Ensemble-Based Algorithms to Detect Disjoint and Overlapping Communities in Networks

Tanmoy Chakraborty  
Dept. of Computer Science  
University of Maryland  
College Park, MD 20742  
Email: tanchak@umiacs.umd.edu

Noseong Park  
Dept. of Computer Science  
University of Maryland  
College Park, MD 20742  
Email: npark@cs.umd.edu

V.S. Subrahmanian  
Dept. of Computer Science  
University of Maryland  
College Park, MD 20742  
Email: vs@umiacs.umd.edu

Abstract—Given a set $AL$ of community detection algorithms and a graph $G$ as inputs, we propose two ensemble methods EnDisCo and MeDOC that (respectively) identify disjoint and overlapping communities in $G$. EnDisCo transforms a graph into a latent feature space by leveraging multiple base solutions and discovers disjoint community structure. MeDOC groups similar base communities into a meta-community and detects both disjoint and overlapping community structures. Experiments are conducted at different scales on both synthetically generated networks as well as on several real-world networks for which the underlying ground-truth community structure is available. Our extensive experiments show that both algorithms outperform state-of-the-art non-ensemble algorithms by a significant margin. Moreover, we compare EnDisCo and MeDOC with a recent ensemble method for disjoint community detection and show that our approaches achieve superior performance. To the best of our knowledge, MeDOC is the first ensemble approach for overlapping community detection.

I. INTRODUCTION

Community detection (CD) has found applications in social, biological, business, and other kinds of networks. However, CD algorithms suffer from various flaws – (i) Most existing CD algorithms are heavily dependent on vertex ordering [11], yielding completely different community structures when the same network is processed in a different order. For example, Figure 1(a) shows dissimilar community structures after running InfoMap [30] on 100 different vertex orderings of the Football network [7]. (ii) Most optimization algorithms may produce multiple community structures with the same “optimal” value of the objective function. For instance, in Figure 1(b), assigning vertex $x$ to either $A$ or $B$ results in the same modularity score [24]. (iii) Different CD algorithms detect communities in different ways, e.g., as dense groups internally [25], or groups with sparse connections externally [10]. It is therefore natural to think of an ensemble approach in which the strengths of different CD algorithms may help overcome the weaknesses of any specific CD algorithm. Some preliminary attempts have been made by [8], [20].

Contributions. In this paper, we design two ensemble CD algorithms. The EnDisCo algorithm runs multiple “base CD algorithms using a variety of vertex orderings to derive a first set of communities. We then consider the memberships of vertices obtained from base CD algorithms as features and derive a latent network using pair-wise similarity of vertices. The final disjoint community structure is obtained by running any CD algorithm again on the latent network. The MeDOC algorithm leverages the fact that many communities returned by base algorithms are redundant and can therefore be grouped into “meta-communities” to avoid unnecessary computation. We use meta-communities to build an association matrix, where each entry indicates the probability of a vertex belonging to a meta-community. Finally, we obtain both disjoint and overlapping community structures via post-processing on the association matrix. To the best of our knowledge, we are the first to propose (i) an ensemble framework for overlapping community detection, and (ii) an overlapping CD algorithm that leverages disjoint community information.

We run experiments to identify the best parameter settings for EnDisCo and MeDOC. Experiments on both synthetic and real-world networks show that our algorithms outperform both the state-of-the-art non-ensemble based methods [2], [30], [34] and a recently proposed ensemble approach [20] by a significant margin. We also show that our ensemble approaches reduce the effect of vertex ordering.

Note: We use the term “community structure” to indicate the result (set of “communities”) returned by an algorithm. Each
community is a set of vertices.

II. RELATED WORK

There has been a great deal of work on clustering data using ensemble approaches (see [33] for a review). However, when it comes to clustering vertices in networks, ensemble approaches have been relatively scarce. Dahlin and Svenson [8] were the first to propose an instance-based ensemble CD algorithm for networks which fuses different community structures into a final representation. A few methods addressed the utility of merging several community structures [29]. A new ensemble scheme called CGGC was proposed to maximize modularity [26]. Kanawati proposed Y ASCA, an ensemble approach to compute the final disjoint community structure. Our work differs from this approach in at least two ways: (i) they either take multiple algorithms or run a particular algorithm multiple times for generating an ensemble matrix, whereas we consider both options; (ii) we are the first to show how aggregating multiple disjoint base communities can lead to discovering both disjoint and overlapping community structures simultaneously. We show experimentally that EnDisCo beats consensus clustering.

III. EnDisCo: Ensemble-based Disjoint Community Detection

EnDisCo (Ensemble-based Disjoint Community Detection) starts by using different CD algorithms to identify different community structures. Second, an “involvement” function is used to measure the extent to which a vertex is involved with a given community, which in turn sets the posterior probabilities of each vertex belonging to different communities. Third, EnDisCo transforms the network into a feature space. Fourth, an ensemble matrix is constructed by measuring the pair-wise similarity of vertices. Finally, we apply any standard CD algorithm on the ensemble matrix and discover the final disjoint community structure.

A. Algorithmic Description

EnDisCo follows three fundamental steps (a pseudo-code is shown in Algorithm 1):

(i) Generating base partitions. Given a network $G=(V,E)$ and a set $\mathcal{A} = \{A_m\}_{m=1}^M$ of $M$ different base CD algorithms, EnDisCo runs each algorithm $A_m$ on $K$ different vertex orderings (randomly selected) of $G$. This generates a set of $K$ different community structures denoted $\Gamma_m = \{C^n_m\}_{k=1}^K$, where each community structure $C^n_m = \{C^{1k}_m, \ldots, C^{ak}_m\}$ constitutes a specific partitioning of vertices in $G$, and each $C^n_m$ might be of different size (Step 3).

(ii) Constructing ensemble matrix. Given a $\Gamma_m$, we then compute the extent of $v$’s involvement in each community $C$ in $C^n_m$ via an “involvement” function $\text{INV}(v,C)$ (Step 12). Possible definitions of $\text{INV}$ are given in Section III-B. For each vertex $v$, we construct a feature vector $F(v)$ whose elements indicate the distance of $v$ (measured by $1 - \text{INV}$) from each community obtained from different runs of the base algorithms (Step 13). The size of $F(v)$ is same as the number of communities $\text{Clu}$ 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 $\Gamma$.

Algorithm 1: EnDisCo: Ensemble-based Disjoint Community Detection

| Data: | Graph $G(V,E)$; |
|-------|-----------------|
| Base algorithms $\mathcal{A} = \{A_m\}_{m=1}^M$; |
| $K$: Number of iterations; |
| $\text{INV}(\cdot)$: Involvement function; |
| $\text{SIM}(\cdot)$: Similarity function between two vectors; |
| $\text{RA}_\text{algo}$: Algorithm for re-clustering |

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

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

2. Generating base partitions

3. Run $A_m$ on $G$ for $K$ different vertex orderings and obtain $K$ community structures, denoted by the set $\Gamma_m$; each community structure $C^n_m \in \Gamma_m$ is of different size and indicated by $C^n_m = \{C^{1k}_m, \ldots, C^{ak}_m\}$.

4. $\Gamma = \Gamma \cup \Gamma_m$;

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. $\text{Clu} = 0$; // Total no of communities

9. Constructing ensemble matrix

10. for each $\Gamma_m$ in $\Gamma$ do

11. for each $C^n_m \in \Gamma_m$ do

12. Compute $d_C^2 = 1 - \text{INV}(v,C)$;

13. $F(v) = F(v) \cup d_C^2$;

14. if $d_C^2 \geq D_v$ then

15. $D_v = d_C^2$;

16. $\text{Clu} = \text{Clu} + 1$;

17. $P(v) = \phi$;

18. for each $F_i(v) \in F(v)$ do

19. Compute $P(C_i|v) = \frac{D_v - F_i(v) + 1}{\text{Clu} + D_v + \text{Clu} - \sum_{k=1}^K F_k(v)}$;

20. $P(v) = P(v) \cup P(C_i|v)$;

21. Build an ensemble matrix $M_{|V| \times |V|}$, where $\forall u,v \in V, M(u,v) = \text{SIM}(F(u),F(v))$;

22. Re-clustering the vertices from $M$;

23. $\text{Return DC}$;
(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:

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

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 $\sum_{k=1}^{C_{lu}} P(C_i|v) = 1$. Add-one smoothing in the numerator allows a non-zero probability to be assigned to all $C_i$s, especially for $C_k$ such that $k = \arg\max_i F_k(v)$.

The set of posterior probabilities of $v$ is:

$$\{P(C_k|v)\}_{k=1}^{C_{lu}}$$

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 $SIM$) whose possible definitions are given in Section III-B between the feature vectors of $u$ and $v$ (Step 20). The ensemble matrix ensures that the more communities a pair of vertices share the more likely they are connected in the network [34].

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

B. Parameter Selection

- **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., $RCC(v,C) = \sum_{u \in C} \frac{1}{dist(v,u)}$; (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., $IDC(v,C) = \frac{1}{dist(v,u_c)}$.

- **Similarity Function ($SIM$):** We consider cosine similarity ($COS$) and Chebyshev distance ($CHE$) (essentially, $1-CHE$) to measure the similarity between two vectors.

- **Algorithm for Re-clustering ($RAlgo$):** We consider each base algorithm as the one to re-cluster the vertices from the ensemble matrix. The idea is to show that a non-ensemble CD algorithm can perform even better when considering the ensemble matrix of network $G$ than the adjacency matrix of $G$. However, one can use any CD algorithm in this step to detect the community structure. We will show the effect of different algorithms used in this step in Section IV-D2.

- **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 2(c) and 2(f)), and therefore we set $K = 0.2|V|$ in our experiments.

C. 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 19 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 and Overlapping Community Detection

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

**Result:** Disjoint ($\mathcal{D}$) and overlapping ($\mathcal{O}$) community structures

```
// Constructing multipartite network
for all $A_m$ in $\mathcal{A}$ do
  Run $A_m$ on $G$ for $K$ different vertex orderings and obtain $K$ community structures, denoted by the set $\Gamma_m = \{C_{km}\}_{k=1}^K$; each community structure $C_{km} \in \Gamma_m$ may be of different size and is denoted by $C_{km} = \{C_{m1}, ..., C_{mk}\}$;
  Construct a $P$-partite graph $GP$ (where $P = M.K$) consisting of $M.K$ partitions, each corresponding to each community structure obtained in Step 2; vertices in partition $m^k$ are communities in $C_{km}$ and edges are drawn between two pair-wise vertices (communities) $C_{km}$ and $C_{kn}$ with the edge weight $W(C_{km}, C_{kn})$;
  Run $RAlgo$ to re-cluster vertices in $GP$ and discover a meta-community structure, $C_{GP} = \{C_{GP}\}_{l=1}^L$;
// Constructing an association matrix
Construct an association matrix $\mathcal{A}_{\mathcal{V}\times\mathcal{L}}$, where $\mathcal{A}(v,l) = \mathcal{F}(v, C_{GP})$, indicating the association of vertex $v$ to a meta-community $C_{GP}$;
// Discovering final community structure
Each row in $\mathcal{A}$ indicates the membership probabilities of the corresponding vertex in $L$ meta-communities;
To get $\mathcal{D}$, we assign a vertex $v$ to community $C^* = \arg\max C \mathcal{A}(v,C)$;
To get $\mathcal{O}$, we assign a vertex $v$ to a set of communities $C^*_v$ so that $\forall C \in C^*_v: \mathcal{A}(v,C) \geq \tau$;
return $\mathcal{D}$, $\mathcal{O}$
```

IV. MeDOC: META-CLUSTERING APPROACH

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

### A. Algorithmic Description

MeDOC has the following four basic steps (pseudo-code is in Algorithm 2):

(i) **Constructing multipartite network.** MeDOC takes a CD algorithm $\mathcal{A}$ which takes $M$ different vertex orderings of $G$. For each ordering $k$, $\mathcal{A}$ produces a community structure $\mathcal{C}_k = \{C_{ik}, i \in [m]\}$ of varying size (Step 2). After running on $K$ vertex orderings, each algorithm $\mathcal{A}_l$ produces $K$ different community structures $\gamma_m = \{C_{ik}, i \in [m]\}$. 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) $\mathcal{G}$ as follows: the vertices are members of $\bigcup_m \mathcal{C}_{m}$, i.e., a community present in a community structure obtained from any of the base algorithms in $\mathcal{A}$ and any vertex ordering, is a vertex of $\mathcal{G}$. We draw an edge from a community $C_{ik}$ to another community $C_{jk}$ and associate a weight $W(C_{ik}, C_{jk})$ (Step 3). Possible definitions of $W$ will be given later in Section IV-B. Since each $\mathcal{C}_{m}$ is disjoint, the vertices in each partition are never connected.

(ii) **Re-clustering the multipartite network.** Here we run any standard CD algorithm $\mathcal{A}L$ on the multipartite network $\mathcal{G}$ and obtain a community structure containing (say) $L$ communities $\mathcal{C}_G = \{C_{G}, i \in [L]\}$. Note that in this step, we indeed cluster the communities obtained earlier in Step 2; therefore each such community $C_{G}$ 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 $C_{G}$ by using a function $\mathcal{F}$ and construct an association matrix $\mathcal{A}$ of size $|V| \times L$, where each entry $h_{\mathcal{A}}(v, l) = \mathcal{F}(v, C_{G})$ (Step 5). Possible definitions of $\mathcal{F}$ will be given later in Section IV-B.

(iv) **Discovering final community structure.** Final vertex-to-community assignment is performed by processing $\mathcal{A}$. The entries in each row of $\mathcal{A}$ 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 $\mathcal{A}$, i.e., $l^* = \arg\max_l h_{\mathcal{A}}(v, l)$. Tie-breaking is handled by assigning the vertex to the community to which most of its direct neighbors belong. Note that every meta-community can be guaranteed to contain at least one vertex, that in turn can not assure $L$ 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 $\delta$. Possible ways to specify threshold will be specified later in Section IV-B.

### B. Parameter Selection

- **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 $(AP) = \frac{1}{2}(\frac{|C_i \cap C_j|}{|C_i|} + \frac{|C_i \cap C_j|}{|C_j|})$.

- **Association Function ($\mathcal{F}$):** Given a meta-community $C$ consisting of (say) $\gamma$ communities, the association of $v$ with $C$ can be calculated as $\mathcal{F}(v, C) = \sum_{i=1}^{\gamma} \delta(v, C_i)$, where $\delta$ returns 1 if $v$ is a part of $C_i$, 0 otherwise. For example, if $C = \{\{1, 2, 3, 5\}, \{1, 2, 7\}, \{2, 7, 8\}\}$, then $\mathcal{F}(1, C) = \frac{3}{4}$. Alternatively, a weighted association measure may assign a score to $v$ w.r.t. $C$ based on the co-occurrence of the other $\gamma - 1$ community members with $v$, i.e., $\mathcal{F}_w(v, C) = \sum_{i=1}^{\gamma} c_i(v, C_i)$, where $c_i(v, C_i) = \frac{1}{\sum_{j \neq i} c_j(v, C_j)}$.

In our earlier example, $\mathcal{F}_w(1, C) = \frac{1}{\frac{1}{4} + \frac{1}{4}} = \frac{3}{5}$.

- **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 $\mathcal{A}$. We then measure the average similarity of vertices in $C$ as follows: $AS(C) = \frac{\sum_{u \in C, v \in C} \cos(F(u), F(v))}{|E_C|}$, where $E_C$ is the set of edges completely internal to $C$, $E_{uv}$ is an edge $(u, v)$, and $\cos$ is cosine similarity. The probability that two vertices are connected in $C$ is then defined as:

$$P(C) = \frac{\sum_{u \in C, v \in C} \cos(F(u), F(v))}{|E_C|}$$

For a vertex $v$, if $P(C \cup \{v\}) > P(C)$, we further assign $v$ to $C$, in addition to its current community. 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 3 and 4).

Other input parameters $\mathcal{A}L$ and $K$ remain same as discussed in Section III-B.

### C. Complexity Analysis

The most expensive step of $\mathcal{A}L$ 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 $O(\bar{a} M K^2)$. However, in practice the network is extremely sparse and leads to $O(\bar{a} M K)$ edges (because in sparse graphs $O(|V|) \sim O(|E|)$). Further, constructing the association matrix would take $O(NL)$ iterations (where $L \ll N$).

### V. RESULTS OF DISJOINT COMMUNITY DETECTION

#### A. Datasets

We use the LFR benchmark model [19] 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 $k$, maximum degree $k_{\text{max}}$, minimum (maximum) community size $c_{\text{min}}$ ($c_{\text{max}}$), average percentage $O_n$ of overlapping vertices and the average number $O_m$ of communities to which a
### TABLE I

**Properties of the real-world networks.** $N$: number of vertices, $E$: number of edges, $C$: number of communities, $p$: average edge-density per community, $S$: average size of a community, $O_n$: average number of community memberships per vertex.

| Networks          | Vertex type | Edge type | Community type | $N$  | $E$   | $C$  | $p$  | $S$  | $O_n$  | Reference |
|-------------------|-------------|-----------|----------------|------|------|------|------|------|---------|-----------|
| University        | Faculty     | School    | School         | 81   | 81   | 1    | 0.21 | 4.76 | 5.64    |            |
| Football          | Team        | Games     | Group-division | 115  | 613  | 12   | 0.64 | 9.66 | 1       | [1]       |
| Railway           | Stations    | Connections | Province      | 301  | 1,224| 21   | 0.24 | 12.26| 1       | [2]       |
| Coauthorship     | Researcher  | Collaborations | Research area | 103,677 | 352,183 | 24   | 0.14 | 376,252| 1         | [3]       |

| Networks          | Vertex type | Edge type | Community type | $N$  | $E$   | $C$  | $p$  | $S$  | $O_n$  | Reference |
|-------------------|-------------|-----------|----------------|------|------|------|------|------|---------|-----------|
| Senate            | Senate      | Similar voting pattern | Congress | 1,884 | 10,662 | 110  | 0.45 | 61.97 | 4.76    |            |
| Flickr            | User        | Friendship | Joined group   | 80,513 | 5,899,882 | 171  | 0.046 | 470.83| 18.96   |            |
| Coauthorship     | Researcher  | Collaborations | Publication venues | 391,526 | 873,775 | 849.3 | 0.231 | 393.18| 10.45   | [4]       |
| LiveJournal       | User        | Friendship | User-defined group | 3,997,962 | 34,681,189 | 310,092 | 0.536 | 40.02 | 3.09    | [5]       |
| Okkat             | User        | Friendship | User-defined group | 1,072,441 | 117,185,083 | 6,288,363 | 0.245 | 34.86 | 90.93   | [4]       |

### B. Baseline Algorithms

We compare EnDisCo and MeDOC with the following algorithms: (i) **Modularity-based** FastGreedy (FstGrdy) [25], Louvain (Louvain) [2] and CNM [7]; (ii) **Vertex similarity-based** WalkTrap (WalkTrap) [23]; (iii) **Compression-based** InfoMap (InfoMap) [30]; (iv) **Diffusion-based** Label Propagation (LabelPr) [29]. These algorithms are also used as base algorithms in AC in our ensemble approaches. We further compare our methods with Consensus Clustering (ConsCl) [20], a recently-proposed ensemble-based framework for disjoint community detection.

### C. Evaluation Metrics

As we know the ground-truth community structure, we measure performance of competing CD algorithms using the standard Normalized Mutual Information (NMI) [9] and Adjusted Rand Index (ARI) [14].

### D. Experimental Results

We first run experiments to identify the best parameters for EnDisCo and MeDOC and then present the comparative analysis among the competing algorithms.

1) **Dependence on the Parameters:** We consider the LFR networks and vary $\mu$. Figure 2(a) shows that the accuracy of EnDisCo is similar for both the involvement functions, while Figure 2(b) shows cosine similarity fully dominating Chebyshev distance. Figure 2(d) shows that Jaccard coefficient performs significantly better than average precision when MeDOC is considered, while Figure 2(e) shows that the weighted association function seems to dominate 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 2(c) and 2(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 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: EnDisCo: $K = 0.2|V|$, $RCC$, $COS$; MeDOC: $K = 0.2|V|$, $JC$, $F\beta$.

2) **Impact of Base CD Algorithms on EnDisCo and MeDOC:** In order to assess the impact of each base algorithm in our ensemble, we measure the accuracy of EnDisCo and MeDOC when that base algorithm is removed from the ensemble — Table III shows that for LFR networks InfoMap has the biggest impact on accuracy according to both the evaluation measures (NMI and ARI) for both EnDisCo and MeDOC (results are same for real networks 1)).

As the final step in both EnDisCo and MeDOC is to run a CD algorithm for re-clustering, we also conduct experiments (Table IV for LFR networks and Appendix I for real networks) to identify the best one. Again, InfoMap proves to be the best.

3) **Comparative Evaluation:** Table II(A) reports the performance of our approaches on all networks using different algorithms in the final step of EnDisCo and MeDOC. The
numbers denote relative performance improvement of our algorithms (E:EnDisCo M:MeDOC) w.r.t. a given algorithm when that algorithm is used in the final step. For instance, the first entry in the last row (5.06) means that for LFR ($\mu = 0.1$) network, the accuracy of EnDisCo (when LabelPr is used for re-clustering in its final step) averaged over NMI and ARI (0.83) is 5.06% higher than that of LabelPr (0.79). The actual values are reported in Appendix [1]. The point to take away from this table is that irrespective of which classical CD algorithm we compare against, EnDisCo and MeDOC always improve the quality of communities found. Moreover, we 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. Further, Table III(B) shows the average improvement of EnDisCo and MeDOC when compared against Consensus Clustering (ConsCl). We see that for disjoint networks, both EnDisCo and MeDOC beat ConsCl with MeDOC emerging in top place.

VI. RESULTS OF OVERLAPPING COMMUNITY DETECTION

A. Datasets

We again use the LFR benchmark to generate synthetic networks with overlapping community structure with the following default parameter settings as mentioned in [18], [12]: $n = 10000$, $k = 50$, $k_{\text{max}} = 150$, $\mu = 0.3$, $O_m = 20$, $O_m = 20$, $O_m = 100$, $c_{\text{min}} = 20$. 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.

We also run experiments with six real-world datasets mentioned in Table II(b) (see details in Appendix [1]).

B. Baseline Algorithms

We compare MeDOC with the following state-of-the-art overlapping community detection algorithms: (i) Local expansion: OSLOM [18], EAGLE [13]; (ii) Agent-based dynamical algorithms: COPRA [12], SLPA [32]; (iii) Detection using mixture model: MOSES [21], BIGCLAM [34].

C. Evaluation Metrics

We use the following evaluation metrics to compare the results with the ground-truth community structure: (a) Overlapping Normalized Mutual Information (ONMI) [22], (b) Omega ($\Omega$) Index [34] (details in Appendix [1]).

D. Experimental Results

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 3 shows the results (on LFR networks) by varying $\mu$, $O_m$ and $O_n$. We observe that Jaccard coefficient as matching function and weighted association measure are better than their alternative. The choice of $K$ is the same as shown in Figure 2(f) — accuracy levels off at $K = 0.2|V|$, and therefore we skip this result in the interest of space. 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 V-B. Figures 3(g)-(3(i) show irrespective of any network parameter selection, our choice of selecting threshold always outperforms others. As shown in Table IV InfoMap seems to be the best choice
TABLE V

ACROSS THE LEADING ALGORITHMS IN DETECTING THE OVERLAPPING COMMUNITY STRUCTURE FROM BOTH SYNTHETIC AND REAL-WORLD NETWORKS. ALL THE DISJOINT ALGORITHMS ARE USED TO CREATE THE MULTIPARTITION NETWORK AND MeDOC IS RUN WITH ITS DEFAULT PARAMETER SETTING.

| Algorithm | LFR (μ = 0.1) | Synthetic Networks | LFR (μ = 0.8) | Senate | Flickr | Real-world Networks |
|-----------|--------------|--------------------|--------------|--------|-------|-------------------|
|           | CNM | UNM | LFR | CNM | UNM | LFR | CNM | UNM | LFR | CNM | UNM | LFR | CNM | UNM | LFR |
| CNM       | 0.85 | 0.90 | 0.83 | 0.84 | 0.76 | 0.75 | 0.70 | 0.74 | 0.72 | 0.74 | 0.72 | 0.74 | 0.74 | 0.74 | 0.76 | 0.76 |
| EAGLE     | 0.81 | 0.83 | 0.75 | 0.76 | 0.70 | 0.74 | 0.72 | 0.74 | 0.74 | 0.76 | 0.72 | 0.74 | 0.74 | 0.74 | 0.76 | 0.76 |
| InfoMap   | 0.80 | 0.81 | 0.76 | 0.74 | 0.72 | 0.74 | 0.74 | 0.76 | 0.74 | 0.76 | 0.74 | 0.76 | 0.74 | 0.74 | 0.76 | 0.76 |
| COPRA     | 0.54 | 0.66 | 0.78 | 0.77 | 0.76 | 0.77 | 0.77 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 |
| RIMOS      | 0.85 | 0.86 | 0.80 | 0.81 | 0.75 | 0.78 | 0.77 | 0.79 | 0.76 | 0.79 | 0.76 | 0.79 | 0.76 | 0.79 | 0.79 | 0.79 |
| MeDOC     | 0.86 | 0.85 | 0.83 | 0.83 | 0.77 | 0.79 | 0.76 | 0.79 | 0.76 | 0.79 | 0.76 | 0.79 | 0.76 | 0.79 | 0.79 | 0.79 |

Fig. 3. Dependencies of MeDOC on different algorithmic parameters. The results are reported on default overlapping LFR networks by varying three parameters μ, ONMI, and ONM. For thresholding, we choose top 5% and 10% highly probable communities for each vertex and then aggregate the output with our threshold selection method, where 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 the re-clustering algorithm. Therefore, in the rest of the experiments, we run MeDOC with K = 0.2|V|, JC, Fw, InfoMap and τ (selected by our method).

2) Impact of Base Algorithms for Overlapping CD: The impact of the base algorithms on MeDOC’s performance is similar to what we saw in the disjoint CD case. The results in Table III show that accuracy decreases most when we drop InfoMap from the base algorithm, followed by Louvain and CNM (see more results in Appendix [1]).

3) Comparative Evaluation: We ran 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 IV shows the performance of the competing algorithms in terms of ONMI and Ω index. In all cases, MeDOC is a clear winner, winning by significant margins. The absolute average of ONMI (Ω) for MeDOC over all networks is 0.83 (0.86), which is 3.58% (4.39%) higher than BigClam, 5.90% (7.49%) higher than MOSES, 8.31% (9.19%) higher than SLPA, 10.67% (10.95%) higher than COPRA, 13.89% (12.95%) higher than EAGLE, and 14.68% (15.21%) higher than OSLOM. Another interesting observation is that the performance improvement seems to be prominent with the deterioration of community quality. For instance, the improvement of MeDOC w.r.t. 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 V-D3 for disjoint communities.

VII. RUNTIME ANALYSIS

Since ensemble approaches require the running all baseline algorithms (which may be parallelized), one cannot expect ensemble methods to be faster than baseline 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 of edges of LFR by changing μ from 0.1 to 0.3. Figure IV shows that our algorithms are much faster than consensus clustering. We further report the results of 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.

VIII. DEGENERACY OF SOLUTIONS

CD algorithms suffer from the problem of “degeneracy of solutions” which states that an optimization algorithm can 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. Figure I showed how InfoMap produces many outputs for different vertex orderings of Football network. We test this by considering the default LFR network and one real-world network (Appendix I shows results on more real world networks) and run the 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 Figure V show the variation of the solutions for EnDisCo, MeDOC and the best baseline algorithm in both disjoint and overlapping community detections. We observe that the median similarity is high with EnDisCo and MeDOC and the variation is comparatively small. These results suggest that
our algorithms provide more robust results than past work and alleviate the problem of degeneracy of solutions.

IX. CONCLUSION

In this paper, we proposed two general frameworks for ensemble community detection. EnDisCo identifies disjoint community structures, while MeDOC detects both disjoint and overlapping community structures. We tested both algorithms on both synthetic data using the LFR benchmark and with several real-world datasets that have associated ground-truth community structure. We show that both EnDisCo and MeDOC are more accurate than existing CD algorithms, though of course, EnDisCo and MeDOC leverage them. We further show that for disjoint CD problems, EnDisCo and MeDOC both beat a well-known existing disjoint ensemble method called consensus clustering [20] – both in terms of accuracy (measured via both Normalized Mutual Information and Adjusted Rand Index) and run-time. To our knowledge, MeDOC is the first ensemble algorithm for overlapping community detection that we have seen in the literature. In future, we would like to develop theoretical explanation to justify the superiority of ensemble approaches compared to the discrete models. Other future direction could be to make the ensemble frameworks parallelized. We will apply the proposed methods to identify communities in specific datasets, such as malware traces, protein interaction networks etc.

ACKNOWLEDGMENT

Parts of this work were funded by ARO Grants W911NF-16-1-0342, W911NF1110344, W911NF1410358, by ONR Grant N00014-13-1-0703, and Maryland Procurement Office under Contract No. H98230-14-C-0137.

REFERENCES

[1] http://www.umiacs.umd.edu/~tanchak/Appendix.pdf
[2] V. D. Blondel, J-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” JSTAP, p. P10008, 2008.
[3] T. Chakraborty, “Leveraging disjoint communities for detecting overlapping community structure,” Journal of Statistical Mechanics: Theory and Experiment, vol. 2015, no. 5, p. 05017, 2015.
[4] T. Chakraborty, S. Sikdar, N. Ganguly, and A. Mukherjee, “Citation interactions among computer science fields: a quantitative route to the rise and fall of scientific research,” Social Netw. Analys. Mining, vol. 4, no. 1, p. 187, 2014.
[5] T. Chakraborty, S. Sikdar, V. Tammana, N. Ganguly, and A. Mukherjee, “Computer science fields as ground-truth communities: their impact, rise and fall,” in ASONAM, Niagara Falls, Canada, 2013, pp. 426–433.
[6] T. Chakraborty, S. Srinivasan, N. Ganguly, A. Mukherjee, and S. Bhownick, “On the permanence of vertices in network communities,” in SICKDD, New York, USA, 2014, pp. 1396–1405.
[7] A. Clauset, M. E. J. Newman, and C. Moore, “Finding community structure in very large networks,” Phys. Rev. E., vol. 70, no. 6, p. 066111, 2004.
[8] J. Dahlin and P. Svensson, “Ensemble approaches for improving community detection methods,” CoRR, vol. abs/1309.0242, 2013.
[9] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, “Comparing community structure identification,” JSTAP, vol. 9, p. P09008, 2005.
[10] S. Fortunato, “Community detection in graphs,” Physics Reports, vol. 486, no. 3-5, pp. 75 – 174, 2010.
[11] B. Good, Y. D. Montjoye, and A. Clauset, “Performance of modularity maximization in practical contexts,” Phy. Rev. E., vol. 81, no. 4, p. 046106, 2010.
[12] S. Gregory, “Finding overlapping communities in networks by label propagation,” New J. Phys., vol. 12, no. 10, p. 103018, 2010.
[13] K. C. H. Shen, X. Cheng and M. B. Hu, “Detect overlapping and hierarchical community structure in networks,” Physica A, vol. 388, no. 8, pp. 1706–1712, 2009.
[14] L. Hubert and P. Arabie, “Comparing partitions,” Journal of classification, vol. 2, no. 1, pp. 193–218, 1985.
[15] L. G. S. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, and M. W. Mahoney, “Think locally, act locally: Detection of small, medium-sized, and large communities in large networks,” Phy. Rev. E., vol. 91, p. 012821, 2015.
[16] R. Kanawati, “Yasca: An ensemble-based approach for community detection in complex networks,” in COCOON. Cham: Springer, 2014, pp. 567–566.
[17] ——, “Ensemble selection for community detection in complex networks,” in SCSM. CA, USA: Springer, 2015, pp. 138–147.
[18] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, “Finding statistically significant communities in networks,” PLoS ONE, vol. 6, no. 4, p. e18961, 2011.
[19] A. Lancichinetti and S. Fortunato, “Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities,” Phy. Rev. E., vol. 80, p. 016118, 2009.
[20] ——, “Consensus clustering in complex networks,” Nature Scientific Reports, vol. 2, 2012.
[21] A. McAldaid and N. Hurley, “Detecting highly overlapping communities with model-based overlapping seed expansion,” in ASONAM, Washington, DC, USA, 2010, pp. 112–119.
[22] A. F. McAldaid, D. Greene, and N. J. Hurley, “Normalized mutual information to evaluate overlapping community finding algorithms,” CoRR, vol. abs/1110.2515, 2011.
[23] T. Nepusz, A. Petróczki, L. Négysessy, and F. Bazzó, “Fuzzy communities and the concept of bridgeness in complex networks,” Phy. Rev. E., vol. 77, p. 016107, 2008.
[24] M. E. Newman, “Modularity and community structure in networks,” PNAS, vol. 103, no. 23, pp. 8577–8582, 2006.
[25] M. E. J. Newman, “Fast algorithm for detecting community structure in networks,” Phy. Rev. E., vol. 69, no. 6, p. 066133, Jun. 2004.
[26] M. Ovelgoënne and A. Geyer-Schulz, “An ensemble learning strategy for graph clustering,” in Graph Partitioning and Graph Clustering, ser. Contemporary Mathematics, vol. 588, pp. 187–206.
[27] G. Palla, I. Farkas, P. Pollner, I. Derényi, and T. Vicsek, “Fundamental statistical features and self-similar properties of tagged networks,” New J. Phys., vol. 10, no. 12, p. 123026, 2008.
[28] P. Pons and M. Latapy, “Computing communities in large networks using random walks,” J. Graph Algorithms Appl., vol. 10, no. 2, pp. 191–218, 2006.
[29] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” Phy. Rev. E., vol. 76, no. 3, 2007.
[30] M. Rosvall and C. T. Bergstrom, “Maps of random walks on complex networks reveal community structure,” PNAS, vol. 105, no. 4, pp. 1118–1123, 2008.
[31] X. Wang, L. Tang, H. Liu, and L. Wang, “Learning with multi-resolution overlapping communities,” Knowl. Inf. Syst., vol. 36, no. 2, pp. 517–535, 2013.
[32] J. Xie and B. K. Szymanski, “Towards linear time overlapping community detection in social networks,” in PAKDD, Malaysia, 2012, pp. 25–36.
[33] R. Xu and D. Wunsch, II, “Survey of clustering algorithms,” Trans. Neural Netw., vol. 16, no. 3, pp. 645–678, May 2005.
[34] J. Yang and J. Leskovec, “Overlapping community detection at scale: A nonnegative matrix factorization approach,” in WSDM. New York, USA: ACM, 2013, pp. 587–596.