STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **7**, December 2019, pp 864–884.
Published online in International Academic Press (www.IAPress.org)

Community Detection in Social Networks Using Consensus Clustering

Masoumeh Kheirkhahzadeh , Morteza Analoui

Department of Computer Engineering, Iran University of Science and Technology, Iran

Abstract Community detection is one of the most appealing research fields in computer science. Although many different methods have been proposed to cluster the nodes of a graph, none of these methods is complete. Each method has strengths and weaknesses in extracting highly coherent groups of nodes (i.e. communities or clusters). The differences that various methods show are typically due to two main factors: 1) structure of the network they operate on, and 2) the strategy they use to find clusters. Since none of these methods is optimal, it seems a good idea to combine them to take advantage of their strengths while minimizing their weaknesses. In this paper, we present a new approach for the community detection problem by considering an ensemble of community detection methods. We refer to our approach as "*Mitra*". The base methods employed in *Mitra*, use different techniques and strategies to find communities for different applications of network data analysis. *Mitra* employs some known base community detection methods, receives their results on a network and builds a bipartite network to combine the communities found by the base methods. Then the fast projection technique compresses and summarizes the bipartite network to a new unipartite one. Then we find the communities of the unipartite network in the final step. We evaluate our approach against real and artificial datasets and compare our method with each one of the base methods. Artificial datasets include a diverse collection of large scale benchmark graphs. In this work, the main experimental evaluation function is Normalized Mutual Information (NMI). We also use several measures to compare the quality and properties of final community structures of the partitions found by all methods.

Keywords Community detection, Consensus clustering, Ensemble clustering, Bipartite networks.

AMS 2010 subject classifications 91C20, 94A13.

DOI: 10.19139/soic-2310-5070-801

1. Introduction

Detecting clusters or communities in real-world graphs such as large social networks, web graphs, and biological networks is a problem of considerable practical interest that has received a great deal of attention [33, 50, 29, 57, 16]. Community detection is used in particular to understand image segmentation, natural language processing, product-customer segmentation, web page sorting, sociological behavior, protein to protein interactions, gene expressions, recommendation systems, medical prognosis, DNA 3D folding and more [12, 13, 14]. Most recent approaches consider that a network community (also sometimes referred to as a module or cluster) is typically thought of as a group of nodes with more and/or better interactions amongst its members than between its members and the remainder of the network [50, 29, 57, 58, 16, 55].

More formally, a partition $P = \{C_1 , C_2,... , C_k\}$ of the vertices of a graph $G = (V, E)$ $(\forall i \in \{1, 2, ..., k\}, C_i \subseteq V)$ represents a good community structure if the proportion of edges inside the community C_i (internal edges) is high compared to the proportion of edges between them (see for example the definitions given in [32]). Identifying communities in a network can provide valuable information about the structural properties of the network, the interactions among nodes in the communities, and the role of the nodes in each community. Thorough reviews of different types of community detection algorithms can be found in [28, 64, 36, 41].

ISSN 2310-5070 (online) ISSN 2311-004X (print)
Copyright © 2019 International Academic Press

Several community detection algorithms give satisfactory results when they are tested on networks. Nevertheless, since some methods disclose serious limitations and biases, and most algorithms are likely to fail in some limit, an ideal method to detect communities in graphs, does not exist [30]. To overcome this problem, it seems a good idea to take advantage of the strengths of different existing methods to find a better solution and avoid the weaknesses of these methods. To reach this goal some approaches are proposed. One of them is ensemble clustering (also known as consensus clustering), that is inspired by ensemble learning, where multiple community detection algorithms run as an ensemble and the identified communities are then combined to improve the communities. Moreover, ensemble clustering is a technique used in some applications such as machine learning and is useful in merging several clustering results into one [25]. In this work, we use ensemble clustering to make a combination of the outputs of some famous community detection algorithms (called base classifiers or base methods), compress their results and get a consensus on the viewpoints of these algorithms on a community detection problem. This fusion process decreases the generalization error, because the more predictors differ, the better the performance of the ensemble is. This also explains why an ensemble of classifiers performs better than a more advanced single classifier, as the error rate can be decreased by increasing the number of classifiers included in the ensemble [62]. In order to create our consensus method *Mitra*, we select a number of broadly used community detection algorithms that perform well on graphs and get their results on the same network. A bipartite network is then generated to combine the information prepared by different base methods in the consensus pool. As the next step, a special projection [42] is applied on this bipartite network to convert it to a unipartite network, called consensus graph. At this step we detect communities of the consensus graph by one of the existing community detection algorithms.

We evaluate our approach on several artificial networks and compare it with each one of the methods used in the fusion. The comparison of the reference communities and the detected ones is according to NMI values [21] (see Subsection 5.1). In order to do more evaluations on the results, we investigate some quality scores on the detected communities. It is known that there is no single perfect quality metric for the comparison of the communities detected by different algorithms [15]. Therefore, we use a number of structural quality functions such as conductance [40] and modularity [51] to evaluate the quality of the detected communities. Before we proceed, it is worthwhile to clarify some nomenclature. We use "consensus clustering" and "ensemble clustering" interchangeably. Moreover, "cluster" is equivalent to "community".

The following of this paper is organized as follows. Section 2 reviews some related work on ensemble clustering. Our new approach *Mitra* is explained in 3, then the introduction of employed base community detection methods and our fast projection method are presented in this Section. Evaluation criteria are explained in Section 5. Important implementation results of the proposed method and comparison with base methods are reported in Section 6 and, finally the paper is concluded in Section 7.

2. Ensemble Clustering

In recent years a large number of methods for detecting community structures have been developed, drawing on knowledge from many different fields, e.g. computer science, statistical mechanics, discrete mathematics, statistics, and sociology. These methods have also been improved to handle weighted, directed, and multi graphs. For a comprehensive review of the field as a whole, the readers are referred to the references [31, 28, 64]. Another approach for improving community detection is to use ensemble clustering which is inspired by ensemble learning. In Ensemble clustering, multiple community detection algorithms run as an ensemble and the identified communities are combined to improve the community qualities. By considering an ensemble of clustering methods, it is possible to consider different definitions of community structure. In addition, more effective algorithms can be found by merging (aggregating) many runs of fast stochastic algorithms as well as several runs of the same algorithm using different settings. Moreover the latter method can be used to analyze the community structure of the network at many different scales, providing insight into the relations between community structures at different levels. The integration of consensus clustering with popular existing techniques leads to more accurate partitions than those delivered by the methods alone on LFR benchmark graphs [43]. Interestingly, this holds even

for methods whose direct application gives poor results on the same graphs, like modularity optimization [31]. In other words, to identify communities one is often confronted with the problem that one has a large number of potential partitions and often no good way to select a single best partition. Consensus clustering attempts to mitigate the problem by identifying common features of an ensemble of partitions [37].

One widely applied method for ensemble clustering is based on constructing a consensus graph. A consensus graph is composed of the set of partitions to be combined, returned by the base community detection algorithms[27, 60]. The consensus graph G_{cons} is defined over the set of nodes of G. Two nodes $v_i, v_j \in V$ are linked in G_{cons} if there is at least one partition where both nodes are in the same cluster. Each link (v_i, v_j) is weighted by the frequency of instances that nodes v_i, v_j are placed in the same cluster. The obtained graph is not necessarily a connected one. If there is not a priori information about the relative importance of the individual groupings, a reasonable goal for the ensemble result is to seek a clustering that shares the most information with the original clusterings[60]. Different approaches can be applied in order to compute the aggregated clustering out of the consensus graph. In the following paragraphs, some of these approaches are pointed out.
In [60], authors transform the consensus graph into a complete one by adding missing links with a null weight, then nodes are finally partitioned into clusters using agglomerative hierarchical clustering with some linkage rule, or by using a classical graph partitioning method such as the Kernighan-Lin algorithm [49]. In another work [52], by maximizing the overlap of the weak input partitions, the authors search for a strong partition. They show that if the process of restarting begin from maximal overlaps of the initial partitions, the quality of the initial weak partitions is not so important and the final result will be good. The point is if the base algorithms do not correspond on the communities, the method will not return any ensemble solution. The authors of [43] show that consensus clustering can be combined with any existing method in a self-consistent way, enhancing the stability and the accuracy of the resulting partitions. This method integrate consensus clustering in a giving method which is different from our approach to combine several different algorithms. The proposed ensemble clustering method in [38] works on the derived network partitions computed around a seed instead of growing communities around selected seeds. An ensemble ranking method to fuse different local modularity functions, computes the local communities. An iterative agglomerative algorithm expands the seeds. One of the drawbacks of this work is that the networks used for evaluating the method are all small networks. In [39] given a set of r base partitions, the authors compute an $r \times r$ pair-wise clustering similarity matrix M. A similarity graph GV is defined over the set of base partitions by using M. Then the given samples are clustered by a community detection algorithm applied on the similarity graph. A modularity-driven ensemble-based approach to multilayer community detection is proposed in [61] by aggregating the community structures separately generated for each network layer. The method is only proposed for multi layer networks and is different from our approach. The other difference is consensus methods use different specifications of network nodes to partition a network whereas, in this work all the base methods use the same specification of nodes. In [24] the proposed ensemble consists of only two clustering algorithms on some networks with simple structures (small values of μ as a network parameter) not for sophisticated networks. Using a modified version of modularity with a null model based on the ensemble of partitions for the consensus clustering step is presented in [37]. In this work the authors propose a hierarchical consensus clustering procedure, based on this modified modularity. A genetic algorithm for detecting a community structure in attributed graphs is proposed in [53]. The method optimizes a fitness function that combines node similarity and structural connectivity. The communities obtained by the method are composed by nodes having both similar attributes and high link density. The key distinguishing feature of our ensemble clustering procedure compared to these existing approaches is that we use different community detection algorithms instead of multiple runs of one algorithm. On the other hand, our approach is not agglomerative or extending method. A notable point is since some of the recent proposed ensemble approaches employ basically different strategies (e.g. using only one community detection algorithm instead of several different algorithms ([43, 52])) in comparison with *Mitra*, or ensemble approaches work on the other types of networks (such as multilayer networks [61]), or ensemble approaches work based on multiple specifications of network nodes [61] (not only one specification like our work), or designed to reach a dissimilar target function (such as [52]), comparison of *Mitra* with the recent similar approaches does not seem fair. Hence, we do not compare *Mitra* with other similar ensemble algorithms in our experiments.

In this paper, we perform an empirical comparison and quality evaluation of the communities identified by a variety of community detection algorithms. These algorithms use different strategies to find communities. We then compare their produced results with the results of our consensus clustering algorithm. Our approach is different from the others in some aspects, such as using different community detection algorithms instead of only one algorithm. The base algorithms are limited to use the same node specification unlike some of the similar approaches which use some different specifications. Moreover our approach is not an agglomerative or extending method. We propose to use a bipartite network as an intermediate network to fuse previously generated results and reach to a consensus graph by fast projection method[42]. The next Section explains our ensemble approach with more details.

3. Our Approach

In this section the overview of the proposed algorithm is presented. Then the based methods are introduced briefly and fast projection technique[42] is explained.

3.1. Our algorithm

Our approach "*Mitra*" consists of the following phases:

1. Apply $k-1$ base community detection algorithms $\{A_1, A_2,... , A_{k-1}\}$ on G to create a set $\mathcal{P}$ of partitions $\{P_1, P_2,... , P_{k-1}\}$ produced by these algorithms
2. Combine the partitions of $\mathcal{P}$ by constructing a bipartite graph G_b
3. Create unipartite graph G_u by applying fast projection on G_b
4. Apply A_k that is also one of the base community detection algorithms on G_u to find the final partition P^*

Figure 1 depicts the above algorithm.In the first phase, a number of popular community detection algorithms (called base methods) are applied to the network graph G to extract node communities, but One desired base method (called A_i) is kept for using in the last step. The set of base methods consists of algorithms that use different strategies to find the communities. The reason for selecting different methods is to add more diversity to the consensus, and consider different strategies or functions in order to find clusters. Each base method produces a partition P_i of the nodes independently. This phase can be done in a parallel manner if sufficient resources are available, to decrease the execution time. The base methods set includes CNM[19], Walktrap[54], CONCLUDE[22], Copra[35], Oslom[45], LPM[56], MCL[23], Infomap[57], Louvain[16]. The base methods are briefly introduced in subsection 3.2.
In the second phase, a bipartite network G_b is created based on the original network G. The bipartite network G_b includes two types of nodes: the real nodes of G as primary nodes and, the secondary nodes associated with the communities found by the base community detection methods. In other words, for each base method we add as many community nodes as the communities found by this base method. Then connect this community node to all its real node members in G_b. As a consequence, at the end of this step, G_b consists of all the real nodes of G and all the community nodes produced by all the base methods. G_b is a weighted graph. In G_b each community node divides probability 1 between all the real nodes connect to it evenly. The number of community nodes is $\sum_{i=1}^{k} |P_i|$, where $|P_i|$ is the number of communities in P_i, $i \in \{1, 2, ..., k\}$. As a result, the number of all the nodes of Gb is $N_b = N + \sum_{i=1}^{k} |P_i|$ where N is the number of real nodes of G.
In the third step, G_b is reduced to a unipartite network G_u by fast projection technique [42]. In fact G_u is our consensus graph. In other words, only the real nodes of G are kept and the community nodes are removed. The removal process of community nodes is done in such a manner that the effect of the removed community nodes is kept by the weights of links of G_u. Therefore G_u consists of the real nodes of G, but the links of G_u are different from the links of G. In other words, the links of G_u show the compressed result produced by the partitions of all the base methods. It is possible to construct G_u in four different ways: it may constructed as an undirected/directed and unweighted/weighted graph. Since fast projection builds links between primary nodes according to two-step walks, G_u is a directed graph. Moreover, fast projection selects links according to the highest probabilities. Consequently,

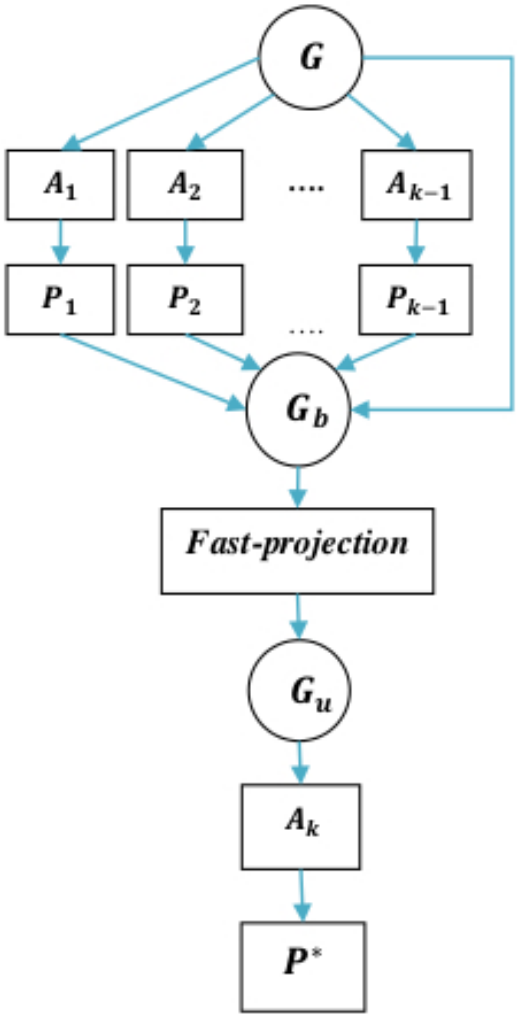


Figure 1. *Mitra* algorithm

G_u is a weighted graph.
In the last step, the final partition P^* is detected by applying A_k on G_u. A_k is one of the base algorithms not applied in the first step. Each base algorithm can be selected as A_k. Since G_u is essentially weighted and directed, some base methods are not capable of detecting communities on G_u. For example, CNM can only be applied to connected graphs, thus it is eliminated from the set of candidate methods which can apply in the last step. Also Walktrap is not able to work on directed graphs, therefore it could only be used in the last step if G_u considered undirected. There are some techniques for converting a directed graph to undirected one such as the techniques in Ref. [39]. We report the case of undirected and unweighted graph in our implementations. It is notable that some base methods are capable of finding overlapping communities, only the versions able to find disjoint communities are used.

To describe "*Mitra*" more formally, let $G =< V, E >$ be an undirected simple graph where V is the set of nodes and E is the set of edges. Suppose we have a set of base community detection algorithms $A = \{A_1, A_2, ..., A_k\}$. We apply algorithms $\{A_1, A_2, ..., A_{k-1}\}$ on G to generate a set of different partitions $P = \{P_1, P_2, ..., P_{k-1}\}$ defined over the same set V by $k-1$ algorithms respectively, i.e. P_i is a partition of the set V generated by

the base algorithm A_i applied on graph G. We have by definition $P_i = \{C_i^1, C_i^2, ..., C_i^l\}$ where $C_i^j \subseteq V$ and $\forall i, \bigcup_j C_i^j = V$ and $\forall j, q \in [1, l], C_i^j \bigcap C_i^k = \phi$. G_b is a bipartite graph generated to show the relations of real nodes and community nodes. G_b is reduced to a unipartite graph G_u that is the consensus graph. A_k is a member of base algorithms applied on the consensus graph G_u to produce consensus clustering P^*. We also consider P_G as the ground truth partition that is the reference partition we are looking for. We also consider average NMI value $\overline{NMI} = \frac{1}{k-1} \sum_{i=1}^{k-1} NMI(P_i, P_G)$. The goal of our ensemble clustering function is to reach to a consensus clustering P^* such that the number of disagreements between P^* and P_G is less than the number of disagreements between the average of P_i's and P_G. In other words, *Mitra* is not producing a simple mean of ensemble results and it is better than an average. In a formal way we have:

$$NMI(P^*, P_G) > \overline{NMI} \tag{1}$$

Where $NMI()$ is a function measuring the distance between two partitions [21]. NMI value equals 1 if the partitions are identical. it has an expected value of 0 if the partitions are independent. The closer the NMI value is to 1, the more similar the partitions are. NMI function is described formally in subsection 5.1. In the following the set of employed base algorithms is introduced.

3.2. Base Methods

In this work, a wide spectrum of community detection methods have experimented that we call base methods. The base methods come from a variety of different theoretical frameworks, as we try to select a set of community detection algorithms, which are comprehensive and exploit some of the most interesting ideas and techniques that have been developed over the last years. These base algorithms use different techniques such as modularity, label propagation, and random walk. Some of these algorithms guide their clustering by employing objective functions and some does not.

Base method set includes CNM[19], Walktrap[54], CONCLUDE[22], Copra[35], Oslom[45], LPM[56], MCL[23], Infomap[57], Louvain[16]. The implementation code for all the algorithms is publicly available in [3, 7, 9, 8, 11, 6, 1, 10, 4] respectively.

here is a short introduction of the base methods. CNM is based on maximization of modularity. Walktrap uses a measure of similarities between vertices based on random walks using modularity optimization. CONCLUDE generates random, non-backtracking walks of finite length to compute the importance of edge centralities. Copra is an extension of the label propagation technique. Oslom is based on the local optimization of a fitness function expressing the statistical significance of clusters with respect to random fluctuations. LPM simulates the spreading of labels based on the simple rule that at each iteration a given vertex takes the most frequent label in its neighborhood. MCL simulates a peculiar process of flow diffusion in a graph. Infomap uses a random walk as a proxy for information flow on a network and minimizes a map equation over all the network clusters. The goal of Louvain is the optimization of modularity, by means of a hierarchical approach.

3.3. Fast projection method

In the following, a bipartite network is defined and our fast projection method (that is applicable on bipartite networks) is explained. Then our algorithm is presented. A bipartite network is a graph whose nodes are divided into two disjoint sets such that no two graph nodes within the same set are adjacent. These node sets are called primary nodes and secondary nodes. In this work, the real nodes of the original network are considered as primary nodes and, community nodes are considered as secondary nodes in the bipartite network graph. Each community node is associated with a community that has links to its node members.

In the following steps the construction steps of a bipartite network G_b is presented which is associated with a simple network graph G. Bipartite graph G_b supposed to be a null graph firstly.

1. Add all the nodes of G to G_b as real nodes
2. Add a community node to G_b for each community detected by each base method applied on G

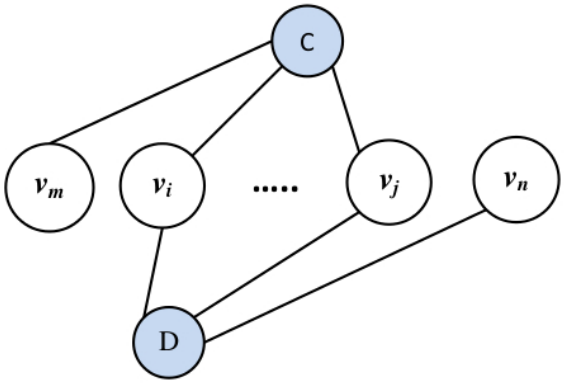


Figure 2. Real nodes v_i and v_j are connected to community nodes C and D

3. connect each community node to all the real nodes of that community
4. assign a weight to each link such that each community node divide weight 1 between its neighbors equally.

Fast projection approach is based on the sampling of important links. In fact most links in a weighted projection carry redundant information for community detection. Therefore, only the important non-redundant links must be sampled. Communities are considered as an indication of similarity of real nodes. Therefore, we apply fast projection technique to identify similar real nodes frequently visited in a sequence by a random walker on a bipartite network. In this work, to show the directed hidden relation between real nodes, the bipartite network is compressed by a one-mode projection called fast projection[42]. The projected network is a unipartite network. The unipartite network only include real nodes. Two nodes of a unipartite network are connected only if they have at least one common neighbor community node in the associated bipartite network.
In the fast projection technique, random walk is deployed to compress information of a bipartite network G_b and produce a unipartite one G_u. A random walker starts walking on G_b from a random real node to a community node, then from the community node to another real node by passing two consecutive links. Therefore, the walker is forced to pass a community node in each step, which is (one of)the common neighbors of the two real nodes.
Here an illustrative example is explained. let C be a detected community in the original graph G by a base method and v_i and v_j be two real nodes belong to C. In the process of building G_b one community node with label C and two links from C to v_i and v_j are added to G_b (see fig. 3a). Suppose C consists of five real nodes. Therefore, the weight of each link from C to the associated real nodes becomes 0.2. After generating G_b fast projection builds G_u from G_b. While applying fast projection, suppose a two-step random walker starts walking on G_b form v_i to C as the first pass and, from C to v_j as the second pass. Therefore, the weight of the candidate link between v_i and v_j becomes 0.4 in G_u. It is possible that other candidate link connects v_i and v_j with a different weight as a result of another community node say D (detected by another base method). At the end of fast projection, v_i and v_j are added to G_u. Moreover, the highest probability between v_i and v_j from the candidate set is selected as w_{ij}. Then a permanent weighted link is placed between v_i and v_j in G_u. The weight of this link is w_{ij}. Figure 2 clarify these concepts.
In detail, fast projection associates each community node with the top X real nodes selected by link weights. For each real node, we take the top X real nodes associated with each of its connected community nodes and include them in a candidate set. For each node in the candidate set, we compute the two-step random walk probability to go to other nodes also in the candidate set and create links to the top Y nodes.

4. Network datasets

To evaluate the performance of our method, two types of datasets are deployed, real and artificial datasets. Zachary [66], Polbooks [2] and Football [33] datasets are considered as real datasets. The datasets comprise 34, 105 and 115 nodes respectively. These datasets are famous and the most of researches in community detection field are tested on these networks. We also use the state-of-the-art artificial benchmark graphs with built-in community structures, i.e., the LFR benchmark [44]. The main reason for using benchmark graphs is the lack of ground truth information for the communities in real-world networks. LFR graphs are characterized by power law distributions of vertex degree and community size, features that are frequently observed in real world networks. An LFR graph has a clear community structure. Thus, it serves as a baseline reference for a network with known and detectable structure. LFR networks were created with standard LFR code [5]. One of the parameters of LFR networks is network size, that determines the number of nodes in the network. The other parameter is mixing parameter, that is a measure of the degree of fuzziness of the clusters. Mixing parameter μ is the ratio of the external degree of each node (with respect to the node cluster) divided by the total degree of the node, so it varies from 0 to 1. The values of μ close to zero correspond to well-separated clusters, whereas the values near 1 indicate a system with very mixed communities (thus hard to identify).

5. Evaluation criteria

The comparison between different clustering algorithms with our approach is done according to two aspects. First, we are interested in the structures of the clusters that various methods are able to find. Basically, we would like to understand how well algorithms perform in terms of optimizing the notion of NMI. In other words, maximization of NMI is the objective of this work. Second, we are interested in the quality of the clusters identified by the algorithms to clarify the behavior of the methods. These two aspects are explained in the following subsections.

5.1. Comparing community structures

Testing an algorithm on any graph with built-in community structure implies defining a quantitative criterion to estimate the goodness of the answer given by the algorithm as compared to the real answer that is expected. This can be done by using suitable similarity measures. For reviews of similarity measures see Refs. [18, 28, 48].

Here $Normalized\ Mutual\ Information(NMI)$ is selected, borrowed from information theory [21] which have proved to be a reliable measure to assess our approach. To evaluate the Shannon information content [47] of a partition, one starts by considering the community assignments x_i and y_i, where x_i and y_i indicate the cluster labels of vertex i in partition $\mathcal{X}$ and $mathcal(Y)$, respectively. One assumes that the labels x and y are values of two random variables X and Y, with joint distribution $P(x,y) = P(X = x, Y = y) = n_(xy)/n$, which implies that $P(x) = P(X = x) = n_x^X/n$ and $P(y) = P(Y = y) = n_y^Y/n$, where n_x^X, n_y^Y and n_{xy} are the sizes of the clusters labeled by x, y and of their overlap, respectively. The mutual information $I(X,Y)$ of two random variables is defined as

$$I(X,Y) = \sum_x \sum_y P(x,y)\ log \frac{P(x,y)}{P(x)P(y)} \tag{2}$$

The measure $I(X,Y)$ tells how much we learn about X if we know Y, and vice versa. Actually $I(X,Y) = H(X) - H(X|Y)$, where $H(X) = -\sum_x P(x)logP(x)$ is the Shannon entropy of X and $H(X|Y) = -\sum x, yP(x,y)logP(x|y)$ is the conditional entropy of X given Y. The mutual information is not ideal as a similarity measure: in fact, given a partition $\mathcal{X}$, all partitions derived from $\mathcal{X}$ by further partitioning (some of) its clusters would all have the same mutual information with $\mathcal{X}$, even though they could be very different from each other. In this case the mutual information would simply equal the entropy $H(X)$, because the conditional entropy would be systematically zero. To avoid that, Danon et al. adopted the normalized mutual information [21]

$$I_{norm}(\mathcal{X},\mathcal{Y}) = \frac{2I(X,Y)}{H(X) + H(Y)} \tag{3}$$

which equals 1 if the partitions are identical. It has an expected value of 0 if the partitions are independent. The normalized mutual information is currently very often used in tests of community detection algorithms. In our experiments we use $\max(H(X), H(Y))$ instead of $(H(X) + H(Y))/2$ in Eq.3, which is a strict version of NMI. Therefore, the following version of NMI is implemented in this work.

$$I_{norm}(\mathcal{X}, \mathcal{Y}) = \frac{I(X,Y)}{max(H(X), H(Y))} \tag{4}$$

5.2. Quality Scores

In order to investigate more properties of the methods, we analyze the quality of the communities detected by the base methods and *Mitra* according to some criteria. In general there are two criteria of interest when thinking about how good of a cluster a set of nodes is. The first is the number of internal edges between the members of a cluster, and the second is the number of external edges between members of the cluster and the remainder of the network [46]. This work follow the trend presented in [46]. In this trend there are two types of objective functions. The first group combines both criteria (number of edges inside and the number of edges crossing) into a single objective function; the second group employs only one of the two criteria (e.g., the number of edges cut).

Let $G(V, E)$ be an undirected graph with $n = |V|$ nodes and $m = |E|$ edges. Let S be the set of nodes in the cluster, where n_S is the number of nodes in S, $n_S = |S|$; m_S the number of edges in S, $m_S = |\{(u, v) : u \in S, v \in S\}|$; and c_S, the number of edges on the boundary of S, $c_S = |\{(u, v) : u \in S, v \notin S\}|$; and $d(u)$ is the degree of node u. The following functions $f(S)$ are considered to represent the quality of a cluster S.

- **Conductance:** $f(S) = \frac{c_S}{2m_S+c_S}$ measures the fraction of total edge volume that points outside the cluster [59, 40].
- **Expansion:** $f(S) = \frac{c_S}{n_S}$ measures the number of edges per node that point outside the cluster [55].
- **Internal density:** $f(S) = 1 - \frac{m_S}{n_S(n_S-1)/2}$ is the internal edge density of the cluster S [55].
- **Cut Ratio:** $f(S) = \frac{c_S}{n_S(n-n_S)}$ is the fraction of all possible edges leaving the cluster [28].
- **Normalized Cut:** $f(S) = \frac{c_S}{2m_S+c_S} + \frac{c_S}{2(m-m_S+c_S)}$ [59].

Although various functions are considered to model community scores, we will primarily work with conductance, that arguably is the simplest notion of cluster quality. Conductance can be simply thought of as the ratio between the number of edges inside the cluster and the number of edges leaving the cluster [59, 40, 46]. Many experiments have shown that Conductance is the best scoring function for networks with well-separated and non-overlapping communities [65, 46, 26]. More formally, conductance $\phi(S)$ of a set of nodes S is $\phi(S) = c_S/min(Vol(S), Vol(V \ S))$, where c_S denotes the size of the edge boundary, $c_S = |\{(u, v) : u \in S, v \notin S\}|$, and $Vol(S) = \Sigma_{u \in S} d(u)$, where $d(u)$ is the degree of node u. Thus, in particular, more community-like sets of nodes have lower conductance.

Now the following four notions of community quality are considered that are based on a single one of the two criteria mentioned at the beginning of this subsection:

- **Modularity:** $\frac{1}{4m}(m_S - E(m_S))$, where $E(m_S)$ is the expected number of edges between the nodes in set S in a random graph with the same node degree sequence.
- **Modularity ratio:** $m_S/E(m_S)$ is alternative definition of the modularity, where we take the ratio of the number of internal edges in S and $E(m_S)$, the expected number of such edges under the null-model.
- **Volume:** $\sum_{u_S} d(u)$ is sum of the degrees of nodes in S.
- **Edges cut:** c_S is number of edges needed to be removed to disconnect nodes in S from the rest of the network.

The most popular quality function is the modularity of Newman and Girvan [51]. Modularity measures the number of internal community edges, relative to a null model of a random graph with the same degree distribution. It is based on the idea that a random graph is not expected to have a cluster structure, so the possible existence of clusters is revealed by the comparison between the actual density of edges in a subgraph and the density one would expect to have in the subgraph if the vertices of the graph were attached regardless of community structure. This expected edge density depends on the chosen null model, i.e. a copy of the original graph keeping some of its structural properties without a community structure. Although modularity has been shown to have a resolution limit [29], some of the most popular clustering algorithms use it as an objective function [16, 63]. Modularity is given by Eq.((5))

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

where the sum runs over all pairs of vertices, A is the adjacency matrix, m the total number of edges of the graph, and k_i represents the degree of the node i. The δ-function yields one if vertices i and j are in the same community $(C_i = C_j)$, zero otherwise.

Since the only contributions to the sum come from vertex pairs belonging to the same cluster, it is a good idea to group these contributions together and rewrite the sum over the vertex pairs as a sum over the clusters [28]:

$$Q = \sum_{S=1}^{k}[\frac{m_S}{m} - (\frac{d_S}{2m})^2] \tag{6}$$

Here, k is the number of clusters, m_S the total number of edges joining vertices of module S and d_S the sum of the degrees of the vertices of S. In Eq.((6)), the first term of each summand is the fraction of edges of the graph inside the module, whereas the second term represents the expected fraction of edges that would be there if the graph were a random graph with the same expected degree for each vertex. The reader is referred to Ref. [26] for a good example of modularity computation.

A higher modularity is often taken as an indication of a better community structure as it is different from the random null model. As such, the modularity can only be used as a comparative measure [20] and has drawbacks, for example random networks usually have higher maximum modularity than real-world networks, and the resolution limit [29]. It can be shown that modularity optimization, i.e. finding the optimal community structure, is an NP-complete problem [17]. It is also possible to show that modularity has many local maxima [34] making the identification of a global maximum very difficult.

6. Experimental results

Experimental comparisons of different community detection algorithms have been conducted in this Section. We compare Louvain, Infomap, CNM, Walktrap, LPM, MCL, Copra, Oslom and CONCLUDE algorithms with our *Mitra* algorithm, on real datasets and LFR benchmark graphs. Infomap is used as the last base community detection method in our implementations. However, as explained in section 3 all the other base methods are usable as the last base method. In the following, the implementation of LFR benchmarks are explained and, NMI results are then provided. After that the results of quality scores for real and artificial networks are reported.

6.1. LFR Dataset Implementation

Each LFR network is associated by some parameters such as network size N and mixing parameter μ which determine the structural properties of the network. We implement our method against the benchmark graphs with three network sizes N =1000, 5000 and 10000 nodes. Mixing parameter μ get a value in range $\{0.1, 0.2, ..., 0.9\}$. We generate 100 realizations of the LFR benchmark for each value of μ and for each network size, creating 2700 graphs (i.e. 100 samples of each network for each pair of N and μ values). In other words, 900 network samples are produced for each network size (i.e. 100 samples for each value of μ). All the values in the diagrams and tables

are averaged over these 100 samples. All the generated networks are undirected and unweighted. Table 1 shows the parameters used for the generated networks. The construction method of an LFR netwrok is provided in [44]. The selected parameter values are based on the literature [44, 45]. As described later in LFR graphs, node degree (i.e. the number of edges connect to a node) and community size (i.e. the number of nodes belong to a community) follow power law distributions. As we show in Table 1, the community sizes are taken from a power law distribution with exponent β and each node is given a degree taken from a power law distribution with exponent γ. Depending on the size of the graph N, other parameters vary accordingly (see Table 1). For instance, one possible network size is 5000. Mixing parameter μ ranges in $\{0.1, 0.2, ..., 0.9\}$, β and γ get the constant values for all network configurations and, the second value column must be assigned to the remained parameters.

Variable	**Value**			**Description**
N	1000	5000	10000	number of nodes in the network
k_{avg}	20	25	50	average degree of nodes
k_{max}	50	100	150	maximum degree of nodes
c_{min}	10	20	20	minimum size of a community
c_{max}	50	100	100	maximum size of a community
μ	$\{0.1, 0.2, ..., 0.9\}$			mixing parameter
β	-1			exponent of community size distribution
γ	-2			exponent of degree distribution

Table 1. The parameters used for generating artificial networks in the simulation studies using the algorithm from Ref. [44]. The mixing parameter, μ is varied in range $\{0.1, 0.2, ..., 0.9\}$ for three network sizes.

6.2. NMI results

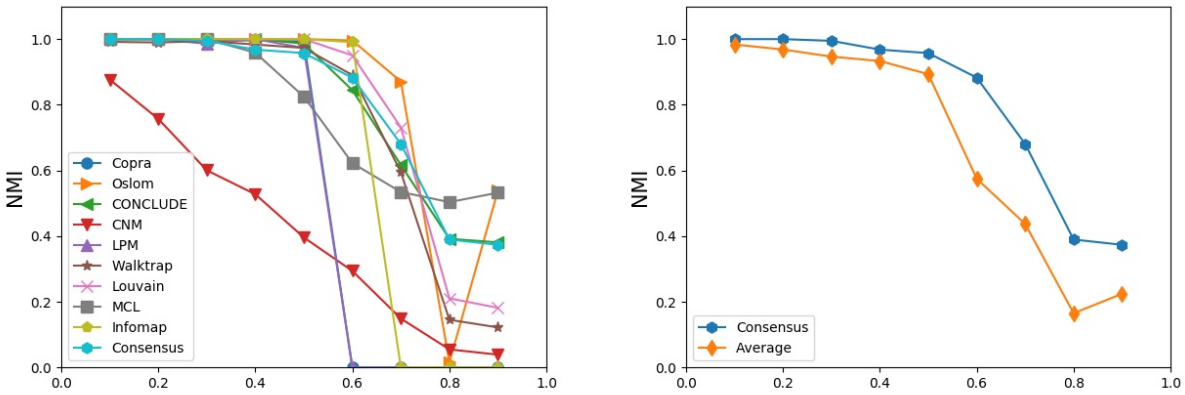
Experimental results on both real and artificial datasets are presented in this section.

6.2.1. Real datasets results Table 2 reports NMI values for all the base methods, *Mitra* and the average value produced by the base methods. All the methods are experimented on Zachary, Polbooks and Football datasets, some of the famous datasets in community detection field. The datasets contain 34, 105 and 115 nodes respectively. Average column in the table represents the mean NMI values produced by the base methods. It is observable in Table 2 that all the methods perform almost the same. The table did not show an interesting priority for *Mitra* in comparison with the base methods. However, these data are special cases in the world and, it is necessary to evaluate our method on more various datasets. Therefore, the artificial datasets are used in the most researches in our field and we also analyze them in the following part.

6.2.2. LFR dataset results In the following, the comparison between ground truth (reference) partition with the partitions produced by the base methods is presented, according to NMI. The base methods are Louvain, Infomap, CNM, Walktrap, LPM, MCL, Copra, Oslom, CONCLUDE. The comparison is applied on LFR benchmark graphs. All the base methods are applied on original graph (i.e. G) whereas, *Mitra* is applied on consensus graph (i.e. G_u). The average curve is the average NMI value of all the base methods. Figures 3-5 show the results for networks with 1000, 5000 and 10000 nodes respectively. For networks with N = 5000, 10000 nodes, community sizes range

NMI	**CNM**	**LPM**	**Walktrap**	**Louvain**	**Copra**	**MCL**	**Oslom**	**Infomap**	**Mitra**	**Average**
Zachary	0.57	0.83	0.59	0.46	0.31	0.25	0.83	0.59	0.83	0.55
Football	0.66	0.92	0.89	0.85	0.88	0.88	0.89	0.84	0.79	0.85
Polbooks	0.50	0.45	0.51	0.51	0.33	0.57	0.55	0.47	0.42	0.49

Table 2. NMI for real datasets



(a) NMI for the base methods and *Mitra*

(b) Comparison with average of base methods

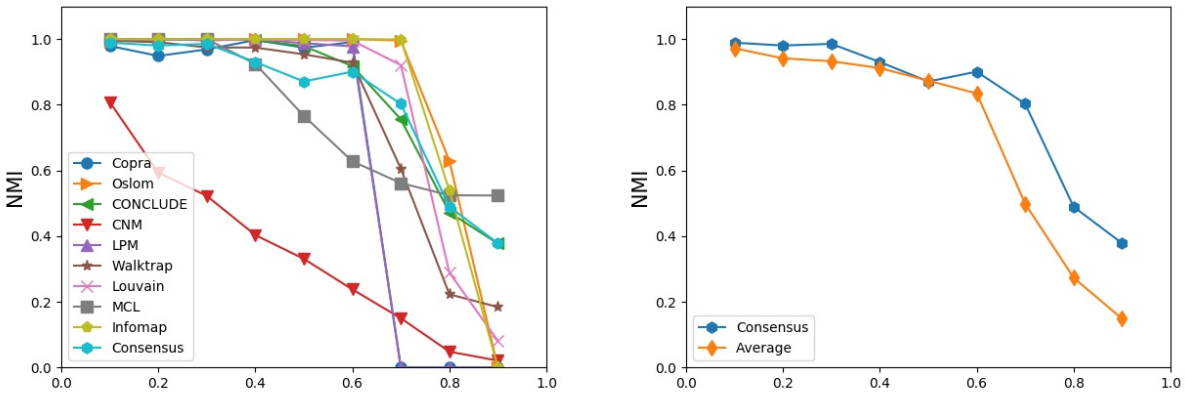
Figure 3. NMI versus mixing parameter for N = 1000.

between 20 to 100 nodes, whereas in the diagrams correspond to N = 1000 nodes, the range goes from 10 to 50 nodes (see Table 1).

As an important point, it is observable in Figures 3b, 4b and 5b that for all networks the average curve of base methods is always under *Mitra* curve. This means that for these networks, our *Mitra* is not a simple average and is better than the average, especially when μ increases and the networks become more complicated. This conclusion is considerable and shows the base methods reinforced each other by the designed consensus. In other words, *Mitra* reinforce the strengths points of the base methods. The distance between the NMI of *Mitra* and average NMI of base methods, is close to 0.2 for the most complicated networks, and the difference is even much more for $\mu = 0.7$. The results show the enhancement of quality (about 0.2) according to NMI values for the most complicated networks where μ goes from 0.6 to 0.9. Same or greater distance is seen for larger networks. In other words, *Mitra* works better for more complicated networks. This seems interestingly considerable and shows the power of *Mitra*.

It is observable in NMI diagrams (Figures3a, 4a and 5a) that Oslom, CONCLUDE and MCL show good performances. However, for greater values of μ, MCL almost assigns each node to a separate community. This assignment is not informative. Oslom shows this behavior for N = 1000 as well. Another considerable property of *Mitra* appears here; *Mitra* produces solutions with an acceptable structure and the close number of communities to the ground truth in all the networks. Moreover, the effect of low quality solutions of some base methods, is covered by *Mitra*. This shows the superiority of *Mitra* over the base methods. This is observable, for example, in Figure 3a. In this case, for N = 1000 and μ = 0.9 the number of ground truth communities is 45. The corresponding number is 130 for *Mitra*, but it is 927 for MCL and 1000 for Oslom. This conclusion is also valid for MCL in the cases of $N = 5000$ and $N = 10000$. As another example for MCL in the case of $\mu = 0.9$ and $N = 10000$, NMI value is 0.56, but the number of communities is 10000 (each node is in a separate community), whereas the community number is 1459 for *Mitra* and 190 for ground truth. CONCLUDE has a close community number to our *Mitra* method and a close value of NMI in all the networks in Figures 3-5. To avoid excess explanation and keep the consistency of this paper, the number of communities are not reported completely.

Now the properties and behavior of the base methods are investigating. To increase the readability of the paper, Figures .3a, 4a and 5a are referred as NMI diagrams in the following. Generally, it is illustrated in the NMI diagrams that all the methods start very well and the NMI values are close to 1 for the smaller values of μ, but all have a decreasing trend and approaching zero for the bigger values of μ. For example in Figure 3a becoming zero starts from μ=0.6. This behavior is predictable because for the lower values of μ the networks are simpler and



(a) NMI for the base methods and *Mitra*

(b) Comparison with average of base methods

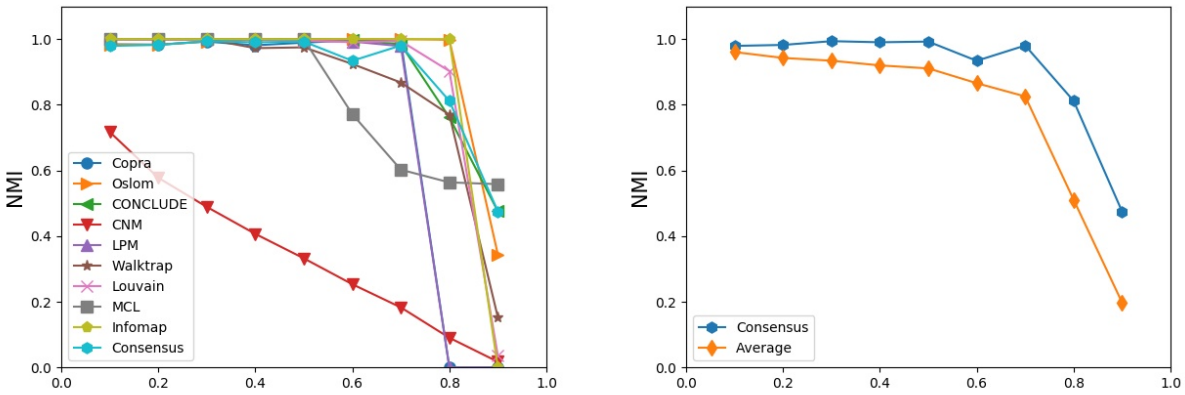
Figure 4. NMI versus mixing parameter for $N = 5000$.

almost all methods can find the correct clusters. When μ grows, the networks become more complicated and the chance of finding the correct clustering decreases.

The comparison of Figures 3-5 illustrates that almost all of the algorithms perform better on the networks of larger size. For example Oslom is not stable for N=1000 (see Figure 3a), but this method is stable and shows a good performance in Figures 4a and 5a. For this reason, Oslom is removed from the ensemble for the case of N = 1000 nodes. As another example, for N=1000 the performance of LPM and Copra heads down to zero starting from μ = 0.6 (see Figure 3a), but in the cases of N=5000 and 10000 (Figures 4a and 5a), their performances become zero starting from μ = 0.7 and μ = 0.8 respectively. As illustrated in Figure 3a, most algorithms perform well except for CNM. The diagrams also show that the performance of MCL decreases faster than the competing diagrams. Nevertheless, considering NMI results, MCL is performing better than other methods when it comes to greater values of μ (e.g. when μ = 0.9). As you see, NMI of other methods drop faster down to 0, while NMI of MCL stays above 0. We can see in the NMI diagrams that CNM always generates an approximately linear curve that is almost the worst case between the base methods. Moreover, the performances of LPM and Copra become zero before all the other base algorithms. On the other hand, some methods are able to find communities even for the most difficult cases and their associated NMI never become zero, e.g. CONCLUDE. CONCLUDE is one of the best performing methods according to NMI diagrams and it also works well based on the quality scores provided in later subsection.

It is shown in the NMI diagrams that NMI becomes zero for some methods. In these cases, the method is not able to find meaningful clusters and in some cases (e.g. Copra) groups all the nodes in one cluster. It is also interesting that although NMI of Infomap on G becomes zero in some points (i.e. this method is unable to find clusters of G), Infomap is still able to find a good partition for consensus graph G_u , when applied as the last base method on the consensus graph. It is notable that most of the observations are similar in Figures 3-5. They only differ in some details. Therefore, we do not repeat our reasoning again for the larger networks.

Finally we conclude from the comprehensive findings that it is necessary for getting a good consensus result to employ good base methods. In addition, removing the better quality base methods generates a weak consensus method. Moreover using a weaker base method as the final algorithm A will produce weak results although the initial base methods generate sufficiently good quality results.



(a) NMI for the base methods and *Mitra*

(b) Comparison with average of base methods

Figure 5. NMI versus mixing parameter for N = 10000.

Scores	Con	Exp	Int-den	Cut-r	N-cut	Mod-r	Mod	E-cut
CNM	0.28	2.58	0.31	0.11	0.38	1.39	0.38	25.33
LPM	0.13	1.19	0.5	0.07	0.23	1	0.36	20
Walktrap	0.21	1.82	0.18	0.08	0.3	1.65	0.37	18.67
Louvain	0.31	2.54	0.03	0.1	0.39	1.94	0.38	21
Copra	0.63	5.3	-0.51	0.17	0.68	3.02	0.31	17.78
MCL	0.88	5.66	-0.05	0.18	0.92	0.67	0.18	12.29
Oslom	0.13	1.19	0.5	0.07	0.23	1	0.36	20
Infomap	0.2	1.7	0.19	0.08	0.28	1.62	0.4	18.67
Mitra	0.13	1.18	0.5	0.07	0.23	1	0.37	20

Table 3. Quality scores for Zachary dataset

6.3. Quality scores results

For further investigation of the properties of methods, in this part we present the quality scores illustrations associated with all the methods. These functions model community scores. For each benchmark network the average value of each score is computed for all the communities detected by each community detection method. This computation is repeated for all 100 samples of network configurations. The mean value produced for these 1000 samples is plotted for each score. Each community score is plotted as a function of μ. Lower value of score f(S) signifies a more community-like set of nodes. The score results are reported for real and artificial datasets.

6.3.1. Real datasets results As describe later, three real datasets are selected to evaluate. Table 3, Table 4 and Table 5 show the results quality scores for the real datasets. In these tables, for the first five scores, smaller values signify a more community-like set of nodes. For modularity and modularity-ratio, greater values represent better community-like groups. The tables do not represent a super excellence for *Mitra*. Nevertheless, the tables are provided to keep consistency of the paper. The table column names are associated with Conductance, Expansion, Internal-density, cut-ratio, Normalized cut, Modularity ratio, Modularity and Edges-cut respectively pol foot

6.3.2. LFR dataset results Figures 6,7 and 8 consider the quality scores for N =1000, 5000 and 10,000 respectively (see 5.2). In order to increase the clarity of the figures, legends are not repeated in all the illustrations

Scores	Con	Exp	Int-den	Cut-r	N-cut	Mod-r	Mod	E-cut
CNM	0.25	1.53	0.56	0.02	0.28	0.88	0.5	18
LPM	0.27	1.65	0.52	0.02	0.31	0.95	0.51	25.2
Walktrap	0.18	1.2	0.64	0.01	0.21	0.72	0.53	19.5
Louvain	0.22	1.45	0.63	0.02	0.25	0.74	0.51	20.5
Copra	0.37	2.26	0.39	0.02	0.4	1.22	0.51	22.89
MCL	0.12	0.82	0.72	0.01	0.15	0.57	0.49	18.67
Oslom	0.22	1.19	0.5	0.01	0.25	1.01	0.49	15.5
Infomap	0.28	1.71	0.51	0.02	0.3	0.98	0.53	18.4
Mitra	0.33	2.43	0.63	0.03	0.38	0.73	0.29	37.5

Table 4. Quality scores for Polbooks dataset

Scores	Con	Exp	Int-den	Cut-r	N-cut	Mod-r	Mod	E-cut
CNM	0.28	3.02	0.4	0.03	0.32	1.2	0.58	45.14
LPM	0.34	3.5	0.15	0.03	0.36	1.7	0.6	31.67
Walktrap	0.31	3.33	0.17	0.03	0.34	1.65	0.6	33.45
Louvain	0.29	3.15	0.25	0.03	0.32	1.5	0.6	36.2
Copra	0.34	3.64	0.14	0.03	0.37	1.72	0.58	33.33
MCL	0.31	3.33	0.17	0.03	0.34	1.66	0.6	33.45
Oslom	0.32	3.36	0.19	0.03	0.34	1.62	0.6	33.82
Infomap	0.31	3.25	0.2	0.03	0.34	1.6	0.6	35
Mitra	0.32	3.37	0.3	0.03	0.35	1.41	0.3	39.4

Table 5. Quality scores for Football dataset

and only showed in Figures 6g, 7g and 8g (i.e. for modularity). The average value of each function over all communities is computed for each network sample. As explained before, 100 samples are generated for each network configuration. For the first five scores, smaller values of score $f(S)$, signify a more community-like set of nodes. Whereas, for modularity and modularity-ratio, greater values represent better community-like groups. Ground curve refers to ground-truth (reference solution) in the figures.

It is apparent that all these scores try to capture the same basic intuition. They reward the sets of nodes that have many internal edges and a few external, pointing to other clusters. It is apparent that the score definitions, although different in value, are highly correlated [65]. Figures. 6-8 illustrate that for simpler networks (i.e. smaller μ values), all the methods show a similar behavior. However, for the more complicated networks (i.e. bigger μ values), the methods produce diverse results and exhibit dispersion. In the following, the focus is on Figures 6-8 for N = 1000, 5000 and 10000 nodes respectively. These figures are referred as score diagrams. The diagrams illustrate cut-ratio and expansion are more correlated scoring metrics, as well as conductance and normalized-cut. This inference also confirms the conclusion in Ref. [65]. Consequently, cut-ratio and normalized-cut are not investigated in the following.

Considering the score diagrams, ground truth and *Mitra* show very similar trend in all the diagrams for different network sizes. However, MCL, Copra, Infomap and LPM show different trends. Interestingly, it means *Mitra* clusters are similar to the reference clusters. This confirms that *Mitra* shows a good performance according to quality scores.

In the score diagrams, conductance has an increasing trend. Although smaller values are preferred, the value of zero means all the nodes are placed in one class and such clustering is not informative. This is seen in LPM, Copra and Infomap for higher values of μ, and in CNM for all generated networks of any size. As another point, MCL and Copra show the worst conductance among all the base methods and their distance to other methods is considerable in all figures. This means that for MCL and Copra, the ratio between the number of edges inside

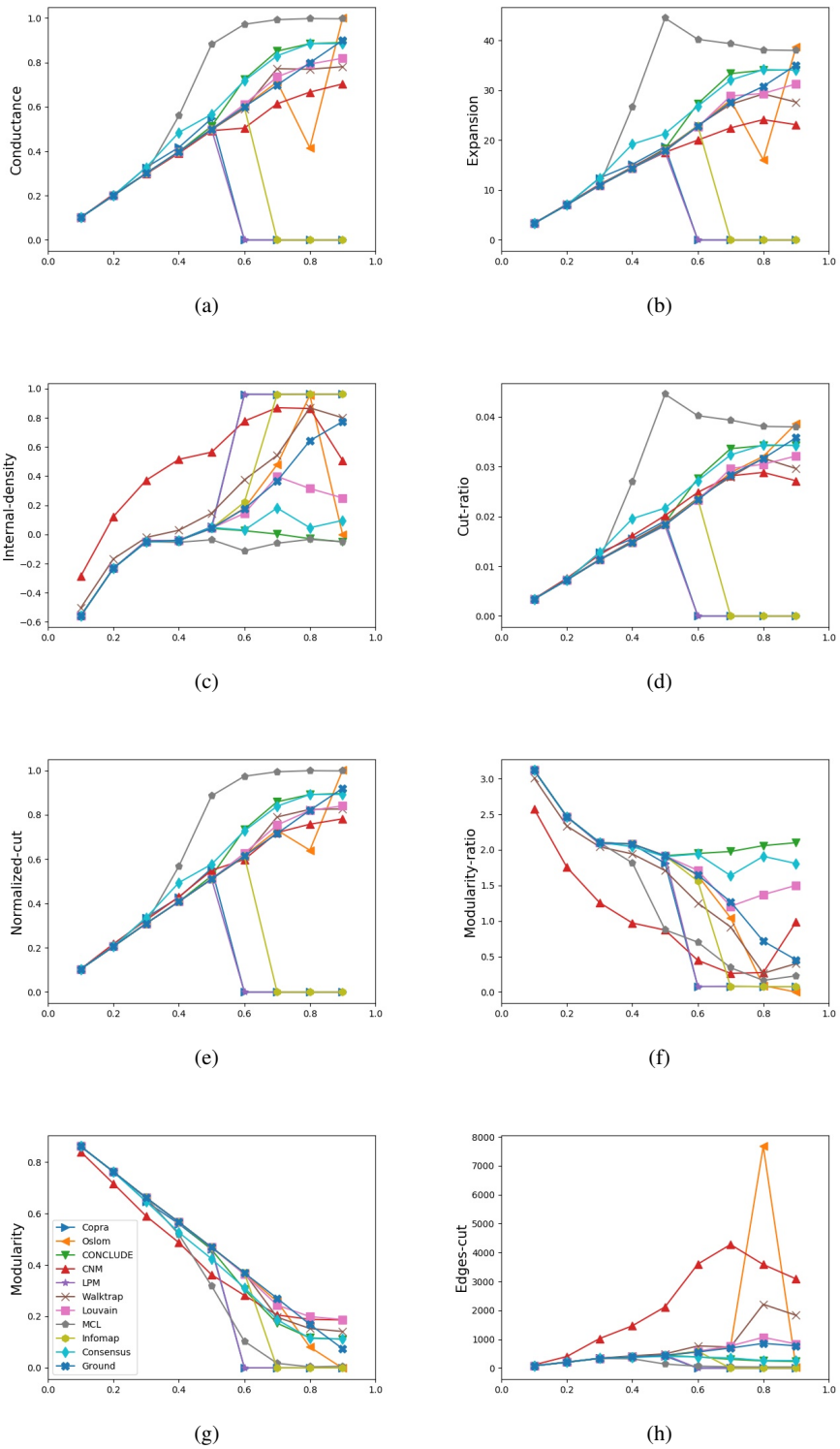
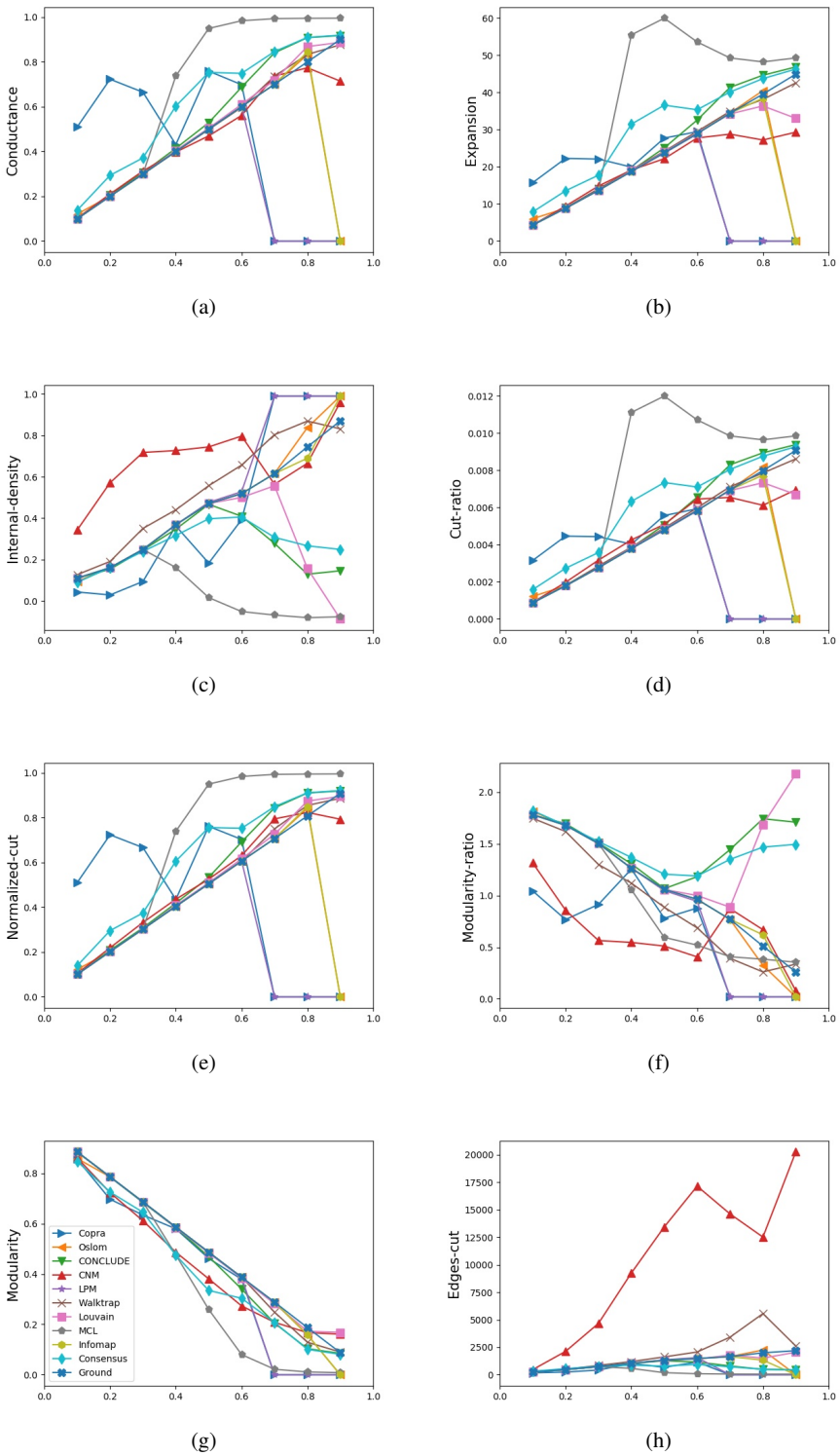
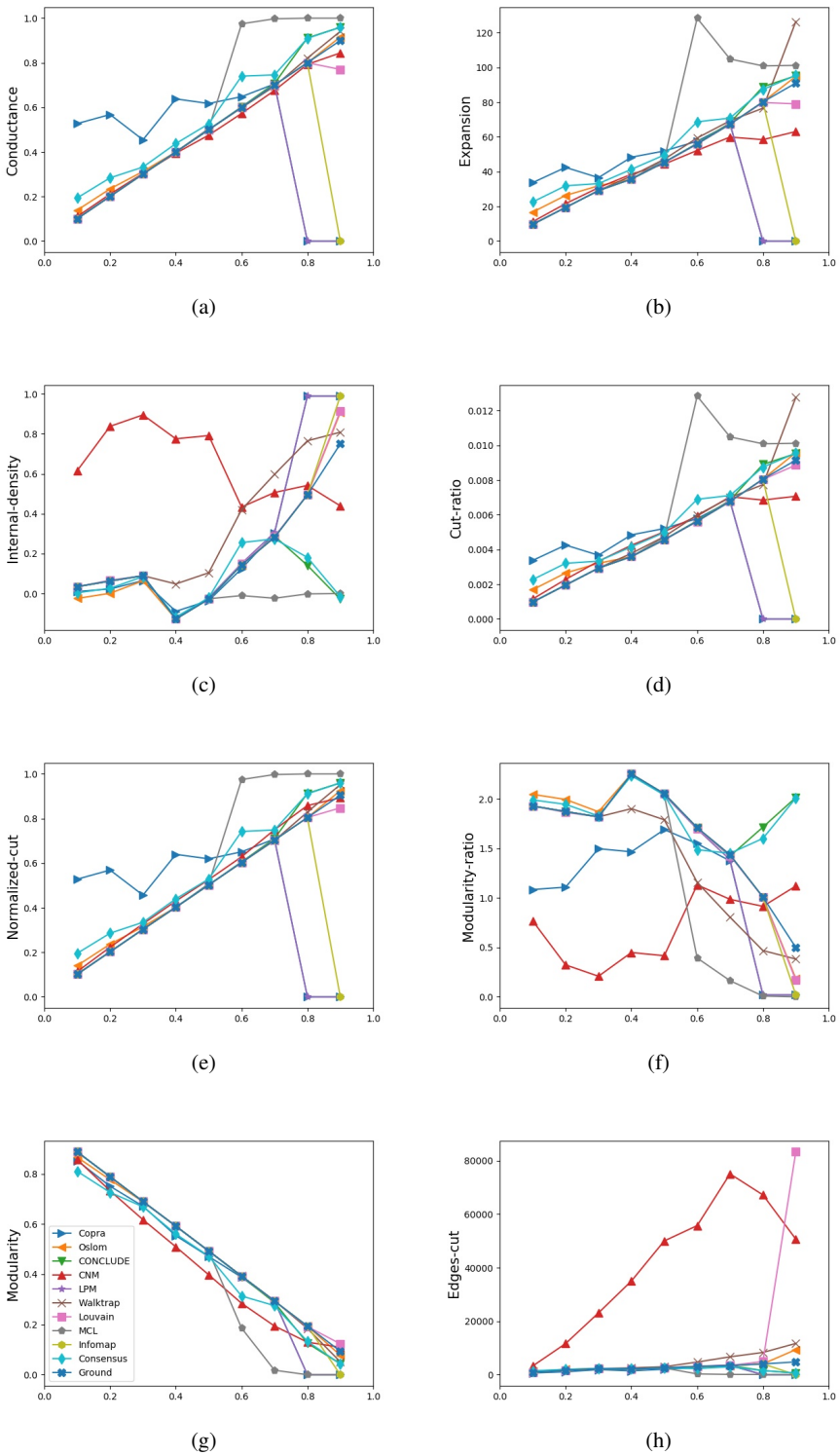


Figure 6. All the scores of subsection 5.2 for all the methods. $N = 1000$.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 7. All the scores of subsection 5.2 for all the methods. $N = 5000$.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 8. All the scores of subsection 5.2 for all the methods. $N = 10000$.

the cluster and the number of edges leaving the cluster raises quickly. In the case of expansion (Figures 6b, 7b and 8b), MCL is the worst performing method again. This means that for MCL, the normalized number of edges pointing outside of the cluster raises quickly. Although MCL is a well performing method according to NMI, deep investigation of its application and produced results makes the method unfavorable according to the number of clusters found and the quality of scores. As illustrated in Figures 6c, 7c and 8c internal-density values are diffused for different methods. Albeit like the other scores, the base methods show a similar behavior. Specially for simpler networks, CNM has a different manner and its distance to the other methods increases in all of the networks with N =1000, 5000 and 10000. It is observable that this event occurs more often in larger networks. In such networks CNM shows a lot of fluctuations in the values. Generally speaking, CNM is not preferred in any of the networks, according to NMI and internal-density. On the other hand, edges-cut of CNM is very high and extremely far from the other methods in all of the figures, which is not favorable. Another general observation is modularity and modularity-ratio tend to decrease roughly monotonically (Figures 6g, 7g, 8g,6f, 7f, 8f). This is not surprising, as the networks are becoming more complicated gradually and, the algorithms are not able to find the correct communities. Although the modularity ratio of the methods are clearly different, they show very similar behavior. This suggests that modularity ratio is not a practical measure to detect a community, and thus may not be preferred. Considering edges-cut, it is observable that some methods (e.g. Oslom and CNM in Figure 6h) produce a diverse set of numbers, from a few edges to more than 10000. For example edges-cut is 7500 for CNM and 4200 for Oslom in Figure 8h. This diversity of edges-cut values shows the structure of clusters produced by Oslom and CNM is different from other methods.
Another important result is for larger networks (e.g. $N = 100,000$), Walktrap is not able to finish the execution and halts. CONCLUDE is also generates a solution in a very long time (almost some days) for these network sizes. We can also draw conclusions about the general behavior of the base algorithms. According to NMI and quality scores, it is observable that almost all of the algorithms follow the same trend in their application for lower values of μ, although they use different definitions for communities and apply different strategies to find them.

7. Conclusion and Future work

In this work, we proposed a new ensemble clustering approach to enhance community detection by employing bipartite networks. The method is called *Mitra*. The proposed method fuses results of base community detection methods and provides a more accurate community structure for a network. We performed an empirical comparison and evaluated the quality of communities identified by a variety of community detection algorithms. Moreover, we compared the performance and quality of the methods according to Normalized Mutual Information (NMI) and extended our study by evaluating some community scores.

We can conclude about the general behavior of our *Mitra* method. The method is good performing, in the sense that its performance never becomes zero according to NMI and is far better than the average performance of the other employed base methods. This conclusion confirms our first assumption that weak points of one base method are covered by the strengths of the others by using *Mitra* approach. Therefore, *Mitra* helps in mitigating the limitations of base methods. Moreover, the experimental evaluations show *Mitra* is not a simple average of base methods and, is far better than the average, especially when networks become more complicated. This conclusion is interestingly remarkable and shows *Mitra* reinforces the strengths points of base methods. Furthermore, *Mitra* produces solutions with an acceptable structure and the close number of communities to the ground truth in all the networks. In other word, its behavior is very close, according to trend and value, to the ground-truth.

As the last point, base methods play the same role in *Mitra*. Therefore, it is appealing to change the priority of roles of base methods to find a better ensemble method. We are going to investigate this in future. Moreover, employing more evaluation criteria for quality estimation in community detection methods is our future plan for continuing this work.

Acknowledgement

We would like to thank Anis Yousefi involved in reviewing the manuscript for her suggestions that have significantly improved the technical quality of this paper. We would also like to thank Martin Rosvall, Maryam Kheirkhahzadeh and Mohammad Ghalambor Dezfuli for helpful discussions, comments, and suggestions that helped to improve the paper.

References

1. http://micans.org/mcl/
2. http://networkrepository.com/polbooks.php
3. https://cs.unm.edu/ aaron/research/fastmodularity.htm
4. https://perso.uclouvain.be/vincent.blondel/research/louvain.html
5. https://sites.google.com/site/andrealancichinetti/files
6. https://sites.google.com/site/andrealancichinetti/software
7. https://www-complexnetworks.lip6.fr/ latapy/pp/walktrap.html
8. https://www.cs.bris.ac.uk/ steve/networks/software/copra.html
9. http://www.emilio.ferrara.name/conclude/
10. http://www.mapequation.org/code.html
11. http://www.oslom.org/software.htm
12. Abbe, E.: Community detection and stochastic block models: recent developments. The Journal of Machine Learning Research **18**(1), 6446–6531 (2017)
13. E. Macedo , Two-Step Semidefinite Programming approach to clustering and dimensionality reduction. Statistics, Optimization & Information Computing, 3 (3), 294–311 (2015)
14. R. Alguliyev,R. Aliguliyev, L Sukhostat, Anomaly detection in Big data based on clustering. Statistics, Optimization & Information Computing, 5 (4), 325–340, (2017)
15. Almeida, H., Guedes, D., Meira Jr, W., Zaki, M.J.: Is there a best quality metric for graph clusters? In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 44–59. Springer (2011)
16. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10,008 (2008)
17. Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., Wagner, D.: Maximizing modularity is hard. arXiv preprint physics/0608255 (2006)
18. Chakraborty, T., Dalmia, A., Mukherjee, A., Ganguly, N.: Metrics for community analysis: A survey. ACM Computing Surveys (CSUR) **50**(4), 54 (2017)
19. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. Physical review E **70**(6), 066,111 (2004)
20. Dahlin, J., Svenson, P.: Ensemble approaches for improving community detection methods. arXiv preprint arXiv:1309.0242 (2013)
21. Danon, L., Diaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment **2005**(09), P09,008 (2005)
22. De Meo, P., Ferrara, E., Fiumara, G., Provetti, A.: Mixing local and global information for community detection in large networks. Journal of Computer and System Sciences **80**(1), 72–87 (2014)
23. Dongen, S.M.: Graph clustering by flow simulation (2000)
24. Duan, L., Binbasioglu, M.: An ensemble framework for community detection. Journal of Industrial Information Integration **5**, 1–5 (2017)
25. Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. Bioinformatics **19**(9), 1090–1099 (2003)
26. Emmons, S., Kobourov, S., Gallant, M., Börner, K.: Analysis of network clustering algorithms and cluster quality metrics at scale. PloS one **11**(7), e0159,161 (2016)
27. Fern, X.Z., Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the twenty-first international conference on Machine learning, p. 36. ACM (2004)
28. Fortunato, S.: Community detection in graphs. Physics reports **486**(3), 75–174 (2010)
29. Fortunato, S., Barthelemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Sciences **104**(1), 36–41 (2007)
30. Fortunato, S., Castellano, C.: Community structure in graphs. In: Computational Complexity, pp. 490–512. Springer (2012)
31. Fortunato, S., Hric, D.: Community detection in networks: A user guide. Physics Reports **659**, 1–44 (2016)
32. Fortunato, S., Latora, V., Marchiori, M.: Method to find community structures based on information centrality. Physical review E **70**(5), 056,104 (2004)
33. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
34. Good, B.H., de Montjoye, Y.A., Clauset, A.: Performance of modularity maximization in practical contexts. Physical Review E **81**(4), 046,106 (2010)
35. Gregory, S.: Finding overlapping communities in networks by label propagation. New Journal of Physics **12**(10), 103,018 (2010)
36. Harenberg, S., Bello, G., Gjeltema, L., Ranshous, S., Harlalka, J., Seay, R., Padmanabhan, K., Samatova, N.: Community detection in large-scale networks: a survey and empirical evaluation. Wiley Interdisciplinary Reviews: Computational Statistics **6**(6), 426–439

(2014)
37. Jeub, L.G., Sporns, O., Fortunato, S.: Multiresolution consensus clustering in networks. Scientific reports **8**(1), 3259 (2018)
38. Kanawati, R.: Community detection in social networks: the power of ensemble methods. In: Data Science and Advanced Analytics (DSAA), 2014 International Conference on, pp. 46–52. IEEE (2014)
39. Kanawati, R.: Ensemble selection for community detection in complex networks. In: International Conference on Social Computing and Social Media, pp. 138–147. Springer (2015)
40. Kannan, R., Vempala, S., Vetta, A.: On clusterings: Good, bad and spectral. Journal of the ACM (JACM) **51**(3), 497–515 (2004)
41. Khatoon, M., Banu, W.A.: A survey on community detection methods in social networks. International Journal of Education and Management Engineering (IJEME) **5**(1), 8 (2015)
42. Kheirkhahzadeh, M., Lancichinetti, A., Rosvall, M.: Efficient community detection of network flows for varying markov times and bipartite networks. Physical Review E **93**(3), 032,309 (2016)
43. Lancichinetti, A., Fortunato, S.: Consensus clustering in complex networks. Scientific reports **2** (2012)
44. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical review E **78**(4), 046,110 (2008)
45. Lancichinetti, A., Radicchi, F., Ramasco, J.J., Fortunato, S.: Finding statistically significant communities in networks. PloS one **6**(4), e18,961 (2011)
46. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th international conference on World wide web, pp. 631–640. ACM (2010)
47. MacKay, D.J.: Information theory, inference and learning algorithms. Cambridge university press (2003)
48. Meilă, M.: Comparing clusteringsan information based distance. Journal of multivariate analysis **98**(5), 873–895 (2007)
49. Newman, M.: Networks: an introduction. Oxford university press (2010)
50. Newman, M.E.: Fast algorithm for detecting community structure in networks. Physical review E **69**(6), 066,133 (2004)
51. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E **69**(2), 026,113 (2004)
52. Ovelgönne, M., Geyer-Schulz, A.: An ensemble learning strategy for graph clustering. Graph Partitioning and Graph Clustering **588**, 187 (2012)
53. Pizzuti, C., Socievole, A.: A genetic algorithm for community detection in attributed graphs. In: International Conference on the Applications of Evolutionary Computation, pp. 159–170. Springer (2018)
54. Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. Graph Algorithms Appl. **10**(2), 191–218 (2006)
55. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the National Academy of Sciences of the United States of America **101**(9), 2658–2663 (2004)
56. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Physical review E **76**(3), 036,106 (2007)
57. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences **105**(4), 1118–1123 (2008)
58. Schaub, M.T., Delvenne, J.C., Rosvall, M., Lambiotte, R.: The many facets of community detection in complex networks. Applied network science **2**(1), 4 (2017)
59. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence **22**(8), 888–905 (2000)
60. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of machine learning research **3**(Dec), 583–617 (2002)
61. Tagarelli, A., Amelio, A., Gullo, F.: Ensemble-based community detection in multilayer networks. Data Mining and Knowledge Discovery **31**(5), 1506–1543 (2017)
62. Touretzky, D.S.: Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference, vol. 8. Mit Press (1996)
63. Waltman, L., van Eck, N.J.: A smart local moving algorithm for large-scale modularity-based community detection. The European Physical Journal B **86**(11), 1–14 (2013)
64. Xie, J., Kelley, S., Szymanski, B.K.: Overlapping community detection in networks: The state-of-the-art and comparative study. ACM Computing Surveys (csur) **45**(4), 43 (2013)
65. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems **42**(1), 181–213 (2015)
66. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of anthropological research **33**(4), 452–473 (1977)