Ensemble-Based Discovery of Disjoint, Overlapping and Fuzzy Community Structures in Networks

Tanmoy Chakraborty, Noseong Park

Indraprastha Institute of Information Technology Delhi (IIIT-D), India
University of North Carolina, Charlotte, USA

Abstract

Though much work has been done on ensemble clustering in data mining, the application of ensemble methods to community detection in networks is in its infancy. In this paper, we propose two ensemble methods: EnDisCo and MeDOC++. EnDisCo performs disjoint community detection. In contrast, MeDOC++ performs disjoint, overlapping, and fuzzy community detection and represents the first ever ensemble method for fuzzy and overlapping community detection. We run extensive experiments with both algorithms against both synthetic and several real-world datasets for which community structures are known. We show that EnDisCo and MeDOC++ both beat the best-known existing standalone community detection algorithms (though we emphasize that they leverage them). In the case of disjoint community detection, we show that both EnDisCo and MeDOC++ beat an existing ensemble community detection algorithm both in terms of multiple accuracy measures and run-time. We further show that our ensemble algorithms can help explore core-periphery structure of network communities, identify stable communities in dynamic networks and help solve the “degeneracy of solutions” problem, generating robust results.

Keywords: Ensemble approach, Community detection, Core-periphery organization, Stable communities

1. Introduction

Though most human beings recognize a community of people when they see one, coming up with a formal mathematical definition of a community has proved challenging, leading to a plethora of diverse technical definitions
which capture the intuitive human understanding of a community with different degrees of accuracy [11]. Because of this, different definitions of a community use different objective functions (such as modularity [2], significance [3], permanence [4, 5, 6]), whose optimization leads to the detection of the underlying community structure.

We ask ourselves the question: can we come up with a computational model of known communities in a network that accurately approximates real-world communities in the network by leveraging the best known existing network community detection algorithms and their associated objective functions? We are not the first to examine this question - ensemble methods have already been pioneered in network community detection by [7, 8], building on past work on clustering (in non-network settings) in data mining [9].

Apart from these factors, many other factors suggest that an ensemble-based approach may lead to significantly improved accuracy:

- **Dependence on vertex ordering:** Existing community finding algorithms are highly dependent on vertex-ordering [10, 11]. If we let an algorithm start from different seed vertices in different iterations for a particular network, it might produce completely different community structures. Each structure can be viewed as a different view of what communities might exist.

- **Degeneracy of solution:** Existing community finding algorithms suffer from the “degeneracy of solutions” problem because they admit an exponential number of distinct high-scoring solutions and typically lack a clear global maximum. Each such solution represents a possible view of the underlying community structure and there is no reason to prefer one over another.

- **Lack of ground-truth communities:** Most real-world networks do not have (partial) ground-truth community structure to validate predicted communities. Therefore, it is extremely challenging to do any cross-validation to tune the input parameters of an algorithm.

In this paper, we extend our previous study [12] where we suggested how to combine multiple solutions to generate ensemble community detection algorithms. However, it was unclear how one can select base solutions. Moreover, there was no single algorithm which can be able to detect three types of communities – disjoint, overlapping and fuzzy. In this paper, we
present a comprehensive experiment to show the superiority of two of our proposed algorithms – EnDisCo, and MeDOC++, along with our previous findings [12]. In particular, the summary of the contributions presented in this paper is as follows:

1. We propose an ensemble-based algorithm called EnDisCo that identifies disjoint communities. EnDisCo is built on the idea that the larger the number of algorithms that place two vertices in the same community, the more likely it is that they really do belong to the same community. We represent each vertex in a feature space and capture the pair-wise distances between vertices. This in turn generates a latent network, capturing the hidden similarities among vertices. A re-clustering algorithm is then used to cluster the vertices in the latent network to produce the final community structure (Section 3).

2. We propose MeDOC++, a meta-clustering based algorithm that to the best of our knowledge is the first ensemble-based generalized community detection algorithm to detect disjoint, overlapping and fuzzy communities in a network. The idea behind this algorithm is to generate meta communities from the “base communities” (i.e. communities generated by existing community detection algorithms) by grouping redundant solutions together. We propose a vertex-to-community association function that provides an accurate estimate of the community membership of different vertices in a network. This association function further allows us to detect overlapping and fuzzy community structures from the network (Section 4).

3. We conduct an experimental analysis using both synthetic and real-world networks whose ground-truth community structure is known. Experimental results are shown separately for disjoint (Section 5), overlapping (Section 6) and fuzzy (Section 7) community detection. We show that our ensemble-based algorithms (particularly MeDOC++) outperform all the state-of-the-art baseline algorithms with significant margin including the best known and best performing “consensus clustering” algorithm for disjoint community detection [8]. We also present a detailed explanation of how to choose the parameters of the ensemble algorithms.

4. We provide four strategies to choose a subset of “base” community detection algorithms in order to obtain highly accurate communities. These strategies are based on two fundamental quantities – *quality* and
diversity. We show that a trade-off of these two quantities is required to select the best subset of base solutions (Section 8).

5. MeDOC++ further allows us to explore the core-periphery structure of communities in a network. We observe that vertices with more association within a community form the core unit of the community, whereas peripheral vertices are loosely associated with the community. Furthermore, we show that one can use MeDOC++ to detect communities in a dynamic time-varying network that are stable, i.e., remain almost invariant over time (Section 9).

6. Finally we show that both EnDisCo and MeDOC++ significantly reduce the problem of “degeneracy of solutions” in community detection (Section 10) and are much faster than consensus clustering [8], a recently proposed ensemble-based disjoint community finding algorithm (Section 11).

In this paper, the major additions of new contributions with our existing work [12] are as follows: (i) We present MeDOC++, the first ensemble based algorithm that can detect disjoint, overlapping and fuzzy community structures (Section 4). (ii) We present results of detecting fuzzy community structures from synthetic and real-world networks (Section 7). To the best of our knowledge, this is the first time where the fuzzy community detection algorithm is verified against real-world ground-truth. (iii) We present four strategies to select a subset of base solutions which produce near-optimal results (Section 8). (iv) We present two new implications of MeDOC++ algorithm – (a) it can explore the core-periphery structure of communities in a network, (b) it can detect stable communities in dynamic networks (Section 9). (v) We present detailed analysis to show how our proposed algorithms reduce the effect of “degeneracy of solutions” compared to other baseline algorithms (Section 10).

Throughout the paper, we use the term “community structure” to indicate the result returned by a community detection algorithm. A community structure therefore consists of a set of “communities” and each community consists of a set of vertices.

2. Related Work

In this section, we present the literature related to the community detection in three subparts. First, we describe past efforts on disjoint community
detection. Second, we discuss fuzzy and overlapping community detection. Finally, we present related work on ensemble-based community detection. Due to the abundance of literature on community detection, we restrict our discussion to best known and/or most recent works. Extensive survey articles on community detection are available in [1], [13] and [14].

2.1. Disjoint Community Detection

Past efforts devoted to community detection mostly assume that nodes are densely connected within a community and sparsely connected across communities. Such efforts include modularity optimization [15, 16, 17, 18, 2], spectral graph-partitioning [19, 20], clique percolation [21, 22], local expansion [23, 24], random-walk based approaches [25, 26], information theoretic approaches [27, 28], diffusion-based approaches [29], significance-based approaches [30] and label propagation [29, 31, 32]. Most of these efforts detect communities in static networks. On the other hand, a large number of algorithms were proposed to detect communities in dynamically evolving networks (i.e., Internet, Online Social Networks), such as LabelRankT [33], Estrangement [34] and intrinsically dynamic community detection algorithm [35]. Several pre-processing techniques [36, 37] have been developed to improve the quality of the solutions generated by these algorithms. These methods generate preliminary community structure on a set of selected vertices and then modify the structure over successive steps to cover all the vertices. Recently, [11] showed the effect of vertex ordering on the performance of the community detection algorithms.

2.2. Fuzzy and Overlapping Community Detection

Another set of community detection algorithms allow a vertex to be a part of multiple communities. “CFinder” [22] was the first method of this kind which was based on clique-percolation technique. However, since more real-world networks are sparse in nature, CFinder generally produces low quality output [38]. The idea of partitioning links instead of vertices to discover community structure has also been explored [39]. Some algorithms are based on local expansion and optimization such as LFM [24], OSLOM [30], EAGLE [40], MOSES [41] and GCE [42]. [43, 44] proposed fuzzy community detection technique. BIGCLAM [45] uses Nonnegative Matrix Factorization (NMF) framework for fuzzy/overlapping community detection. Zhang et al. used NMF to detect overlapping communities given the feature vector of vertices and known number of communities [46].
There is another type of algorithms which exploit local expansion and optimization to detect overlapping communities. For instance, “RankRemoval” algorithm [47] uses a local density function. LFM [48] and MONC [49] maximize a fitness function over successive iterations. OSLOM [30] measures the statistical significance of a cluster w.r.t a global null model during community expansion. [50] proposed a combination of “belongingness” and “modified modularity” for local expansion. EAGLE [40] and GCE [42] use an agglomerative framework to detection overlapping communities. COCD [51] first identifies cores after which the remaining vertices are attached to the cores with which they have the largest connections.

[52] modeled overlapping community detection as a nonlinear constrained optimization problem and solved by simulated annealing methods. [53], [54], [55], [56] used mixer model to solve this problem. [57] used affinity propagation clustering methods for overlapping community detection. [58] proposed a seed set expansion approach for community detection.

The label propagation algorithm has been extended to overlapping community detection. COPRA [59] updates “belonging coefficient” of a vertex by averaging the coefficient from all its neighbors at each time step. SLPA [32], [31] propagates labels across vertices based on the pairwise interaction rules. [60] proposed a game-theoretic framework in which a community is associated with a Nash local equilibrium.

Beside these, CONGA [61] uses GN algorithm [62] to split a vertex into multiple copies. [63] proposed an iterative process that reinforces the network topology and proximity that is interpreted as the probability of a pair of vertices belonging to the same community. [64] proposed an approach focusing on centrality-based influence functions. [65] proposed fuzzy clustering based disjoint community detection technique.

2.3. Community Detection using Ensemble Approach

There has been a plethora of research in traditional data mining (not involving networks) to cluster data points using an ensemble approach (see [9] for a detailed review). These approaches can be classified into two categories [9]: object co-occurrence based approaches and median partitioning based approaches. However, when it comes to the case of clustering vertices in networks, there are very few such attempts. [7] proposed an instance-based ensemble clustering method for network data by fusing different community structures. [29] addressed the advantages of combining multiple community
structures. CGGC [66] presents a modularity maximization based ensemble technique. YASCA is another ensemble approach that can detect ego-centered communities [67, 68], and identified the importance of the quality and diversity of base outputs [69].

Recently, [8] proposed “consensus clustering” which leverages a consensus matrix for disjoint graph clustering. It detects consensus clustering by reweighting the edges based on how many times the pair of vertices are allocated to the same community by different identification methods. This has been proved to be a stable method, outperforming previous approaches for disjoint community detection. However, we differ from them w.r.t. four different points as follows:

1. Past work measures the number of times two vertices are assigned to the same community, thus ignoring the global similarity of vertices; whereas we aim at capturing the global aspect by representing the network into a feature space or grouping the redundant base communities into a meta community.
2. They either take multiple algorithms or run a particular algorithm multiple times for generating inputs to an ensemble algorithm, whereas we consider both of them.
3. They run algorithms multiple times to generate consensus matrix in each iteration, and keep on repeating the same steps until the matrix converges to a block diagonal matrix which leads to a huge computational cost; whereas we run base algorithms multiple time only in the first step and leverage the base solutions in later steps which decreases the overall runtime. This leads to significant performance gains.
4. We are the first to show how aggregating multiple disjoint base communities can lead to discover disjoint, overlapping and fuzzy community structures simultaneously.

However, we consider consensus clustering as one of the state-of-the-art techniques for disjoint community detection and compare it with our algorithms. For overlapping and fuzzy community detection, we present the first ever “ensemble based” algorithm in the literature.

3. EnDisCo: Ensemble-based Disjoint Community Detection

In this section, we present EnDisCo (Ensemble-based Disjoint Community Detection), an ensemble based algorithm to generate disjoint communities in
**Algorithm 1: EnDisCo: Ensemble-based Disjoint Community Detection**

**Data:** Graph $G(V, E)$;  
Base algorithms $\mathcal{AL} = \{Al_m\}_{m=1}^M$;  
$K$: Number of iterations;  
$\text{INV}(., .)$: Involvement function;  
$\text{SIM}(., .)$: Similarity function between two vectors;  
$R\text{Algo}$: Algorithm for re-clustering  

**Result:** Disjoint community structure $\mathbb{D}C$ 

1. $\Gamma = \phi$  
   // Set of all base community structures  

   // Generating base partitions

2. for each algorithm $Al_m \in \mathcal{AL}$ do  
3. Run $Al_m$ on $G$ for $K$ different vertex orderings and obtain $K$ community structures, denoted by the set $\Gamma_m$; each community structure $C_k \in \Gamma_m$ is of different size and indicated by $C_k = \{C_{1k}, \ldots, C_{ak}\}$;  
4. $\Gamma = \Gamma \cup \Gamma_m$;  

   // Constructing ensemble matrix

5. for each $v \in V$ do  
6. $F(v) = \phi$;  
   // Feature vector of $v$  
7. $D_v = 0$;  
   // Max distance of $v$ to any community  
8. $Clu = 0$;  
   // Total no of communities  
9. for each $\Gamma_m \in \Gamma$ do  
10. for each $C_k \in \Gamma_m$ do  
11. for each $C \in C_k$ do  
12. Compute $d^C_v = 1 - \text{INV}(v, C)$;  
13. $F(v) = F(v) \cup d^C_v$;  
14. if $d^C_v \geq D_v$ then  
15. $D_v = d^C_v$;  
16. $Clu = Clu + 1$;  
17. $P(v) = \phi$;  
18. for each $F_i(v) \in F(v)$ do  
19. // Posterior probability of $v$ in $C_i$  
20. Compute $P(C_i|v) = \frac{D_v - F_i(v) + 1}{Clu - D_v + Clu - \sum_{k=1}^{Clu} F_k(v)}$;  
21. $P(v) = P(v) \cup P(C_i|v)$;  
22. Build an ensemble matrix $\mathbb{M}_{|V| \times |V|}$, where  
   $\forall u, v \in V; \mathbb{M}(u, v) = \text{SIM}(P(u), P(v))$;  
23. return $\mathbb{D}C$
Table 1: Few important notations used in this paper.

| Notation | Description |
|----------|-------------|
| $G(V, E)$ | An undirected network with sets of vertices $V$ and edges $E$ |
| $\mathcal{AL}$ | $\{A_{m=1}^M\}$, set of $M$ base disjoint community detection algorithms |
| $K$ | Number of iterations (number of vertex orderings) |
| $C_m^k$ | $\{C_{m}^{1k} \ldots C_{m}^{MK}\}$, base community structure discovered by a base disjoint algorithm on $k^{th}$ vertex ordering |
| $\Gamma_m$ | $\{C_{m}^{1k} \ldots C_{m}^{MK}\}$, set of base disjoint community structures discovered by base algorithm on $K$ different vertex orderings |
| $\Gamma$ | $\{\Gamma_{m}^{1K} \ldots \Gamma_{m}^{MK}\}$, set of all $MK$ base disjoint community structures |
| $F(v)$ | Feature vector of a vertex $v$ |
| $D_v$ | Maximum distance of $v$ to any community |
| $Clu$ | $\bar{a}MK$, approximate total number of communities in $\Gamma$ ($\bar{a}$ is the average size of a base community structure) |
| $\text{INV}(v,c)$ | Involvement function of $v$ in community $C$ |
| $P(C_i|v)$ | Posterior probability of $v$ being part of a community $C_i$ |
| $P(v)$ | Probability distribution of $v$ being part of different communities |
| $\text{SIM}(u,v)$ | Similarity function between two vertices |
| $M(C_i,C_j)$ | Ensemble matrix, where $M(u,v)$ indicates the similarity of vertices $u$ and $v$ |
| $W(C_i,C_j)$ | Matching function between two communities |
| $\mathcal{F}(v,C)$ | A function measuring the association between vertex $v$ and community $C$ |
| $\mathcal{C}_{GP}$ | $\{C_{GP}^{l} \}_{l=1}^L$, meta-communities obtained from P-partite graph $GP$ |
| $\hat{A}$ | Association matrix, where $\hat{A}(v,l)$ indicates the association of $v$ in meta-community $l$ |
| $\hat{A}$ | Normalized association matrix, where $\hat{A}(v,l) = \frac{A(v,l)}{\sum_{l'=1}^L A(v,l')}$ |
| $\tau$ | A thresholds needed to detect the overlapping community structure in MeDOC++ |
| OC | Final disjoint community structure |
| OC | Final overlapping community structure |
| FC | Final fuzzy community structure |

networks. **EnDisCo** generates a *strong* start by producing different community structures using an ensemble of base community detection algorithms. Then an involvement function is used to measure the extent to which a vertex is involved with a specific community detected by a base algorithm. This in turn sets the posterior probability that a vertex belongs to any one of many different communities. This step transforms a network into a feature space. Following this, an ensemble matrix that measures the pair-wise similarity of vertices in the feature space is constructed, and this serves as a latent adjacency matrix in the next step. Finally, we apply a re-clustering algorithm on the ensemble matrix and discover the final disjoint community structure.

### 3.1. Algorithmic Description

**EnDisCo** follows three fundamental steps (a pseudo-code is shown in Algorithm 1, a toy example of the work-flow is presented in Figure 1, and important notations are shown in Table 1):
Generating base partitions. Given a network $G = (V, E)$ and a set $\mathcal{AL} = \{A_l\}_{l=1}^M$ of $M$ different base community detection algorithms, EnDisCo runs each algorithm $A_l$ on $K$ different vertex orderings (randomly selected) of $G$. This generates a set of $K$ different community structures denoted $\Gamma = \{\mathcal{C}_m\}_{m=1}^M$, where each community structure $\mathcal{C}_m = \{C_{1m}, \ldots, C_{mk}\}$ constitutes a specific partitioning of vertices in $G$, and each $C_{km}$ might be of different size (Step 3).

Constructing ensemble matrix. Given a $\Gamma$, we then compute the extent of $v$’s involvement in each community $C$ in $\mathcal{C}_m$ via an “involvement” function $\operatorname{INV}(v, C)$ (Step 12). Possible definitions of $\operatorname{INV}$ are given in Section 3.2. For each vertex $v$, we construct a feature vector $F(v)$ whose elements indicate the distance of $v$ (measured by $1 - \operatorname{INV}$) from each community obtained from different runs of the base algorithms (Step 13). The size of $F(v)$ is the number of communities $Clu$ in $\Gamma$ (approx. $\bar{a}MK$, where $\bar{a}$ is the average size of a base community structure). Let $D_v$ be the largest distance of $v$ from any community in the sets in $\Gamma$ (i.e., $D_v = \max_i F_i(v)$ in Step 15). We define the conditional probability of $v$ belonging to community $C_i$ (Step 19) as:

$$P(C_i|v) = \frac{D_v - F_i(v) + 1}{Clu \cdot D_v + Clu - \sum_{k=1}^{Clu} F_k(v)}$$  (1)

The numerator ensures that the greater the distance $F_i(v)$ of $v$ from community $C_i$, the less likely $v$ is to be in community $C_i$. The normalization factor in the denominator ensures that $\sum_{k=1}^{Clu} P(C_i|v) = 1$. We further observe that $\forall v \in V$ and $\forall i$, $P(C_i|v) > 0$. Add-one smoothing in the numerator allows a non-zero probability to be assigned to all $C_i$s, especially for $C_{\hat{k}}$ such that $\hat{k} = \arg\max_k F_k(v)$. In this case $P(C_{\hat{k}}|v)$ is assigned its maximum value $P(C_{\hat{k}}|v) = \frac{1}{Clu \cdot D_v + Clu - \sum_{k=1}^{Clu} F_k(v)}$. The larger the deviation of $F_k(v)$ from $D_v$, the more the increase of $P(C_i|v)$, the corresponding community $C_i$ becomes more likely for $v$.

The set of posterior probabilities of $v$ is: $P(v) = \{P(C_k|v)\}_{k=1}^{Clu}$ (Step 20), which in turn transforms a vertex into a point in a multi-dimensional feature space. Finally, we construct an ensemble matrix $M$ whose entry $M(u, v)$ is the similarity (obtained from a function $\operatorname{SIM}$ whose possible definitions are given in Section 3.2) between the feature vectors of $u$ and $v$ (Step 21). The ensemble matrix ensures that the more communities a pair of vertices share
Figure 1: A toy example depicting the work-flow of EnDisCo algorithm. The broken lines indicate the community boundaries. Assume that the base algorithms produce three different community structures in Step 2, and we use Restricted Closeness Centrality (RCC) and Cosine Similarity (COS) as involvement and similarity functions respectively. The definitions of COS and RCC are described in Section 3.2.

The more likely they are connected in the network [13].

(iii) Discovering final community structure. In Step 22 we use a community detection algorithm RAlgo to re-cluster the vertices from M and discover the final disjoint community structure (Step 22).

3.2. Parameter Selection

We now describe different parameters of EnDisCo:

• Involvement Function (INV): We use two functions to measure the involvement of a vertex $v$ in a community $C$: (i) Restricted Closeness Centrality (RCC): This is the inverse of the average shortest-path distance from the vertex $v$ to the vertices in community $C$, i.e., $RCC(v, C) = \frac{|C|}{\sum_{u \in C} dist(v, u)}$; (ii) Inverse Distance from Centroid (IDC): we first identify the vertex with highest closeness centrality (w.r.t. the induced subgraph of $C$) in community $C$, mark it as the centroid of $C$ (denoted by $u_c$), and then measure the involvement of $v$ as the inverse of the shortest-path distance between $v$ and $u_c$, i.e., $IDC(v, C) = \frac{1}{dist(v, u_c)}$. 


Example 3.1. In Figure 1, the RCC of vertex D for community $C_{12}$ is measured as follows. The shortest-path distances of D from E, F and G are 1, 1 and 2 respectively. Then $RCC(D, C_{12}) = \frac{3}{1+1+2} = \frac{3}{4}$. On the other hand, the centroid of community $C_{12}$ is vertex G and the distance between D and G is 2. Therefore, $IDC(D, C_{12}) = \frac{1}{2}$.

- **Similarity Function (SIM):** We consider cosine similarity ($COS$) and Chebyshev similarity ($CHE$) (Chebyshev similarity is defined as $(1 - CHE_d)$ where $CHE_d$ is the Chebyshev distance) to measure the similarity between two vectors.

- **Algorithm for Re-clustering (Ralgo):** we consider each base community detection algorithm as a candidate to re-cluster vertices from the ensemble matrix. The idea is to show that existing community detection algorithms can perform even better when they consider the ensemble matrix of network $G$ as opposed to the adjacency matrix of $G$. However, one can use any community detection algorithm in this step to detect the community structure. We will show the effect of different algorithms used in this step in Section 5.4.2.

- **Number of Iterations (K):** Instead of fixing a hard value, we set $K$ to be dependent on the number of vertices $|V|$ in the network. We vary $K$ from 0.01 to 0.50 (with step 0.05) of $|V|$ and confirm that for most of the networks, the accuracy of the algorithm converges at $K = 0.2|V|$ (Figures 3(c) and 3(f)), and therefore we set $K = 0.2|V|$ in our experiments.

3.3. Complexity Analysis

Suppose $N = |V|$ is the number of vertices in the network, $M$ is the number of base algorithms and $K$ is the number of vertex orderings. Further suppose $\bar{a}$ is the average size of the community structure. Then the loop in Step 5 of Algorithm 1 would iterate $\bar{a}NMK$ times (where $M, K \ll N$). The construction of the ensemble matrix in Step 21 would take $O(N^2)$. Graph partitioning is NP-hard even to find a solution with guaranteed approximation bounds — however, heuristics such as the famous Kernighan-Lin algorithm take $O(N^2 \cdot \log(N))$ time.
Algorithm 2: MeDOC++: A Meta Clustering based Disjoint, Overlapping and Fuzzy Community Detection

Data: Graph $G(V, E)$; 
Base algorithms $\mathcal{AL} = \{Al_m\}_{m=1}^M$; 
$K$: Number of iterations; 
$W(., .)$: Matching between pair-wise communities; 
$RAlgo$: Algorithm for re-clustering; 
$F(., .)$: vertex-to-community association; 
$\tau$: threshold for overlapping community detection

Result: Disjoint ($\mathbb{D}C$), overlapping ($\mathbb{O}C$) and fuzzy ($\mathbb{F}C$) community structures

// Constructing multipartite network
1 for $Al_m$ in $\mathcal{AL}$ do
2 \hspace{1em} Run $Al_m$ on $G$ for $K$ different vertex orderings and obtain $K$
3 \hspace{1em} community structures, denoted by the set $\Gamma_m = \{C^k_m\}_{k=1}^K$; each
4 \hspace{1em} community structure $C^k_m \in \Gamma_m$ may be of different size and is
5 \hspace{1em} denoted by $\mathcal{C}^k_m = \{C^i_{mk}\}_{i=1}^q$;
6 Construct a $P$-partite graph $GP$ (where $P = M.K$) consisting of $M.K$
7 \hspace{1em} partitions, each corresponding to each community structure obtained
8 \hspace{1em} in Step 2: vertices in partition $m^k$ are communities in $C^k_m$ and edges
9 \hspace{1em} are drawn between two pair-wise vertices (communities) $C^i_{mk}$ and $C^j_{nk}$
10 \hspace{1em} with the edge weight $W(C^i_{mk}, C^j_{nk})$;
// Re-clustering the multipartite network
4 Run $RAlgo$ to re-cluster vertices in $GP$ and discover a
5 \hspace{1em} meta-community structure, $\mathcal{C}_{GP} = \{C^l_{GP}\}_{l=1}^L$;
// Constructing an association matrix
5 Construct an association matrix $A_{|V| \times L}$, where $A(v, l) = F(v, C^l_{GP})$, indicating the association of vertex $v$ to a meta-community $C^l_{GP}$;
// Discovering final community structure
6 Each row in $A$ indicates the memberships of the corresponding vertex
7 \hspace{1em} in $L$ meta-communities;
8 To get $\mathbb{D}C$, we assign a vertex $v$ to community $C^* = \text{argmax}_C A(v, C)$;
9 To get $\mathbb{O}C$, we assign a vertex $v$ to a set of communities $C^*_v$ so that
10 \hspace{1em} $\forall C \in C^*_v : A(v, C) \geq \tau$;
11 To get $\mathbb{F}C$, we first normalize each entry in $A$ by the sum of entries in
12 \hspace{1em} the corresponding row and obtain a normalized association matrix $\hat{A}$, i.e., $
13 \hat{A}(v, l) = \frac{A(v, l)}{\sum_{l' \in L} A(v, l')}$; and assign a vertex $v$ to a community $C$
14 \hspace{1em} with the membership probability of $\hat{A}(v, C)$;
10 return $\mathbb{D}C$, $\mathbb{O}C$, $\mathbb{F}C$
Figure 2: A toy example depicting the work-flow of MeDOC++ algorithm. The broken lines indicate the community boundaries. Assume that the base algorithms produce three different community structures in Step 2, and we use Jaccard Similarity ($JC$) and simple association ($F$) as matching and association functions respectively. The definitions of $JC$ and $F$ are described in Section 4.2. The threshold $\tau$ is chosen as $\frac{1}{6}$.

4. MEDOC++: Meta-clustering Approach

MedOC++ (Meta Clustering based Disjoint, Overlapping and Fuzzy Community Detection) starts by executing all base community detection algorithms, each with different vertex orderings, to generate a set of community structures. It then creates a multipartite network. After this, another community detection algorithm is used to partition the multipartite network. Finally, a vertex-to-community association function is used to determine the strength of membership of a vertex in a community. Unlike EnDisCo, MeDOC++ can yield disjoint, overlapping and fuzzy community structures from the network.

4.1. Algorithmic Description

MedOC++ has the following four basic steps (pseudo-code is in Algorithm 2) a toy example of the work-flow of MeDOC++ presented in Figure 2 and important notations are shown in Table 1:

(i) Constructing multipartite network. MeDOC++ takes a set $\mathcal{AC} = \{A_{lm}\}_{m=1}^{M}$ of $M$ base community detection algorithms as input and runs each $A_{lm}$ on $K$ different vertex orderings of $G$. For each ordering $k$, $A_{lm}$
produces a community structure \( C^m_k = \{C^{1k}_m, ..., C^{ak}_m\} \) of varying size (Step 2). After running on \( K \) vertex orderings, each algorithm \( Al_m \) produces \( K \) different community structures \( \Gamma_m = \{C^k_m\}_{k=1}^K \). Therefore at the end of Step 2, we obtain \( K \) community structures each from \( M \) algorithms (essentially, we have \( P = M.K \) community structures). We now construct a \( P \)-partite network (aka meta-network) \( GP \) as follows: vertices are members of \( \bigcup_m C^m_k \), i.e., a community present in a base community structure (obtained from any of the base algorithms in \( \mathcal{A} \) and any vertex ordering) is a vertex of \( GP \). We draw an edge from a community \( C^{ik}_m \) to a community \( C^{jk'}_n \) and associate a weight \( W(C^{ik}_m, C^{jk'}_n) \) (Step 3). Possible definitions of \( W \) will be given later in Section 4.2. Since each \( C^m_k \) is disjoint, the vertices in each partition are never connected.

(ii) Re-clustering the multipartite network. Here we run any standard community detection algorithm \( RAlgo \) on the multipartite network \( GP \) and obtain a community structure containing (say) \( L \) communities \( \mathcal{C}_{GP} = \{C^l_{GP}\}_{l=1}^L \). Note that in this step, we cluster the communities obtained earlier in Step 2; therefore each such community \( C^l_{GP} \) obtained here is called a “meta-community” (or community of communities) (Step 4).

(iii) Constructing an association matrix. We determine the association between a vertex \( v \) and a meta-community \( C^l_{GP} \) by using a function \( F \) and construct an association matrix \( A \) of size \( |V| \times L \), where each entry \( A(v, l) = F(v, C^l_{GP}) \) (Step 5). Possible definitions of \( F \) will be given later in Section 4.2.

(iv) Discovering final community structure. Final vertex-to-community assignment is performed by processing \( A \). The entries in each row of \( A \) denote membership probabilities of the corresponding vertex in \( L \) communities. For disjoint community assignment, we label each vertex \( v \) by the community \( l \) in which \( v \) possesses the most probable membership in \( A \), i.e., \( l^* = \arg\max_l A(v, l) \). Tie-breaking is handled by assigning the vertex to the community to which most of its direct neighbors belong. Note that not every meta-community can be guaranteed to contain at least one vertex. Thus, we cannot guarantee that there will be \( L \) communities in the final community structure. For discovering overlapping community structure, we assign a vertex \( v \) to those communities for which the membership probability exceeds a
threshold $\tau$. Possible ways to specify this threshold will be specified later in Section 4.2.

For fuzzy community detection, we first normalize each entry of $A(v,l)$ by the sum of entries in the corresponding row so that the membership probability of each vertex in different communities sums up to 1. This in turn returns a new association matrix $\hat{A}(v,l)$. Accordingly we assign each vertex $v$ to a community $C$ with the membership probability of $\hat{A}(v,C)$ (see Step 9).

### 4.2. Parameter Selection

Here we describe the parameters used in MeDOC++ algorithm:

- **Matching Function ($W$):** Given two communities $C_i$ and $C_j$, we measure their matching/similarity via Jaccard Coefficient ($JC$) $\frac{|C_i \cap C_j|}{|C_i \cup C_j|}$ and average precision ($AP$) $\frac{1}{2} \left( \frac{|C_i \cap C_j|}{|C_i|} + \frac{|C_i \cap C_j|}{|C_j|} \right)$.

  **Example 4.1.** In Figure 2 the Jaccard Coefficient between $C_{11}$ and $C_{21}$ is $JC(C_{11},C_{21}) = \frac{3}{4}$. The average precision between them is $AP(C_{11},C_{21}) = \frac{1}{2} \left( \frac{3}{4} + \frac{3}{3} \right) = \frac{7}{8}$.

- **Association Function ($F$):** Given a meta-community $C$ consisting of (say,) $\gamma$ communities, the association of $v$ with $C$ can be calculated as $F(v,C) = \frac{\sum_{i=1}^{\gamma} \delta(v,C_i)}{\gamma}$, where $\delta$ returns 1 if $v$ is a part of $C_i$, 0 otherwise. Alternatively, a weighted association measure may assign a score to $v$ w.r.t. $C$ based on the co-occurrence of the other community members with $v$, i.e., $F_w(v,C) = \frac{\left| \bigcap_{\ell \in C} \bigcap_{i \in C} \delta(v,C_i) \right|}{\left| \bigcup_{\ell \in C} \bigcup_{i \in C} \delta(v,C_i) \right|}$.

  **Example 4.2.** In Figure 2 the simple association between vertex $A$ and meta community $MC_1$ is $F(A,MC_1) = \frac{3}{3} = 1$ because $A$ is present in $C_{11}$, $C_{21}$ and $C_{31}$ communities which are parts of $MC_1$. On the other hand, the weighted association between $A$ and $MC_1$ is $F_w(A,MC_1) = \frac{\left| \bigcap_{i \in C} \delta(v,C_i) \right|}{\left| \bigcup_{i \in C} \delta(v,C_i) \right|} = \frac{3}{4}$.

- **Threshold ($\tau$):** We choose the threshold $\tau$ automatically as follows. We first assign each vertex to its most probable community – this produces a disjoint community structure. Each vertex $v_i$ is represented by a feature vector $F(v_i)$ which is the entire $i$'th row of the association
matrix $A$. We then measure the average similarity of vertices in $C$ as follows: $\text{AS}(C) = \frac{\sum_{(u,v) \mid u,v \in C \wedge E_{uv} \in E_C} \text{COS}(F(u), F(v))}{|E_C|}$, where $E_C$ is the set of edges completely internal to $C$, $E_{uv}$ is an edge $(u, v)$, and $\text{COS}$ is cosine similarity. The probability that two vertices are connected in $C$ is then defined as:

$$P(C) = \frac{e^{[\text{AS}(C)]^2}}{1 + e^{[\text{AS}(C)]^2}}$$  \hspace{1cm} (2)

For a vertex $v$, if $P(C \cup \{v\}) \geq P(C)$, we further assign $v$ to $C$, in addition to its current community.

**Example 4.3.** In Figure 2, let us measure the threshold for community $DC1$ that we obtain in the final step after discovering the disjoint community structure. From the association matrix $A$ in the figure, $F(A) = \{1, 0\}$, $F(B) = \{1, 0\}$ and $F(C) = \{1, 0\}$. So the average similarity of vertices in $DC1$ in terms of cosine similarity is $\text{AS}(DC1) = 1$. Then the probability of two vertices in $DC1$ being connected is $P(\text{DC1}) = 0.88$. If we add $D$ to $DC1$, the new probability $P(\text{DC1} \cup D) = 0.80$ and $P(\text{DC1} \cup D) < P(\text{DC1})$. Therefore, $D$ is not assigned to $DC1$.

It is worth noting that the selection of a threshold depends upon which community we start with. For instance, in Figure 2, if we start from community $DC2$ and calculate the increase in probability after assigning $B$ to it, it would treat this assignment as a valid assignment. If we continue assigning the other vertices, the entire network would get assigned to a single community. To avoid this situation, we start from that community for which the membership probability $P(C)$ is highest among all and keep assigning other vertices to it. In Figure 2, we start from $DC1$. Once the members in a given community are finalized, we will not perturb their community membership further. This in turn also reduces runtime.

We compare our threshold selection method with the following method: each vertex is assigned to its top $n\%$ high probable communities (we set $n$ to 5% or 10%). Our experiments show that MeDOC++ delivers excellent performance with our threshold selection method (see Figures 4(g) and 4(i)).

Other input parameters $RAlgo$ and $K$ remain same as discussed in Section 3.2.
Table 2: Properties of the real-world networks with disjoint community structure. N: number of vertices, E: number of edges, C: number of communities, ρ: average edge-density per community, S: average size of a community.

| Network   | N   | E     | C | ρ  | S    |
|-----------|-----|-------|---|----|------|
| University| 81  | 817   | 3 | 0.54 | 27   |
| Football  | 115 | 613   | 12| 0.64| 9.66 |
| Railway   | 301 | 1,224 | 21| 0.24| 13.26|
| Coauthorship| 103,677 | 352,183 | 24| 0.14| 3762.58|

4.3. Complexity Analysis

The most expensive step of MeDOC++ is to construct the multipartite network in Step 3. If M is the number of base algorithms, K is the number of vertex orderings and \( \bar{a} \) is the average size of a base community structure, the worst case scenario occurs when each vertex in one partition is connected to each vertex in other partitions — if this happens, the total number of edges is \( \mathcal{O}(\bar{a}^2M^2K^2) \). However, in practice the network is extremely sparse and leads to \( \mathcal{O}(\bar{a}MK) \) edges (because in sparse graphs \( \mathcal{O}(|V|) \sim \mathcal{O}(|E|) \)). Further, constructing the association matrix would take \( \mathcal{O}(NL) \) iterations (where \( L \ll N \)).

5. Experiments: Results of Disjoint Community Detection

In this section, we evaluate EnDisCo and MeDOC++ for disjoint community detection. We start by explaining the datasets used in this experiment, followed by the baseline algorithms taken to compare against our algorithms, and the evaluation metrics used to compare the detected communities with the ground-truth. Then in the experimental results we will show how we select the parameters and the comparative analysis.

5.1. Datasets

We use both synthetic networks with community structures embedded, and real-world networks of different sizes with known ground-truth community structure.

5.1.1. Synthetic Networks

We use the LFR benchmark model [70] to generate synthetic networks with ground-truth community structure by varying the number of vertices \( n \), mixing parameter \( \mu \) (the ratio of inter- and intra-community edges), average
degree $\bar{k}$, maximum degree $k_{\max}$, minimum (maximum) community size $c_{\min}$ ($c_{\max}$), average percentage $O_n$ of overlapping vertices and the average number $O_m$ of communities to which a vertex belongs. The parameter $\mu$ controls the quality of the community structure – the more the value of $\mu$, the more the inter-community edges and the less the quality of the community structure. We vary this parameter in order to generate different network and community structures. We also vary $O_m$ and $O_n$ to obtain communities in different extent of overlapping (see Section 6). Unless otherwise stated, we generate networks with the same parameter configuration used in [4, 67] for disjoint community structure: $n = 10000$, $\bar{k} = 50$, $k_{\max} = 150$, $\mu = 0.3$, $O_n = 0$, $O_m = 1$, $c_{\max} = 100$, $c_{\min} = 20$. Note that for each parameter configuration, we generate 50 LFR networks, and the values in all the experiments are reported by averaging the results.

5.1.2. Real-world Networks

We also use the following four real-world networks mentioned in Table 2 for experiments:

**Football network:** This network constructed from [16] contains the network of American football games between Division IA colleges during the regular season of Fall 2000. The vertices in the network represent teams (identified by their college names) and edges represent regular-season games between the two teams they connect. The teams are divided into conferences (indicating communities) containing around 8-12 teams each.

**Railway network:** This network proposed by [4] consists of vertices representing railway stations in India, where two stations $s_i$ and $s_j$ are connected by an edge if there exists at least one train-route such that both $s_i$ and $s_j$ are scheduled halts on that route. Here the communities are states/provinces of India since the number of trains within each state is much higher than the trains in-between two states.

**University network:** This network generated by [52] is a friendship network of a faculty of a UK university, consisting of 81 vertices (individuals) and 817 connections. The school affiliation of each individual is stored as a vertex attribute. Schools act as communities in this network.

**Coauthorship network:** This network suggested by Chakraborty et al. [4] is derived from the citation dataset [71]. Here each vertex represents an author. An undirected edge between authors is drawn if the two authors coauthor at least one paper. The communities are marked by the research fields since authors have a tendency to collaborate with other authors within
the same field. It may be possible that an author has worked on multiple fields, which causes the communities to overlap. We assign an author to that research community in which he/she has published the most papers. However, later in Section 7 for fuzzy community detection we will leverage this information to prepare the fuzzy ground-truth community structure.

5.2. Baseline Algorithms

There exist numerous community detection algorithms which differ in the way they define community structure. Here we select the following set of algorithms as our baselines and categorize them according to the principle they use to identify communities as per [72]: (i) Modularity-based approaches: FastGreedy (FastGrdy) [18], Louvain (Louvain) [15] and CNM [16]; (ii) Vertex similarity-based approaches: WalkTrap (WalkTrap) [26]; (iii) Compression-based approaches: InfoMap (InfoMap) [28]; (iv) Diffusion-based approaches: Label Propagation (LabelPr) [29]; (v) Ensemble approaches: The most recent ensemble-based disjoint community detection algorithm is Consensus Clustering (ConsCl) [8]. This algorithm starts by running a base algorithm multiple times and generates a consensus matrix, i.e., a matrix based on the co-occurrence of vertices in communities of the input partitions. The consensus matrix is further used as an input to the base algorithm adopted, leading to a new set of partitions, which generate a new consensus matrix, until a unique partition is finally reached, which cannot be altered by further iterations.

Note that all these algorithms, except consensus clustering are also used as base algorithms in AL in our ensemble approaches.

5.3. Evaluation Metrics

Since the ground-truth community structure is available for each network, we use the following two standard metrics to compare the detected community structure with the ground-truth: Normalized Mutual Information (NMI) [73] and Adjusted Rand Index (ARI) [74]. The larger the value of NMI and ARI, the better the matching between two community structures.

5.4. Experimental Results

We first run experiments to identify the best parameters for EnDisCo and MeDOC++ for disjoint community detection and then compare them with competing algorithms.
Figure 3: Dependencies of the performance of EnDisCo (left panel) and MeDOC++ (right panel) on different parameters. The quality of the ground-truth community is varied by changing \( \mu \) from 0.1 to 0.8 (keeping the other LFR parameters default) and the performance is measured using NMI. In (c) and (f), we vary \( K \) and report the accuracy for three different LFR and Football networks (the results are similar for the other real networks and are not shown). The value corresponding to one parameter is reported by averaging the values for all possible combinations of the other parameters. The results are statistically significant (for multiple curves in (c) and (f), we report the range of \( p \)-values).

5.4.1. Dependency on the Parameters

We consider the LFR networks and vary \( \mu \). Figure 3(a) shows that the accuracy of EnDisCo is similar for both the involvement functions, while Figure 3(b) shows that cosine similarity outperforms Chebyshev similarity. Figure 3(d) shows that Jaccard coefficient performs significantly better than average precision when MeDOC++ is considered, while Figure 3(e) shows that the weighted association function is superior to the other for \( \mu < 0.6 \) and exhibits similar performance for \( \mu \geq 0.6 \). We further vary the number of iterations \( K \) to obtain communities with different vertex orderings – Figures 3(c) and 3(f) show that for the networks with strong community structure (such as LFR (\( \mu = 0.1 \), Football), the accuracy levels off at \( K = 0.2|V| \); however with increasing \( \mu \) leveling off occurs at larger values of \( K \). Note that the patterns observed here for the LFR network are similar for other networks. Therefore unless otherwise stated, in the rest of the experiment we
show the results of our algorithms with the following parameter settings for disjoint community detection: \textbf{EnDisCo}: $K = 0.2|V|$, \textit{RCC}, \textit{COS}; \textbf{MeDOC++}: $K = 0.2|V|$, \textit{JC}, \textit{Fw}.

5.4.2. Impact of Base Community Detection Algorithms on \textbf{EnDisCo} and \textbf{MeDOC++}

In order to assess the impact of each base algorithm in our ensemble, we measure the accuracy of \textbf{EnDisCo} and \textbf{MeDOC++} when that base algorithm is removed from the ensemble in isolation — Table 3 shows that for LFR and real-world networks \textbf{InfoMap} has the biggest impact on accuracy according to both the evaluation measures (NMI and ARI) for both \textbf{EnDisCo} and \textbf{MeDOC++}. We also observe that the overall accuracy of the ensemble algorithms decrease after removing each base algorithm. Therefore, irrespective of the quality of the base algorithms, we might need all of them to obtain high quality community structures as output.

However, from this result it is not clear — (i) whether we need all the outputs obtained from $K$ vertex orderings of a base algorithm, (ii) whether a certain combination of the base algorithms would produce the same accuracy as that obtained from using all the base algorithms. We will discuss more on these issues in Section 8.

5.4.3. Impact of Re-clustering Algorithms on \textbf{EnDisCo} and \textbf{MeDOC++}

As the final step in both \textbf{EnDisCo} and \textbf{MeDOC++} is to run an algorithm for re-clustering, we also conduct experiments to identify the best re-clustering algorithm. Table 4 shows that for both LFR networks and real networks, \textbf{InfoMap} is the best re-clustering algorithm.

5.4.4. Comparative Evaluation

Table 5 and Table 6 report the performance of our approaches on synthetic and real-world networks respectively using different algorithms in the final step of \textbf{EnDisCo}, \textbf{MeDOC++} and \textbf{ConsCl} for synthetic networks. The numbers denote relative performance improvement of \textbf{EnDisCo}, \textbf{MeDOC++} and \textbf{ConsCl} with respect to a given algorithm when that algorithm is used in the final step. For instance, the last entry in the last row (7.82) means that for LFR ($\mu = 0.6$) network, the accuracy of \textbf{MeDOC++} (when \textbf{LabelPr} is used for re-clustering in its final step) averaged over NMI and ARI is 7.82% higher than that of the standalone \textbf{LabelPr}. The actual values are reported in Table 7. The point to take away from this table is that irrespective of which
Table 3: Impact of each base algorithm on the accuracy of EnDisCo and MeDOC++. The results are reported on default LFR and real networks with default parameter settings of the proposed algorithms (we use InfoMap as the final re-clustering algorithm). Each base algorithm is removed in isolation during the construction of ensemble matrix.

(a) LFR Network

| No. | Base Algorithm | Disjoint | Overlapping |
|-----|----------------|----------|-------------|
|     |                | EnDisCo  | MeDOC++     | MeDOC++     |
|     |                | NMI ARI  | NMI ARI     | ONMI \(\Omega\) |
| (1) | All            | 0.85 0.89| 0.87 0.90   | 0.84 0.87   |
| (2) | (1) - FstGrdy  | 0.83 0.88| 0.84 0.88   | 0.83 0.85   |
| (3) | (1) - Louvain  | 0.82 0.86| 0.85 0.86   | 0.81 0.84   |
| (4) | (1) - CNM      | 0.82 0.85| 0.83 0.87   | 0.82 0.85   |
| (5) | (1) - InfoMap  | 0.80 0.81| 0.81 0.82   | 0.80 0.81   |
| (6) | (1) - WalkTrap | 0.84 0.88| 0.85 0.81   | 0.83 0.86   |
| (7) | (1) - LabelPr  | 0.84 0.87| 0.86 0.87   | 0.83 0.85   |

(b) Real-world Network

| No. | Base Algorithm | Football | Senate      |
|-----|----------------|----------|-------------|
|     |                | EnDisCo  | MeDOC++     | MeDOC++     |
|     |                | NMI ARI  | NMI ARI     | ONMI \(\Omega\) |
| (1) | All            | 0.90 0.92| 0.92 0.93   | 0.81 0.85   |
| (2) | (1) - FstGrdy  | 0.89 0.90| 0.91 0.90   | 0.80 0.83   |
| (3) | (1) - Louvain  | 0.87 0.86| 0.88 0.91   | 0.78 0.80   |
| (4) | (1) - CNM      | 0.86 0.87| 0.88 0.91   | 0.79 0.82   |
| (5) | (1) - InfoMap  | 0.84 0.87| 0.85 0.83   | 0.76 0.79   |
| (6) | (1) - WalkTrap | 0.89 0.91| 0.89 0.88   | 0.80 0.81   |
| (7) | (1) - LabelPr  | 0.89 0.90| 0.91 0.90   | 0.80 0.81   |

classical community detection algorithm we compare against, EnDisCo and MeDOC++ always improve the quality of communities found. Moreover, our proposed algorithms outperform ConsCl for all the networks. We further observe from the results of LFR networks that with the deterioration of the community structure (increase of \(\mu\)), the improvement increases for all the re-clustering algorithms. This essentially indicates that ensemble based approaches are even more useful if the underlying community structure is not well-separated.

We further compare EnDisCo and MeDOC++ with the performance of competing algorithms on real-world networks. Table 6 presents the same patterns observed for synthetic networks. Once again, our proposed algorithms outperform both ConsCl and the other standalone algorithms. For the Football network, our algorithms perform as well as other baseline algorithms, because the underlying community structure is very clear in the Football network [16]. Interestingly, our algorithms exhibit better performance for the Coauthorship network which has weaker community structure [4].

Both the results on synthetic and real-world networks lead to the conclu-
Table 4: Impact of each algorithm at the final stage of EnDisCo and MeDOC++ to re-cluster vertices. The results are reported on (a) default LFR network and (b) two real-world networks with the default parameter values of the proposed algorithms.

(a) LFR Network

| Re-clustering Algorithm | Disjoint | Overlapping |
|-------------------------|---------|-------------|
|                         | NMI     | ARI         | NMI     | ARI         | ONMI | Ω |
| FstGrdy                 | 0.79    | 0.80        | 0.80    | 0.83        | 0.81  | 0.84 |
| Louvain                 | 0.82    | 0.84        | 0.83    | 0.86        | 0.82  | 0.83 |
| CNM                     | 0.83    | 0.81        | 0.83    | 0.86        | 0.81  | 0.80 |
| InfoMap                 | 0.85    | 0.89        | 0.87    | 0.90        | 0.84  | 0.87 |
| WalkTrap                | 0.75    | 0.78        | 0.77    | 0.82        | 0.76  | 0.79 |
| LabelPr                 | 0.77    | 0.79        | 0.78    | 0.80        | 0.75  | 0.77 |

(b) Real-world Network

| Re-clustering Algorithm | Football | Senate |
|-------------------------|----------|--------|
|                         | NMI     | ARI | NMI | ARI | ONMI | Ω |
| FstGrdy                 | 0.86    | 0.89 | 0.87 | 0.91 | 0.78 | 0.77 |
| Louvain                 | 0.87    | 0.91 | 0.88 | 0.89 | 0.79 | 0.84 |
| CNM                     | 0.87    | 0.90 | 0.89 | 0.90 | 0.80 | 0.81 |
| InfoMap                 | 0.90    | 0.92 | 0.92 | 0.93 | 0.81 | 0.85 |
| WalkTrap                | 0.81    | 0.87 | 0.84 | 0.82 | 0.76 | 0.79 |
| LabelPr                 | 0.78    | 0.79 | 0.80 | 0.82 | 0.75 | 0.78 |

Table 5: Relative percentage improvement (averaged over NMI and ARI) of ConsCl, EnDisCo and MeDOC++ over the baseline algorithms for disjoint community detection from synthetic networks. Each row corresponds to an algorithm Al and the value indicates the performance improvement of the ensemble approach with Al as the re-clustering algorithm over the isolated performance of Al without ensemble.

| Algorithm | LFR ($\mu = 0.1$) | LFR ($\mu = 0.3$) | LFR ($\mu = 0.6$) |
|-----------|--------------------|--------------------|--------------------|
|           | ConsCl EnDisCo MeDOC++ | ConsCl EnDisCo MeDOC++ | ConsCl EnDisCo MeDOC++ |
| FstGrdy   | 1.92 2.39 2.93       | 2.93               | 3.02               |
| Louvain   | 1.86 1.97 2.04       | 2.04               | 2.40               |
| CNM       | 1.98 2.07 2.46       | 2.03               | 2.83               |
| InfoMap   | 0 0 0               | 0.98               | 1.44               |
| WalkTrap  | 3.43 4.43 4.97       | 3.91               | 5.08               |
| LabelPr   | 3.90 5.06 5.72       | 4.01               | 5.39               |

Conclusions that – (i) EnDisCo and MeDOC++ algorithms are quite competitive to the standalone algorithms for those networks which have prominent community structure; (ii) EnDisCo and MeDOC++ algorithms are more effective than the standalone algorithms for those networks whose underlying communities are weakly separated and difficult to detect by any traditional community detection algorithm.
Table 6: Relative percentage improvement (averaged over NMI and ARI) of ConsCl, EnDisCo and MeDOC++ over the baseline algorithms for disjoint community detection from real-world networks. Each row corresponds to an algorithm $Al$ and the value indicates the performance improvement of the ensemble approach with $Al$ as the re-clustering algorithm over the isolated performance of $Al$ without ensemble.

| Algorithm | Football | Railway | University | Coauthorship |
|-----------|----------|---------|------------|--------------|
| FstGrdy   | 0        | 0       | 0.96       | 1.22         |
| Louvain   | 0.84     | 1.23    | 1.10       | 1.72         |
| CNM       | 0        | 0       | 1.10       | 1.92         |
| InfoMap   | 1.42     | 2.21    | 2.13       | 3.21         |
| WalkTrap  | 2.23     | 3.01    | 2.21       | 3.46         |

Table 7: Actual accuracy values (in terms of NMI and ARI) of the proposed ensemble approaches (with default parameter setting) and ConsCl on both synthetic and real-world networks.

| Network   | ConsCl   | EnDisCo | MeDOC++ |
|-----------|----------|---------|---------|
| LFR($\mu=0.1$) | 0.89 | 0.91 | 0.94 |
| LFR($\mu=0.3$) | 0.85 | 0.86 | 0.89 |
| LFR($\mu=0.6$) | 0.76 | 0.79 | 0.84 |
| Football   | 0.90     | 0.92    | 0.92    |
| Railway    | 0.71     | 0.73    | 0.79 |
| University | 0.75     | 0.79    | 0.86   |
| Coauthorship | 0.61 | 0.65 | 0.70 |

6. Experiments: Results for Overlapping Community Detection

In this section, we evaluate MeDOC++ for overlapping community detection. We start by explaining the datasets used in this experiment, followed by the baseline algorithms used to compare our method and the evaluation metrics used to compare the detected communities with the ground-truth. We then show how to choose the best parameters followed by the comparative evaluation.

6.1. Datasets

Here we briefly describe the synthetic and real-world networks that we use in this experiment.

6.1.1. Synthetic Networks

We again use the LFR benchmark to generate synthetic networks with overlapping community structure with the following default parameter settings as mentioned in [30, 59]: $n = 10000$, $k = 50$, $k_{\text{max}} = 150$, $\mu = 0.3$, $\beta = 2$. For each network, we conduct a series of experiments by varying $\mu$ from 0.1 to 0.8, with an increment of 0.1. For each simulation, we run every algorithm 10 times and report the average NMI and ARI values.

6.1.2. Real-world Networks

Table 6 shows the relative percentage improvement of ConsCl, EnDisCo, and MeDOC++ over the baseline algorithms for disjoint community detection from real-world networks. Each row corresponds to an algorithm $Al$ and the value indicates the performance improvement of the ensemble approach with $Al$ as the re-clustering algorithm over the isolated performance of $Al$ without ensemble.
Table 8: Properties of the real-world networks with overlapping community structure. 

| Network     | $N$  | $E$   | $C$  | $\rho$ | $S$  | $O_m$ |
|-------------|------|-------|------|--------|------|-------|
| Senate      | 1,884| 16,662| 110  | 0.45   | 81.59| 4.76  |
| Flickr      | 80,513| 5,899,882| 171 | 0.046  | 470.83| 18.96 |
| Coauthorship| 391,526| 873,775| 8,493| 0.231  | 393.18| 10.45 |
| LiveJournal | 3,997,962| 34,681,189| 310,092| 0.536  | 40.02 | 3.09  |
| Orkut       | 3,072,441| 117,185,083| 6,288,363| 0.245  | 34.86 | 95.93 |

$O_n = 20\%$, $O_m = 20$, $c_{max} = 100$, $c_{min} = 20$. We generate 50 LFR networks for each parameter configuration — the experiments reported averages over these 50 networks. We further vary $\mu$ (0.1-0.8 with increment of 0.05), $O_m$ and $O_n$ (both from 15% to 30% with increment of 1%) depending upon the experimental need.

6.1.2. Real-world Networks

We also run experiments with following six real-world datasets mentioned in Table 8:

**Senate network:** This network combines voting pattern of 110 US-Senates [75, 76]. Each vertex represents a senator and the senators are connected in that session to their 3 nearest neighbors measured by voting similarities. The ground-truth communities are marked based on the senators who served in the same instance of the senate, i.e., senators who served during the same term.

**Flickr:** This dataset is built by forming links between images sharing common metadata from Flickr [77]. Edges are formed between images from the same location, submitted to the same gallery, group, or set, images sharing common tags, images taken by friends, etc. Communities are the user-specified groups.

**Coauthorship:** The coauthorship network [78] here is exactly the same as mentioned before for disjoint community structure in Section 5.1.2 except the ground-truth community marking which in this case is the publication venues (conferences or journals).

**LiveJournal:** LiveJournal is a free on-line blogging community where users declare their friends. LiveJournal also allows users to form a group which other members can then join. Here user-defined groups are considered.
as ground-truth communities. [45] provided the LiveJournal friendship social network and ground-truth communities.

**Orkut**: In this datasets, users in Orkut social networking site are nodes, and links are their friendships. Ground-truth communities are user-defined groups. [45] provided the Orkut friendship social network and ground-truth communities.

6.2. **Baseline Algorithms**

There are several standalone overlapping community detection algorithms, which are different based on the underlying working principle. We take six state-of-the-art algorithms from three different categories mentioned in [14]: (i) **Local expansion**: OSLOM [30] and EAGLE [40]; (ii) **Agent-based dynamical algorithms**: COPRA [59] and SLPA [32]; (iii) **Detection using mixture model**: MOSES [41] and BIGCLAM [45].

6.3. **Evaluation Metrics**

To compare the detected overlapping community structure with the ground-truth, we consider two standard validation metrics: Overlapping Normalized Mutual Information (ONMI) \(^1\) and Omega Index (Ω Index) \(^2\). The larger the value of ONMI and Omega index, the better the matching between two community structures.

6.4. **Experimental Results**

In this section, we present a detailed description of the comparative evaluation of the competing algorithms for overlapping community detection. We first describe the parameter selection process for MeDOC++, followed by the results showing the impact of the base algorithms on MeDOC++. We then present the performance of the competing algorithms.

6.4.1. **Parameter Settings**

We first try to identify the best parameter settings for MeDOC++. These include: matching function \(W\), association function \(F\), number of iterations \(K\) and threshold \(\tau\). Figure 4 shows the results on LFR networks by varying \(\mu\), \(O_m\) and \(O_n\). The results are almost identical with that of disjoint setting shown in Figure 3. We observe that Jaccard coefficient as matching

---

\(^1\) https://github.com/aaronmcdaid/Overlapping-NMI

---

27
function and weighted association measure are better than their respective alternatives. The choice of $K$ is the same as shown in Figure 3(f) – accuracy almost levels off at $K = 0.2|V|$. We experiment with two choices of thresholding: top 5% and 10% most probable communities per vertex, and compare with the threshold selection mechanism described in Section 4.2. Figures 4(g) and 4(i) show that irrespective of any network parameter selection, our choice of selecting threshold always outperforms others. As shown in Table 4, InfoMap seems to be the best choice for the re-clustering algorithm. Therefore unless otherwise stated, in the rest of the experiments, we run MeDOC++ with $K = 0.2|V|$, JC, $F_w$, InfoMap and $\tau$ (selected by our method).

6.4.2. Impact of Base Algorithms for Overlapping Community Detection

The impact of the base algorithms on MeDOC++’s performance is similar to what we saw in the disjoint community detection case. The results in Table 3 show that accuracy decreases most when we drop InfoMap from the base algorithm, followed by Louvain and CNM.
Table 9: Accuracy of all the competing algorithms in detecting the overlapping community structure from synthetic networks. All the disjoint algorithms are used to create the multipartite network and MeDOC++ is run with its default parameter setting.

| Algorithm | Synthetic Networks | | | | | |
|-----------|---------------------|---|---|---|---|---|
|           | LFR ($\mu = 0.1$)  | ONMI | $\Omega$ | ONMI | $\Omega$ | ONMI | $\Omega$ |
| OSLOM     | 0.80 | 0.78 | 0.74 | 0.78 | 0.72 | 0.73 |
| EAGLE     | 0.81 | 0.83 | 0.75 | 0.76 | 0.70 | 0.74 |
| COPRA     | 0.80 | 0.81 | 0.76 | 0.74 | 0.72 | 0.74 |
| SLPA      | 0.84 | 0.86 | 0.78 | 0.77 | 0.76 | 0.77 |
| MOSES     | 0.85 | 0.86 | 0.80 | 0.81 | 0.75 | 0.78 |
| BIGCLAM   | 0.86 | 0.85 | 0.81 | 0.83 | 0.77 | 0.79 |
| MeDOC++   | 0.88 | 0.91 | 0.84 | 0.87 | 0.82 | 0.84 |

Table 10: Accuracy of all the competing algorithms in detecting the overlapping community structure from real-world networks. All the disjoint algorithms are used to create the multipartite network and MeDOC++ is run with its default parameter setting.

| Algorithm | Real-world Networks | Senate | Flickr | Coauthorship | LiveJournal | Orkut |
|-----------|---------------------|--------|--------|--------------|-------------|-------|
|           | ONMI | $\Omega$ | ONMI | $\Omega$ | ONMI | $\Omega$ | ONMI | $\Omega$ | ONMI | $\Omega$ |
| OSLOM     | 0.71 | 0.73 | 0.68 | 0.74 | 0.70 | 0.71 | 0.71 | 0.75 | 0.71 | 0.76 |
| EAGLE     | 0.73 | 0.74 | 0.69 | 0.76 | 0.71 | 0.74 | 0.74 | 0.76 | 0.70 | 0.77 |
| COPRA     | 0.74 | 0.77 | 0.73 | 0.78 | 0.75 | 0.79 | 0.76 | 0.82 | 0.74 | 0.76 |
| SLPA      | 0.74 | 0.76 | 0.72 | 0.74 | 0.76 | 0.77 | 0.78 | 0.85 | 0.75 | 0.79 |
| MOSES     | 0.75 | 0.78 | 0.74 | 0.76 | 0.79 | 0.78 | 0.81 | 0.82 | 0.78 | 0.82 |
| BIGCLAM   | 0.76 | 0.79 | 0.75 | 0.76 | 0.80 | 0.84 | 0.84 | 0.87 | 0.81 | 0.84 |
| MeDOC++   | 0.81 | 0.85 | 0.79 | 0.84 | 0.82 | 0.86 | 0.86 | 0.88 | 0.83 | 0.86 |

6.4.3. Comparative Evaluation

We run MeDOC++ with the default setting on three LFR networks and five real-world networks. The performance of MeDOC++ is compared with the six baseline overlapping community detection algorithms. Table 9 shows the performance of the competing algorithms in terms of ONMI and $\Omega$ index for synthetic network. In all cases, MeDOC++ is a clear winner, winning by significant margins. The absolute average of ONMI ($\Omega$) for MeDOC++ over all synthetic networks is 0.85 (0.87), which is 4.50% (5.66%) higher than BIGCLAM, 6.25% (6.53%) higher than MOSES, 7.14% (8.75%) higher than SLPA, 11.84% (13.97%) higher than COPRA, 12.83% (12.01%) higher than EAGLE, and 12.38% (13.97%) higher than OSLOM. Another interesting observation is that for synthetic networks, the more the community structure deteriorates with the increase in $\mu$, the harder it becomes to detect the communities. It is in
then “hard to detect” cases that the performance of MeDOC++ significantly improves compared to the baselines. Another interesting observation is that the performance improvement seems to be prominent with the deterioration of community quality. For instance, the improvement of MeDOC++ with respect to the best baseline algorithm (BIGCLAM) is 2.32% (7.06%), 3.70% (4.82%) and 6.49% (6.33%) in terms of ONMI (Ω) with the increasing value of μ ranging from 0.1, 0.3 and 0.6 respectively. This once again corroborates our earlier observations in Section 5.4.4 that MeDOC++ is highly effective for those networks where the underlying community structure is not prominent and hard to detect.

In Table 10, we show the performance of the competing algorithms on real-world networks. MeDOC++ once again outperforms other competing methods. The average absolute ONMI of MeDOC++ over all networks is 0.82, which is followed by BIGCLAM (0.79), MOSES (0.77), OSLOM (0.76), SLPA (0.75), COPRA (0.74) and EAGLE (0.71). In short, MeDOC++ performs the best irrespective of any network and used validation measure.

7. Experiments: Results of Fuzzy Community Detection

In the case of overlapping communities, there are two different ways of defining overlap – in crisp overlapping in which each vertex belongs to one or more communities (the membership is binary – there is no notion of the strength of membership in a community); whereas in case of fuzzy overlapping, a vertex may also belong to more than one community but the strength of its membership to each community can vary. For instance, a person on Facebook might belong to multiple groups, but he may be much more active in one group compared to another. In this case, his degree of membership in that one group would be considered to be larger.

We earlier observed in Step 9 of MeDOC++ that it can assign each vertex \( v \) to a community \( C \) with a membership probability of \( \hat{A}(v, C) \). In this section, we provide a comprehensive analysis of the performance of MeDOC++ in detecting fuzzy community structure. We start by explaining the datasets used in this experiment, followed by the baseline algorithms and the evaluation metrics. We then describe the results of our comparative experimental evaluation.
7.1. Datasets

We first describe the construction of synthetic networks with fuzzy community structure, followed by the real-world network.

7.1.1. Synthetic Network

The synthetic network generated by the LFR model \cite{70} does not contain the fuzzy community structure. \cite{43} proposed a modified version of the LFR model to generate synthetic fuzzy community structure. Here we adopt their approach to generate synthetic networks. First, we generate crisp overlapping communities from the LFR model. Second, the crisp communities are converted to fuzzy form by adding a random membership probability to each occurrence of a vertex. The membership probabilities are chosen from a uniform distribution. Next a network is constructed from the fuzzy communities using the following formula:

\[ p_{ij} = s_{ij}p_1 + (1 - s_{ij})p_0 \] (3)

where \( p_{ij} \) is the probability of the existence of an edge \( e_{ij} \), \( s_{ij} \) is the co-membership of vertices \( i \) and \( j \); and \( p_{ij} = p_1 \) if \( \exists c \in C [i \in c \land j \in c] \); else \( p_0 \). In the above equation, \( p_0 \) and \( p_1 \) are chosen so as to preserve the specified average degree (\( \langle k \rangle \)) and mixing parameter (\( \mu \)) in the generated LFR network. The final network then satisfies all of the original parameters of LFR with the exception of the degree distribution (\( k_{\text{max}} \)), maximum degree and \( \tau_1 \), the exponent of the power-law distribution of vertex degrees). However, other parameters of LFR (such as \( \mu, O_m, O_n \) etc.) represent the same functionalities in this model. More details can be found in \cite{43}. Unless otherwise stated, we generate the synthetic networks with the following parameter setting: \( n = 10000, \bar{k} = 50, k_{\text{max}} = 150, \mu = 0.3, O_n = 20\%, O_m = 20, c_{\text{max}} = 100, c_{\text{min}} = 20. \)

7.1.2. Real-world Network

There is no real-world network where fuzzy community memberships of vertices are known. Therefore, the existing fuzzy community detection algorithms were mostly tested either with synthetic networks \cite{43,81}, or by calculating community evaluation metrics such as modularity \cite{82}. Here we

\footnote{We took the implementation of the synthetic model by the author available at 
http://www.cs.bris.ac.uk/~steve/networks/}
use the metadata information of the coauthorship network mentioned in Section 5.1.2 to construct the ground-truth. We recall that in the coauthorship network, authors are the vertices, edges are drawn based on coauthorship relations, and communities are different research areas. We then assign each author into a community with the community membership indicated by the fraction of papers the author has written on the corresponding research area. It also ensures that the sum of community memberships of each author is 1.

7.2. Baseline Algorithms

Fewer fuzzy methods have been proposed in the past. [52] presented “FuzzyClust” that maps the problem to a nonlinear constrained optimization problem and solves it. [46] used the fuzzy $c$-means algorithm to detect up to communities after converting the network into the feature space. [83] presented a method based on Bayesian non-negative matrix factorization (NMF). Finally, FOG [84] clusters “link data”, which includes networks as a special case, into fuzzy communities based on stochastic framework. [43] suggested “MakeFuzzy” algorithm which is used as a post-processing technique after a crisp overlapping algorithm to detect the fuzzy community structure. He further showed that MakeFuzzy along with EAGLE [40] outperforms other state-of-the-art algorithms.

In our experiment, we consider FuzzyClust algorithm of [52] and the NMF algorithm of [83] (with default parameters) as two baseline algorithms. We also consider MakFuzzy+EAGLE (henceforth, named as “M-E”) as another baseline to compare with MeDOC++.

7.3. Evaluation Metric

There exist very few metrics for comparing two fuzzy community structures. As per as we are aware, Fuzzy Rand Index (FRI) proposed by [85] is the only one metric for this purpose. Since this metric is used infrequently, we here explain this metric. This is a redefined version of the original Rand Index [13]:

$$RI_u(C_1, C_2) = \frac{s(C_1, C_2)}{N}$$  \hspace{1cm} (4)

where $s(C_1, C_2) = N - \sum_{i,j \in V} |f(i, j, C_1) - f(i, j, C_2)|$, and $f(i, j, C_1) = 1$ is vertices $v_i$ and $v_j$ appear in the same community, 0 otherwise. Then the expected Rand Index is defined as: $RI_e(C_1, C_2) = \frac{s(C_1)s(C_2) + (N-s(C_1))(N-s(C_2))}{N^2}$, where $N$ is the total number of vertices and $s(C) = \sum_{i,j \in V} f(i, j, C)$. 32
Table 11: Accuracy (in terms of Fuzzy Rand Index) of the fuzzy community detection algorithms for both synthetic and real-world networks. Synthetic networks are generated by varying $\mu$ and setting other parameters to default values. We run MeDOC++ in two settings – (i) MeDOC++: Algorithm 2 to detect fuzzy communities, (ii) MeDOC+++ MakeFuzzy: crisp overlapping communities are detected by MeDOC++, followed by M-E to post-process the output.

| Algorithm            | Synthetic | Real-world |
|----------------------|-----------|------------|
|                      | LFR ($\mu = 0.1$) | LFR ($\mu = 0.3$) | LFR ($\mu = 0.6$) | Coauthorship |
| FuzzyClust           | 0.71      | 0.68       | 0.65          | 0.62         |
| NMF                  | 0.74      | 0.70       | 0.68          | 0.65         |
| M-E                  | 0.78      | 0.74       | 0.71          | 0.67         |
| MeDOC++              | 0.78      | 0.75       | 0.70          | 0.68         |
| MeDOC+++ MakeFuzzy   | 0.78      | 0.76       | 0.73          | 0.68         |

The function $f(i, j, C)$ indicates the extent to which $i$ and $j$ appear in the same community in $C$, which depends on the membership probability of $v_i$ and $v_j$ as follows:

$$f(i, j, C) = 1 - \frac{1}{2} \sum_{c \in C} [\alpha_{ic} - \alpha_{jc}]$$  \hspace{1cm} (5)$$

where $\alpha_{ic}$ is the membership probability of $i$ in community $c$.

7.4. Experimental Results

We choose the same parameters for MeDOC++ as shown in Figure [4], i.e., $K = 0.2|V|$, $JC$ as pair-wise similarly of communities, $F_w$ as weighted association measure, InfoMap as re-clustering algorithm. Further, we consider two setups for MeDOC++: (i) MeDOC++ is run with the default parameter setting to detect the fuzzy community structure, (ii) MeDOC++ is run to detect the crisp overlapping community first, and then MakeFuzzy is used as a post-processing technique to detect the fuzzing overlapping communities (we call it MeDOC+++ MakeFuzzy).

Table 11 presents the results of three baseline algorithms along with two setups of MeDOC++. We observe that both the setups of MeDOC++ tend to be very competitive with the M-E, which seems to be the best baseline algorithm. However, incorporating MakeFuzzy into MeDOC++ outperforms other competing algorithms with a significant margin.
8. Selection of Base Outputs

In Section 5.4.2, we observed that removal of each base algorithm from the entire set reduces the overall accuracy of the ensemble algorithms with a certain extent. However, it was not clear (i) whether we need to consider all \( K \) outputs obtained from running each base algorithm \( K \) times, (ii) whether a subset of base algorithms are enough to get similar accuracy. In this section, we address these questions. In particular, we ask a general question - given a large set of different base solutions, how do we select a subset of solutions to form a smaller yet better performing ensemble algorithm than using all the available base solutions?

To this end, we investigate two properties of the detected solutions that have already been identified to be effective in literature \([86, 87, 88, 89]\) – the quality and the diversity.

8.1. Defining Quality and Diversity

**Quality.** Since the original communities to which vertices in a network belong are not known a priori, we propose to use an internal quality measure as follows. Given an ensemble solution \( E \) combining the set of all base community structures \( \Gamma = \{C^k_m\}, \forall m \in M \land k \in K \), the following quality function is used to measure the similarity of each solution with the ensemble:

\[
\text{Quality}(C^k_m, \Gamma) = \sum_{m' \in M} \sum_{n=1}^K Q(C^k_m, C^n_{m'})
\] (6)

\( Q \) can be any of the standard evaluation metrics such as NMI, ARI mentioned in Section 5.3 for disjoint communities and ONMI, Omega index in Section 6.3 for overlapping communities. However, we use NMI and ONMI for disjoint and overlapping communities respectively as suggested by Strehl and Ghosh \([90]\). Intuitively, Quality measures how well a particular base solution agrees with the general trend present in \( \Gamma \).

**Diversity.** There are many different diversity measures proposed for cluster ensembles in data mining \([91]\). However, to make the function consistent with the quality measure, we consider pair-wise NMI/ONMI among the base solutions. In particular, we measure the pair-wise similarity of two base solutions as \( Q(C^k_m, C^{k'}_{m'}) \) and compute the sum of all pair-wise similarities \( \sum_{i \neq j, i,j \in M \land k,k' \in K \land C^k_i,C^{k'}_j \in \Gamma} Q(C^k_i, C^{k'}_j) \). The lower the value, the higher the
diversity. We consider this diversity measure because it has already been shown to be effective for cluster ensemble [92].

Note that the base solution selection strategies that we will present here do not limit themselves to any particular quality and diversity functions.

8.2. Selection Strategies

Among the $MK$ number of solutions obtained from base community detection algorithms, we select $S$ solutions based on the following criteria individually:

8.2.1. Only Quality

This strategy simply ranks all solutions based on $Quality$ and selects top $S$ solutions to include in the ensemble algorithm. The solution with the highest $Quality$ essentially indicates that it has high consistency with the rest of the solutions. Generally, we expect that if we take only high quality solutions, due to the high similarity among them this strategy may lead to huge redundancy in the chosen set of solutions. This in turn reduces the ability to obtain improved results for those portions of the data which none of the selected solutions is able to capture properly. This explains the need for diversity amongst the solutions.

8.2.2. Only Diversity

We consider a greedy solution to select $S$ solutions which are highly diverse. We start by selecting the solution with highest $Quality$\footnote{However, one can start by selecting the solution which has highest pair-wise diversity. However, we observed that it ended up selecting poor solutions in the ensemble which leads to decrease in performance.}. We then incrementally add one solution at a time such that the resulting ensemble has the highest diversity. This process continues until the required number of solutions are chosen. It is commonly believed that diversifying the base solution is beneficial because mistakes made by one base algorithm may be compensated by another. However, it may result in the inclusion of low quality solutions in the ensemble. This is one of the reasons to choose the solution with highest quality first in the greedy strategy.
Figure 5: (a) A hypothetical network; (b) top three vertices found using PageRank are highlighted with large circles, and (c) top three nodes found using VRRW are highlighted with large circles. This example is taken from [94].

8.2.3. Combining Quality and Diversity

There is always a trade-off between quality and diversity in selecting base solutions, which can be viewed as a multi-objective optimization framework. A traditional way of handling such problems is to combine them into a single aggregate objective function. We choose \( S \) such solutions, denoted by \( C_S \) that maximize the following objective function:

\[
J = \alpha \sum_{c \in C_S} \text{Quality}(c, \Gamma) + (1 - \alpha) \sum_{c_i, c_j \in C_S, c_i \neq c_j} (1 - Q(c_i, c_j))
\]  

(7)

The parameter \( \alpha \) controls the trade-off between these two quantities. However, it is computationally expensive to select \( S \) solutions out of \( MK \) solutions [93]. Therefore, we adopt the following greedy strategy. We start by adding the solution with highest quality and incrementally add solutions one at a time that maximizes \( J \). We set 0.5 as the default value of \( \alpha \). However, we will examine different choices of \( \alpha \) in Figure 8.

8.2.4. PageRank-based Selection

This approach leverages the well-known PageRank algorithm to select the top \( S \) solutions which are of high quality but which are as diversely separated in the network as possible. We adopt Vertex Reinforced Random Walk (VRRW), a time heterogeneous random walk process used by [95]. Let us assume that we construct a network, where vertices are the base solutions and
the weight of an edge connecting two vertices \( i \) and \( j \) indicates the pair-wise diversity \((1 - Q)\) between corresponding base solutions. Unlike in traditional PageRank where the transition probabilities remain static throughout the iterations, in VRRW they change over time/iteration based on the following equation:

\[
p_T(i, j) = (1 - \lambda) \cdot p^*(j) + \lambda \cdot \frac{p_0(i, j) \cdot p_T(j)}{D_T(i)}
\]

where

\[
D_T(i) = \sum_{k \in V} p_0(i, j)p_T(k)
\]

Here, \( p_T(i, j) \) is the transition probability from vertex \( i \) to vertex \( j \) at time \( T \), \( p^*(j) \) is a distribution which represents the prior preference of visiting vertex \( j \), \( p_0(i, j) \) is the “organic” transition probability prior to any reinforcement. \( \lambda \) is set to 0.9 as suggested by \cite{94}. \( p_T(k) \) denotes the probability that the walk is at vertex \( k \) at time \( T \): \( p_T(k) = \sum_{(n,k) \in E} p_T(n,k)p_T-1(n) \). We set \( p^*(j) = Quality(C_j, \Gamma) \) and \( p_0(i, j) \) as follows:

\[
p_0(i, j) = \begin{cases} 
\alpha \cdot \frac{(1-Q(C_i,C_j))}{Weighted\_deg(C_i)} & \text{if } i \neq j \\
1 - \alpha, & \text{otherwise}
\end{cases}
\]

where \( Weighted\_deg(C_i) \) is the weighted degree of vertex \( i \), representing community structure \( C_i \). A schematic diagram of the VRRW process compared to PageRank is presented in Figure \ref{fig:vrrw_page_rank}. In this strategy, vertices are ranked based on VRRW score at the stationary state. Then we collect top \( S \) high ranked vertices, which in turn produces \( S \) high quality base communities which are diversely separated in the network.

8.3. Experimental Results

To evaluate the ensemble selection strategies, we apply each strategy on all the datasets separately for disjoint and overlapping community detections. The results are reported by averaging the values after ten independent runs. In Figures \ref{fig:ensemble_results} and \ref{fig:ensemble_results_overlapping} we plot the performance of different selection strategies as a function of ensemble size \( S \) (where \( S \) is represented as a fraction of the full ensemble set). We also show the results after considering all the base solutions in the ensemble set using the horizontal green line.

\footnote{Readers are encouraged to read the details in \cite{95}.}
It is evident from both these results that the full ensemble is always best. However, we might achieve the same performance by selecting a subset of the base solutions. Both “combining quality and diversity” and “VRRW” strategies seem to be more consistent and achieve promising performance toward our goal, that is to select smaller and better performing ensembles. Among them, the VRRW-based strategy tends to achieve the maximum accuracy with just 60-80% size of the full ensemble. For the large networks such as Coauthorship network in Figure 6(d), the separation of the performance is much prominent for EnDisCo and VRRW-based strategy reaches the maximum accuracy with 40% of the full ensemble. The reason might be that this strategy explicitly seeks to remove redundant solutions and retains quality solutions at the same time.

**Sensitivity of $\alpha$ for “combining quality and diversity” strategy:** So far we conducted all the experiments with $\alpha = 0.5$ for “combining quality and diversity” strategy. Here we examine how this strategy gets affected by varying the value of $\alpha$. We experiment with a variety of $\alpha$ values including $0, 0.1, \cdots, 0.9, 1$ and compare the results with “only quality” ($\alpha = 1$) and “only diversity” ($\alpha = 0$). The smaller values of $\alpha$ takes “diversity” into account; whereas larger values prefer “quality”. In Figure 8, we present the results of EnDisCo and MeDOC++ for both disjoint and overlapping community detection by varying the size of the selected base solutions and five different values of $\alpha (0, 0.3, 0.5, 0.7, 1)$. Note that each accuracy value shown here is the average of ten runs. In general, we observe that assigning very high or very low values to $\alpha$ might be beneficial for some cases, but in general the result is more stable and robust for $\alpha = 0.5$. We also observe that the accuracy never decreases with the increase of $S$ and gets saturated quickly for $\alpha = 0.5$.

**9. Other Implications of MeDOC++**

In this section, we present two other useful capabilities of MeDOC++. We show that MeDOC++ can be used to explore the core-periphery structure of communities in a network and to detect stable communities from dynamic networks.
Exploring Community-centric Core-periphery Structure

The association values of individual vertices in a community obtained from MeDOC++ provide additional information about the membership strength in that community. This phenomenon might be related to the core-periphery organization [96, 97] of the induced subgraph composed of the vertices in a community. We hypothesize that the more the association of a vertex in a community, the more the vertex is likely to be a core part of the community. To verify this, we first explore the core-periphery structure (also known as k-core decomposition [97]) of each community in a network.

Core-periphery organization: For each community detected by MeDOC++, we start by creating the induced subgraph composed of vertices in that community. We then recursively remove vertices which have degree one until no such degree-one vertices remain in the network. The removed vertices form the 1-shell of the network ($k^1$-shell index $k^1 = 1$). Similarly, we obtain 2-shell by recursively removing degree-two vertices from the resultant network obtained from the last stage. We keep on increasing the value of $k$ until all
Figure 7: Comparison of the performance of different selection strategies: OQ: only quality, OD: only diversity, Q+D: combining quality and diversity with $\alpha = 0.5$, VRRW: vertex reinforced random walk, with the variation of the size of the base solutions for overlapping community detection. We compare the performance of these strategies with the full ensemble (Full) represented by the solid green line.

The idea is to show how the association value of a vertex to a community is related to its community-centric shell-index\(^5\). For each network, we first detect the community using MeDOC++ and measure the association values of vertices. Then, for each detected community, we extract the core-periphery structure. In Figure 9, we present the correlation between association values of vertices with their positions in the core-periphery structure. For better visualization, we divided the total number of shells into three broad shells, and the range of association value into four buckets. We note that the inner core is highly dominated by the vertices with association value ranging $0.75 - 1$, whereas the outermost layer is dominated by vertices having association value $0 - 0.25$. Further, we observe that as we move from core to periphery, vertices with low association score start dominating. This result indicates that the association value derived from MeDOC++ has high correlation with

\(^{5}\)Note that the average core-periphery structure of communities in a network is different from the core-periphery structure of a network. Here we are interested in the former case.
9.2. Discovering Stable Communities in Dynamic Networks

Most real-world networks are dynamic—either vertices or edges get added to the network or get deleted over time. As a result, the community structure of the network might change over time. [98] argued that despite the change in community structure, there are some stable communities that persist. Such stable communities may be important for many reasons. For instance, in an election campaign, vertices in stable communities might be individuals who are strong supporters of a particular candidate. In human interaction networks, the stable communities might be the groups of individuals who are close friends or family members. For instance, there are a number of efforts [99] in which researchers have provided subjects with cell phones (or cell phone apps), and a temporal human interaction network is built by identifying mobile phones that are in close proximity with one another during a given time window. In such cases, communities that are stable between 8am
Figure 9: Core-periphery organization of the disjoint (top panel) and overlapping (bottom panel) communities detected by MeDOC++ and its relation with the association values. The colors represent different ranges of association values and the area covered by each colored region in each $k$-shell denotes the proportion of vertices with the corresponding association values occupied in that shell. The innermost shell is the core region and the outermost shell is the periphery region. For better visualization, we divide the total number of shells identified from the communities in each network into three broad shells and the range of association values into four buckets; thus the core-periphery structure in each network has three concentric layers and each layer is divided into four association ranges.

and 6pm might represent people who work together, while communities that are stable during the 8pm to 6am time frame might represent families. We hypothesize that the association values of individual vertices obtained via the MeDOC++ algorithm can discover stable communities in dynamic networks. In particular, vertices with an association value tending to 1 might form stable communities in a network.

To this end, we collect two dynamic networks with unknown ground-truth community structure\[^{[100]}\]: (i) Hypertext 2009 contact network: this is a dynamic network of face-to-face proximity over about 2.5 days of 110 attendees who attended ACM Hypertext 2009 conference, (ii) School contact network: this dataset contains the temporal network of contacts between 298 students over 4 days in a high school in Marseilles, France. We divide each network into five subnetworks based on the equal division of the temporal

[^{[100]}]: http://www.sociopatterns.org/datasets/
Table 12: Similarity of the stable communities of temporal networks in two consecutive time stamps obtained from MeDOC++ for two dynamic networks (Hypertext and school contact networ).

| Network  | Measure | $t_1 - t_2$ | $t_2 - t_3$ | $t_3 - t_4$ | $t_4 - t_5$ |
|----------|---------|-------------|-------------|-------------|-------------|
| Hypertext| NMI     | 0.79        | 0.76        | 0.80        | 0.82        |
|          | ARI     | 0.81        | 0.78        | 0.79        | 0.83        |
| School   | NMI     | 0.76        | 0.79        | 0.78        | 0.80        |
|          | ARI     | 0.75        | 0.76        | 0.82        | 0.83        |

dimension in such a way that number of vertices is same in each subnetwork. However the edge connectivity might vary across five time windows.

In each time window $t_i$, we run MeDOC++ on the corresponding network. We allow MeDOC++ to detect communities with the default parameter setting. Once we obtain the association matrix $A$ (in Step 5 of Algorithm 2), we considered only those vertices whose association score is exactly 1 and group them accordingly (resulting disjoint stable community structure). We believe that these vertices form the stable communities. We repeat the same procedure for each temporal network separately and detect the stable community structure. Then we measure the similarity (based on NMI and ARI) between stable community structures obtained from the temporal networks in two consecutive time windows ($t_i$ and $t_{i+1}$), where $i$ ranges from 1 to 4) of each dynamic network. In Table 12, we observe that the temporal similarity between two consecutive stable communities is quite high, and it remains almost same over time. This indicates that the stable community structure of a network do not vary much over the successive time periods, and MeDOC++, quite efficiently, detects such stable communities from dynamic networks.

10. Handling Degeneracy of Solutions

Most community finding algorithms suffer from the well-known “degeneracy of solutions” problem [10] which states that these algorithms typically produce exponentially many solutions with (nearly-)similar optimal value of the objective function (such as modularity); however the solutions may be structurally distinct from each other.

We hypothesize that since ensemble algorithms combine multiple views of the underlying community structure of a network, they might suffer less from the problem of degeneracy of solutions. We test this by considering the
real-world networks, and running each competing algorithms on 100 different vertex orderings of each network. We then measure the pair-wise similarity of the solutions obtained from each algorithm. The box plots in Figures 10 and 11 show the variation of the solutions for all the competing algorithms on both disjoint and overlapping community detections respectively. We observe that the median similarity is high with EnDisCo and MeDOC++ and the variation is significantly less. In fact, all the ensemble algorithms, i.e., EnDisCo, MeDOC++ and ConsCl show low variance. These results suggest that ensemble based algorithms always provide more robust results than standalone algorithms and alleviate the problem of degeneracy of solutions.

11. Runtime Analysis

Since ensemble approaches require the running all base algorithms (which may be parallelized), one cannot expect ensemble methods to be faster than standalone approaches. However, our proposed ensemble frameworks are much faster than existing ensemble approaches such as consensus clustering. To show this, for each ensemble algorithm, we report Θ, the ratio between the runtime of each ensemble approach and the sum of runtimes of all base algorithms, with increasing number of vertices in LFR. We vary the number
Figure 11: Box plots indicating the variation of the solutions obtained from the overlapping community finding algorithms on real-world networks.

Figure 12: The value of $\Theta$ w.r.t. the increase of vertices in LFR networks. $\text{EnDisCo}$ and $\text{MeDOC++}$ are compared with $\text{ConsCl}$. The results are statistically significant (since there are multiple curves, we report the range of $p$-values).

of edges of LFR by changing $\mu$ from 0.1 to 0.3. Figure 12 shows that our algorithms are much faster than consensus clustering. We further report the results of $\text{MeDOC++}$ for overlapping community detection which is almost same as that of disjoint case since it does not require additional steps apart from computing the threshold.
12. Conclusion

In this paper, we proposed two general frameworks for ensemble community detection. EnDisCo identifies disjoint community structures, while MeDOC++ detects both disjoint, overlapping and fuzzy community structures. MeDOC++ is the first ever ensemble overlapping (and fuzzy) community detection algorithms in the literature.

We tested our algorithms on both synthetic data using the LFR benchmark as well as with several real-world datasets that have an associated ground-truth community structure. We showed that both EnDisCo and MeDOC++ are more accurate than existing standalone community detection algorithms (though of course, EnDisCo and MeDOC++ leverage them by using the known community detection algorithms in the ensembles). We further showed that for disjoint community detection problems, EnDisCo and MeDOC++ both beat the best performing existing disjoint ensemble method called consensus clustering [8] – both in terms of accuracy and run-time. To our knowledge, MeDOC++ is the first ensemble algorithm for overlapping and fuzzy community detection that we have seen in the literature.

We further presented extensive analysis of how to select a subset of solutions to obtain near-optimal accuracy. The association values of vertices obtained from MeDOC++ help us exploring the core-periphery structure of the communities in a network and detecting the stable communities in dynamic networks. We showed that ensemble approaches generally reduce the effect of degeneracy of solution significantly.

References

[1] S. Fortunato, Community detection in graphs, Physics Reports 486 (2010) 75 – 174.

[2] M. E. Newman, Modularity and community structure in networks, PNAS 103 (2006) 8577–8582.

[3] V. A. Traag, G. Krings, P. V. Dooren, Significant scales in community structure, Scientific Reports 3 (2013).
[4] T. Chakraborty, S. Srinivasan, N. Ganguly, A. Mukherjee, S. Bhowmick, On the permanence of vertices in network communities, in: SIGKDD, New York, USA, pp. 1396–1405.

[5] T. Chakraborty, Leveraging disjoint communities for detecting overlapping community structure, Journal of Statistical Mechanics: Theory and Experiment 2015 (2015) P05017.

[6] T. Chakraborty, S. Srinivasan, N. Ganguly, A. Mukherjee, S. Bhowmick, Permanence and community structure in complex networks, ACM Trans. Knowl. Discov. Data 11 (2016) 14:1–14:34.

[7] J. Dahlin, P. Svenson, Ensemble approaches for improving community detection methods., CoRR abs/1309.0242 (2013).

[8] A. Lancichinetti, S. Fortunato, Consensus clustering in complex networks, Nature Scientific Reports 2 (2012).

[9] R. Xu, D. Wunsch, II, Survey of clustering algorithms, Trans. Neur. Netw. 16 (2005) 645–678.

[10] B. Good, Y. D. Montjoye, A. Clauset, Performance of modularity maximization in practical contexts, Physical Review E 81 (2010) 046106.

[11] T. Chakraborty, S. Srinivasan, N. Ganguly, S. Bhowmick, A. Mukherjee, Constant Communities in Complex Networks, Scientific Reports 3 (2013).

[12] T. Chakraborty, N. Park, V. Subrahmanian, Ensemble-based algorithms to detect disjoint and overlapping communities in networks, in: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 73–80.

[13] T. Chakraborty, A. Dalmia, A. Mukherjee, N. Ganguly, Metrics for community analysis: A survey, ACM Comput. Surv. 50 (2017) 54:1–54:37.

[14] J. Xie, S. Kelley, B. K. Szymanski, Overlapping community detection in networks: The state-of-the-art and comparative study, ACM Comput. Surv. 45 (2013) 43:1–43:35.
[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech (2008) P10008.

[16] A. Clauset, M. E. J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111.

[17] R. Guimera, L. A. N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (2005) 895–900.

[18] M. E. J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (2004) 066133.

[19] M. E. J. Newman, Community detection and graph partitioning, CoRR abs/1305.4974 (2013).

[20] T. Richardson, P. J. Mucha, M. A. Porter, Spectral tripartitioning of networks, Phys. Rev. E 40 (2009) 027104.

[21] I. Farkas, D. Ábel, G. Palla, T. Vicsek, Weighted network modules, New Journal of Physics 9 (2007) 180.

[22] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (2005) 814–818.

[23] J. Baumes, M. Goldberg, M. Magdon-Ismail, Efficient identification of overlapping communities, in: Proceedings of the 2005 IEEE international conference on Intelligence and Security Informatics, ISI’05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 27–36.

[24] A. Lancichinetti, S. Fortunato, J. Kertesz, Detecting the overlapping and hierarchical community structure of complex networks, ArXiv E-prints (2008).

[25] P. De Meo, E. Ferrara, G. Fiumara, A. Provetti, Enhancing community detection using a network weighting strategy, Journal of Information Science 222 (2013) 648–668.

[26] P. Pons, M. Latapy, Computing communities in large networks using random walks, J. Graph Algorithms Appl. 10 (2006) 191–218.
[27] M. Rosvall, C. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, PNAS 104 (2007) 7327.

[28] M. Rosvall, C. T. Bergstrom, Maps of random walks on complex networks reveal community structure, PNAS 105 (2008) 1118–1123.

[29] U. N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (2007) 036106.

[30] A. Lancichinetti, F. Radicchi, J. J. Ramasco, S. Fortunato, Finding statistically significant communities in networks, PLoS ONE 6 (2011) e18961.

[31] J. Xie, B. Szymanski, Community detection using a neighborhood strength driven label propagation algorithm, in: IEEE NSW, pp. 188–195.

[32] J. Xie, B. K. Szymanski, Towards linear time overlapping community detection in social networks, in: PAKDD, Malaysia, pp. 25–36.

[33] J. Xie, M. Chen, B. K. Szymanski, Labelrankt: Incremental community detection in dynamic networks via label propagation, CoRR abs/1305.2006 (2013).

[34] V. Kawadia, S. Sreenivasan, Sequential detection of temporal communities by estrangement confinement, Scientific Reports 2 (2012).

[35] B. Mitra, L. Tabourier, C. Roth, Intrinsically dynamic network communities, CoRR abs/1111.2018 (2011).

[36] D. A. Bader, H. Meyerhenke, P. Sanders, D. Wagner (Eds.), Graph Partitioning and Graph Clustering - 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, February 13-14, 2012. Proceedings, volume 588 of Contemporary Mathematics, American Mathematical Society, 2013.

[37] J. Riedy, D. A. Bader, K. Jiang, P. Pande, R. Sharma, Detecting Communities from Given Seeds in Social Networks, Technical Report GT-CSE-11-01, Georgia Institute of Technology, 2011.
[38] S. Fortunato, A. Lancichinetti, Community detection algorithms: A comparative analysis: Invited presentation, extended abstract, in: Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, ICST, Brussels, Belgium, Belgium, pp. 27:1–27:2.

[39] Y.-Y. Ahn, J. P. Bagrow, S. Lehmann, Link communities reveal multiscale complexity in networks, Nature 466 (2010) 761–764.

[40] K. C. H. Shen, X. Cheng, M. B. Hu, Detect overlapping and hierarchical community structure in networks, Physica A 388 (2009) 1706–1712.

[41] A. McDaid, N. Hurley, Detecting highly overlapping communities with model-based overlapping seed expansion, in: ASONAM, Washington, DC, USA, pp. 112–119.

[42] C. Lee, F. Reid, A. McDaid, N. Hurley, Detecting highly-overlapping community structure by greedy clique expansion, in: ACM KDD-SNA, pp. 33–42.

[43] S. Gregory, Fuzzy overlapping communities in networks, Journal of Statistical Mechanics: Theory and Experiment 2011 (2011) P02017.

[44] K. Zhao, S.-W. Zhang, Q. Pan, Fuzzy analysis for overlapping community structure of complex network, in: CCDC, pp. 3976–3981.

[45] J. Yang, J. Leskovec, Overlapping community detection at scale: A nonnegative matrix factorization approach, in: WSDM, ACM, New York, USA, 2013, pp. 587–596.

[46] S. Zhang, R.-S. Wang, X.-S. Zhang, Identification of overlapping community structure in complex networks using fuzzy c-means clustering, Physica A: Statistical Mechanics and its Applications 374 (2007) 483–490.

[47] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. Magdon-Ismail, N. Preston, Finding communities by clustering a graph into overlapping subgraphs., in: IADIS AC, IADIS, 2005, pp. 97–104.

[48] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New J. Phys. 11 (2009) 033015.
[49] F. Havemann, M. Heinz, A. Struck, J. Glser, Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels, JSTAT 2011 (2011) P01023.

[50] D. Chen, M. Shang, Z. Lv, Y. Fu, Detecting overlapping communities of weighted networks via a local algorithm, Physica A 389 (2010) 4177–4187.

[51] N. Du, B. Wang, B. Wu, Overlapping community structure detection in networks, in: CIKM, ACM, New York, USA, 2008, pp. 1371–1372.

[52] T. Nepusz, A. Petrócz, L. Négyessy, F. Bazsó, Fuzzy communities and the concept of bridgeness in complex networks, Phys. Rev. E 77 (2008) 016107.

[53] M. E. J. Newman, E. A. Leicht, Mixture models and exploratory analysis in networks, PNAS 104 (2007) 9564–9569.

[54] W. Ren, G. Yan, X. Liao, L. Xiao, Simple probabilistic algorithm for detecting community structure, Phys. Rev. E 79 (2009) 036111.

[55] K. Nowicki, T. A. B. Snijders, Estimation and prediction for stochastic blockstructures, Journal of the American Statistical Association 96 (2001) 1077–1087.

[56] M. Zarei, D. Izadi, K. A. Samani, Detecting overlapping community structure of networks based on vertex–vertex correlations, J. Stat. Mech. Theor. Exp. 2009 (2009) P11013.

[57] F. Ding, Z. Luo, J. Shi, X. Fang, Overlapping community detection by kernel-based fuzzy affinity propagation, in: ISA, pp. 1–4.

[58] J. J. Whang, D. F. Gleich, I. S. Dhillon, Overlapping community detection using seed set expansion, in: CIKM, ACM, 2013, pp. 2099–2108.

[59] S. Gregory, Finding overlapping communities in networks by label propagation, New J. Phys. 12 (2010) 103018.

[60] W. Chen, Z. Liu, X. Sun, Y. Wang, A game-theoretic framework to identify overlapping communities in social networks, Data Min. Knowl. Discov. 21 (2010) 224–240.
[61] S. Gregory, An algorithm to find overlapping community structure in networks, in: PKDD, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 91–102.

[62] M. Girvan, M. E. J. Newman, Community structure in social and biological networks, PNAS 99 (2002) 7821–7826.

[63] Y. Zhang, J. Wang, Y. Wang, L. Zhou, Parallel community detection on large networks with propinquity dynamics, in: KDD, ACM, 2009, pp. 997–1006.

[64] A. István, R. Palotai, M. S. Szalay, P. K. Csermely, Community landscapes: An integrative approach to determine overlapping network module hierarchy, identify key nodes and predict network dynamics, PLoS ONE 5 (2010) e12528.

[65] P.-G. Sun, L. Gao, S. S. Han, Identification of overlapping and non-overlapping community structure by fuzzy clustering in complex networks, Inf. Sci. 181 (2011) 1060–1071.

[66] M. Ovelgönne, A. Geyer-Schulz, An ensemble learning strategy for graph clustering, in: Graph Partitioning and Graph Clustering, volume 588 of Contemporary Mathematics, American Mathematical Society, 2012, pp. 187–206.

[67] R. Kanawati, Yasca: An ensemble-based approach for community detection in complex networks, in: COCOON, Springer, Cham, 2014, pp. 657–666.

[68] R. Kanawati, YASCA: A collective intelligence approach for community detection in complex networks, CoRR abs/1401.4472 (2014).

[69] R. Kanawati, Ensemble selection for community detection in complex networks, in: SCSM, Springer, CA, USA, 2015, pp. 138–147.

[70] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Phy. Rev. E 80 (2009) 016118.

[71] T. Chakraborty, S. Sikdar, V. Tammana, N. Ganguly, A. Mukherjee, Computer science fields as ground-truth communities: their impact, rise and fall, in: ASONAM, ACM, 2013, pp. 426–433.
[72] G. K. Orman, V. Labatut, H. Cherifi, Comparative evaluation of community detection algorithms: a topological approach, Journal of Statistical Mechanics: Theory and Experiment 2012 (2012) P08001.

[73] L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech. 9 (2005) P09008.

[74] L. Hubert, P. Arabie, Comparing partitions, Journal of classification 2 (1985) 193–218.

[75] L. G. S. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, M. W. Mahoney, Think locally, act locally: Detection of small, medium-sized, and large communities in large networks, Phys. Rev. E 91 (2015) 012821.

[76] K. Kloster, D. F. Gleich, Personalized pagerank solution paths, CoRR abs/1503.00322 (2015).

[77] X. Wang, L. Tang, H. Liu, L. Wang, Learning with multi-resolution overlapping communities, Knowl. Inf. Syst. 36 (2013) 517–535.

[78] G. Palla, I. J. Farkas, P. Pollner, I. Derényi, T. Vicsek, Fundamental statistical features and self-similar properties of tagged networks, New J. Phys. 10 (2008) 123026.

[79] L. M. Collins, C. W. Dent, Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions, Multivariate Behavioral Research 23 (1988) 231–242.

[80] G. Murray, G. Carenini, R. Ng, Using the omega index for evaluating abstractive community detection, in: Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012, pp. 10–18.

[81] C. Vehlow, T. Reinhardt, D. Weiskopf, Visualizing fuzzy overlapping communities in networks, IEEE Transactions on Visualization and Computer Graphics 19 (2013) 2486–2495.

[82] J. Su, T. C. Havens, Fuzzy community detection in social networks using a genetic algorithm, in: 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 2039–2046.
[83] I. Psorakis, S. Roberts, B. Sheldon, Efficient Bayesian Community Detection using Non-negative Matrix Factorisation (2010).

[84] G. B. Davis, K. M. Carley, Clearing the fog: Fuzzy, overlapping groups for social networks., Social Networks 30 (2008) 201–212.

[85] E. Hllermeier, M. Rifqi, S. Henzgen, R. Senge, Comparing fuzzy partitions: A generalization of the rand index and related measures., IEEE Trans. Fuzzy Systems 20 (2012) 546–556.

[86] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: A review, SIGKDD Explor. Newsl. 6 (2004) 90–105.

[87] L. I. Kuncheva, S. T. Hadjitodorov, Using diversity in cluster ensembles, in: IEEE International Conference on Systems, Man and Cybernetics, volume 2, pp. 1214–1219 vol.2.

[88] S. T. Hadjitodorov, L. I. Kuncheva, L. P. Todorova, Moderate diversity for better cluster ensembles, Inf. Fusion 7 (2006) 264–275.

[89] X. Z. Fern, W. Lin, Cluster ensemble selection, Statistical Analysis and Data Mining 1 (2008) 128–141.

[90] A. Strehl, J. Ghosh, Cluster ensembles — a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (2003) 583–617.

[91] J. Li, K. Yi, Q. Zhang, Clustering with Diversity, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 188–200.

[92] X. Z. Fern, C. E. Brodley, Random projection for high dimensional data clustering: A cluster ensemble approach., in: T. Fawcett, N. Mishra (Eds.), ICML, AAAI Press, 2003, pp. 186–193.

[93] X. Z. Fern, W. Lin, Cluster ensemble selection, in: SDM, SIAM, 2008, pp. 787–797.

[94] T. Chakraborty, N. Modani, R. Narayanam, S. Nagar, Discern: A diversified citation recommendation system for scientific queries, in: 2015 IEEE 31st International Conference on Data Engineering, pp. 555–566.
[95] Q. Mei, J. Guo, D. Radev, Divrank: The interplay of prestige and diversity in information networks, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’10, ACM, New York, NY, USA, 2010, pp. 1009–1018.

[96] X. Zhang, T. Martin, M. E. J. Newman, Identification of core-periphery structure in networks, Phys. Rev. E 91 (2015) 032803.

[97] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, E. Shir, From the Cover: A model of Internet topology using k-shell decomposition, PNAS 104 (2007) 11150–11154.

[98] C. C. L. K.-z. Y. D.-p. Chen Xiao-qiang, ZHOU Li-hua, Detecting stable communities in dynamic networks, Journal of Chinese Computer Systems 36 (2015) 1977.

[99] N. Eagle, A. Pentland, D. Lazer, Inferring Social Network Structure using Mobile Phone Data, PNAS (2007).

[100] T. Chakraborty, On the discovery of invariant groups in complex networks, J. Complex Networks 5 (2017) 734–749.