[31] R. Pastor-Satorras, A. V´azquez, and A. Vespignani, Dynamical and correlation properties of the Internet. Phys. Rev. Lett. 87, 258701 (2001).

[32] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: Bringing order to the web, in: WWW '98: Proceedings of the 7th International World Wide Web Conference, 1998, pp. 161–172.

[33] L. Peel, DB. Larremore, A. Clauset The ground truth about metadata and community detection in networks. Sci Adv 3(5)(2017):e1602548.

[34] TP. Peixoto. Revealing consensus and dissensus between network partitions. https://doi.org/2005.13977, 2020.

[35] A. Ponomarenko, L. Pitsoulis, M. Shamshetdinov. Overlapping community detection in networks based on link partitioning and partitioning around medoids. PLoS One. 2021 Aug 25;16(8):e0255717.

[36] P. Pons and M. Latapy. Computing communities in large networks using random walks, Journal of Graph Algorithms and Applications, volume 10, no. 2, 2006, Pages 191–218, 2006.

[37] E. Ravasz and A.-L. Barab'asi, Hierarchical organization in complex networks. Phys. Rev. E 67, 026112 (2003).

[38] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical Organization of Modularity in Metabolic Networks. Science, 297(5586):15511555, 2002.

[39] S. Sarkar and A. Dong, Community detection in graphs using singular value decomposition, Phys. Rev. E 83, 046114 – Published 21 April 2011.

[40] BD. Vincent, G. Jean-Loup, L. Renaud and L. Etienne. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008 (10): P10008.

[41] L. Yanhua and Z. L. Zhang, Digraph Laplacian and the Degree of Asymmetry, nternet Mathematics Vol. 8, No. 4: 381–401, 2012.

[42] D. J. Watts and S. H. Strogatz, Collective dynamics of 'small-world' networks. Nature 393, 440–442 (1998).

[43] Zachary, W. An information flow model for conflict and fission in small groups. J. Anthropol. Res. 33, 452–473 (1977).

[44] The Red Hot Jazz Archive, available at http://www.redhotjazz.co