Hacd: Hierarchical Agglomerative Community Detection In Social Networks

Conference Paper · October 2019

DOI: 10.1109/MLSP.2019.8918734

4 authors, including:

Ekta Gujral
Walmart
29 PUBLICATIONS **212** CITATIONS
SEE PROFILE

Evangelos E. Papalexakis
Carnegie Mellon University
182 PUBLICATIONS **5,151** CITATIONS
SEE PROFILE

Anup Rao
Yale University
67 PUBLICATIONS **1,356** CITATIONS
SEE PROFILE

HACD: Hierarchical Agglomerative Community Detection in Social Networks

Ekta Gujral[§], Georgios Theocharous[♭], Anup Rao[♭],

[§]University of California Riverside, [♭] Adobe Research
egujr001@ucr.edu,(theochar,anuprao)@adobe.com

ABSTRACT

Communities (also referred as clusters) are essential building blocks of all networks. Hierarchical clustering methods are common graph-based approaches for graph clustering. Traditional hierarchical clustering algorithms proceed in a bottom-up or top-down fashion to encode global information in the graph and cluster according to the global modularity of the graph. It measures the strength of the division of a network into clusters. Mathematically, it is the fraction of the edges that fall within the given cluster minus the expected fraction if edges were distributed at random. In this paper, we propose an efficient Hierarchical Agglomerative Community Detection (HACD) algorithm, that combines the local information in a graph with membership propagation to solve the problem, achieving *10-25%* quality improvement over all baselines. The first contribution of this paper is to present fundamental limitations of the general modularity optimization-based approach. We show that based only on modularity information, the method does not provide high-quality clusters. Furthermore, even with modularity optimization, we experimentally show that the final level partitioning of such methods cannot successfully cluster data that contain highly mixed structures at different levels and densities. Based on these findings, the second contribution of this paper is a novel method to propagate knowledge throughout the graph, to split or merge the communities in order to evaluate the consistency of individual clusters. Our approach is bottom-up graph-based clustering, is scale-free, and can determine clusters at all scales. We extensively evaluate HACD's performance in comparison to state-of-the-art approaches across six real world and seven synthetic benchmark datasets, and demonstrate that HACD, through combining graph's local information and membership propagation, outperforms the baselines in terms of finding well-integrated communities.

1 INTRODUCTION

Graph partitioning methods have explored various structures and properties of a wide range of systems, including social networks[28], world wide web[28], biological networks [21], and many more. Often, network analysis focuses on detecting natural division of graphs into communities or clusters. The two broad classes of methods that have been used in the literature to solve the community detection problem are divisive methods and agglomerative methods. The divisive methods partition the graph or network into multiple subgraphs by removing edges in each iteration, whereas agglomerative methods start placing nodes in different communities and iteratively cluster them based on an objective function or similarity criteria. Both approaches are famous and successful in clustering the nodes, however, they have limitations. For example, divisive method like minimum cut[24], which tries to minimize the sum of weights of removed edges, have the disadvantage of often dividing the network unevenly. For example in Figure 1, simply minimizing normalized cuts produces a "bad" partition, while additionally taking into consideration the structure produces a "good "partition[42]. To tackle this problem, various modified cost functions that normalize the weight on edges are proposed [9, 29], but they do not have the capability to preserve the community structure as they divide the network into a pre-defined fixed number of communities of the same size.

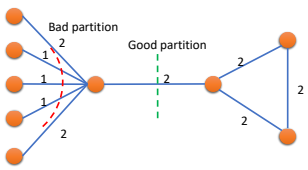

Figure 1: Simple synthetic weighted graph. Clustering algorithms may produce a bad partition, which is indicated by the red dashed line. A better partitioning should also consider the similarity of nodes in a group.

Community detection is a graph clustering problem. There is no global or single accepted definition of community within any network, for example, in brain imaging, communities may consist of a brain region whose functions are highly co-related and in biological networks, communities are a group of proteins or RNA associated with specific cellular functions, but a widely accepted and popular definition is that *a community is a collection of nodes that have strong inter-connection within the community than what we would expect to occur at random.* This leads to the methods based on Modularity [10]. The Newman-Girvan modularity [10] is one of the most common clustering measures used in the literature and was originally proposed from the clustering perspective discussed below. There are different methods for calculating modularity. The most common way of the concept is the randomization of the edges in such a way that it preserves the degree of each vertex. Let us consider a graph G with $\mathcal{V}$ nodes and $\mathcal{E}$ edges such that the graph can be partitioned into two communities using a membership variable s. If a node v belongs to community 1, $s_v = 1$, or v belongs to community 2, $s_v = -1$. Let A is adjacency matrix where $A_{vu} = 0$ means there's no edge (no interaction) between nodes v and w and $A_{vu} = 1$ means there is an edge between the two. Modularity Q is then defined as the fraction of edges that fall within group 1 or 2, minus the expected number of edges within groups 1 and 2 for a random graph with the same node degree distribution or degree sequence as the given network. The expected number of edges shall be computed using the concept of Configuration Models.

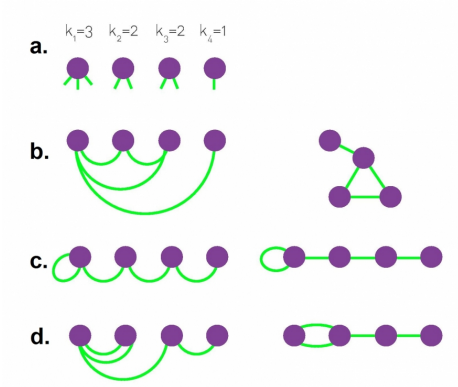

Figure 2: Degree sequence and different network realizations in the configuration model. Note: The degree sequence of an graph is the non-increasing sequence of its vertex degrees.

In network science, the configuration model is a method for generating random networks from given degree sequence as shown in Figure 2[1]. It is widely used as a reference model for real-life social networks, because it allows the user to incorporate arbitrary degree distributions. It is two step algorithm, (i) Take a degree sequence, i. e. assign a degree k_v to each vertex. The degrees of the vertices are represented as half-links or stubs. The sum of stubs must be even in order to be able to construct a graph ($\Sigma k_i = 2m$). The degree sequence can be drawn from a theoretical distribution or it can represent a real network that can be determined from the adjacency matrix of the network. (ii) Choose two vertices uniformly at random. Connect them with an edge using up one of each node's stubs. Choose another pair from the remaining $2m - 2$ stubs and connect them. Continue until you run out of stubs. Thus, even though the node degree distribution of the graph remains intact, the configuration model results in a completely random network.

Now, if we randomly select two nodes v and u with node degrees k_v and k_u respectively and rewire the stubs for these two nodes, then,

$$E[u, v] = \frac{k_v k_u}{2|\mathcal{E}|}$$

where $E[u, v]$ is expectation of full edges between v and u and $2|\mathcal{E}|$ is total number of rewiring possibilities. Hence, the difference between the actual number of edges between node v and u and the expected number of edges between them is

$$Q = A_{vu} - \frac{k_v k_u}{2|\mathcal{E}|} \tag{1}$$

Summing over all node pairs gives the equation for modularity, Q

$$Q = \frac{1}{2|\mathcal{E}|} \sum_{vw} \left[A_{vu} - \frac{k_v k_u}{2|\mathcal{E}|}\right]\left(\frac{s_v s_u + 1}{2}\right) \tag{2}$$

Modularity is a global quality function and aims to find the community structure of the network as a whole. By optimizing the modularity measure over the space of all partitions, one aims to identify groups of nodes that are more densely connected to each other than one would expect from a statistical null model of the network. This statistical null model is commonly chosen to be the configuration model with preserved degree sequence. The

[1]Image by Albert-László Barabási

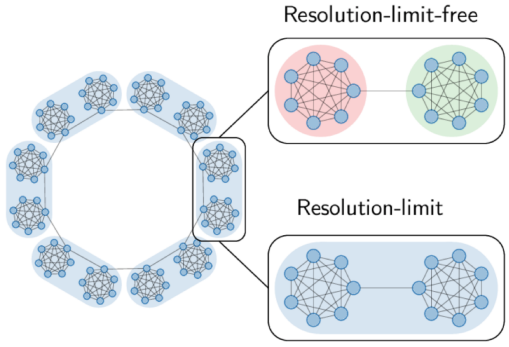

Figure 3: Example of Resolution limit problem. The clustering as optimised by modularity (maximum Q). Each shaded area represents a cluster in two alternative clusterings of a schematic network. The thicker links have double the weight of the thinner links.

null model proposed by Newman and Girvan consists of a randomized version of the original graph, where edges are rewired at random, under the constraint that the expected degree of each vertex matches the degree of the vertex in the original graph. However, a by-product of this choice of a global null-model is the tendency of modularity to balance the size of the groups in terms of their total connectivity. While different variants of modularity aim to account for this effect, it means modularity can be interpreted as a trade-off between a cut-based measure and an entropy. Modularity is typically optimized or maximized with spectral or greedy algorithms [3, 20].

The exact maximization of modularity is hard, so several heuristics [3, 20, 23, 40] were proposed to maximize modularity. Actually, the methods of greedy optimization of modularity often tend to form large communities through combination of small ones. Recent research shows that modularity is not a scale-invariant measure; hence, by relying on its maximization, detection of communities smaller than a certain size is impossible. This serious problem is famously known as the resolution limit of modularity-based algorithm [20] as shown in Figure 3. Nevertheless, modularity optimization for community detection is still popular among researchers. The main reason is the claim that the resolution limit can be removed by adopting a tunable resolution parameter [9] of modularity. It enables one to predefine the size of the clusters to arbitrary values, from very large to very small. However, real networks are characterized by the coexistence of clusters of very diverse sizes, whose distribution follows a power-degree law [7, 8]. Therefore, there is no characteristic cluster size, and tuning a resolution parameter may not help [9]. Another line of work requires the nodes to be more connected to other nodes within the community than those are outside the community, either individually or in groups. This group measure leads to minimizing the community's conductance. The conductance is measure of quality of the cut $(C, \hat{C})$. Given a graph $G = (\mathcal{V}, \mathcal{E})$ with adjacency matrix A, the conductance of set of nodes C is defined as :

$$\Phi(C) = \frac{\sum_{v \in C, u v \notin C} A_{vu}}{\min\{vol(C), vol(\hat{C})\}}$$

Where $\hat{C} = \mathcal{V} / C$. Conductance is measured with respect to the set C or $\hat{C}$ with smaller volume, and is the probability of picking an edge from the smaller set that crosses the cut. However, optimization of

conductance [29, 30] will always give one community which would be the graph itself. In this paper, we define a flexible notion of a node's membership score, merit function and design a community detection framework, which efficiently explores its local neighborhood and propagates the membership. The algorithm starts by placing each data point in a cluster by itself and then repeatedly merges or splits clusters until the merit function's stopping condition is met; the clustering process stops when all data points are placed in a respective cluster and there is no room for improvement in the merit score. Our algorithm generalizes prior work which is based on rigid notions of network modularity [3, 10], and we show that the added flexibility in exploring modularity is the key to detecting well-integrated communities. We demonstrate the efficacy of our method over existing state-of- the-art techniques on clustering in several real-world networks from diverse domains. Taken together, our work represents a new way for efficiently detecting hierarchical communities in complex networks. Our contributions in this work are as follows:

- We propose an unsupervised hierarchical agglomerative community detection (HACD) method based on the graph's local information and membership propagation on undirected graphs and formulate merit function $\Delta\mathcal{M}$ for clustering as a maximization problem of this function $\Delta\mathcal{M}$. The goal of our proposed detection framework is to automatically detect the number of clusters and a node assignment to most relevant cluster for a given input graph.
- Our algorithm can find the communities of various sizes. Moreover, the clustering result does not depend on the order of processed nodes. Experimental results show that our algorithm is effective and efficient. To speed up our methods, we carefully handle the correlations between graph node and edge potentials by systematically investigating their relationship in a data-driven way.
- We compare our method with baseline approaches on a variety of real-world datasets and show that it satisfies all the desired properties, and it is scalable.

Reproducibility: For reproducibility, the source code will be made available upon publication at the link.

2 BACKGROUND AND RELATED WORK

Many state-of-art methods have been proposed to perform community identification or detection [45, 48, 49] by using different approaches. They can be grouped into categories, based on different criteria, like the actual operational method or the underlying concept of community. In this section, we present a critical review of baseline and few other methods.

2.1 Review of methods based on optimization

In this section, we first review the definition of modularity and the corresponding optimization approaches. Then, we discuss the coexisting problems of modularity maximization.

2.1.1 Definition of Modularity. Comparing results of different network partitioning algorithms can be challenging, especially when network structure is not known beforehand. A concept of modularity defined in Eq. (3) provides a measure of the quality of a particular partitioning of a network. Modularity (Q) quantifies the community strength by comparing the fraction of edges within the community with such fraction when random connections between the nodes are made. The justification is that a community should have more links between themselves than a random gathering of people. Thus, the Q value close to 0 means that the fraction of edges inside communities is no better than the random case, and the value of 1 means that a network community structure has the highest possible strength. Formally, modularity (Q) can be defined as Eq. (3):

$$Q = \sum_{C_v \in C}^{K} \frac{|\mathcal{E}^{in}_{C_v}|}{|\mathcal{E}|} - \left(\frac{2|\mathcal{E}^{in}_{C_v}| + |\mathcal{E}^{out}_{C_v}|}{2|\mathcal{E}|}\right)^2 \tag{3}$$

where C is the set of all the communities, C_v is a specific community in C, $E^{in}_{c_v}$ is the number of edges between nodes within community C_v,$E^{in}_{c_v}$ is the number of edges from the nodes in community C_v to the node outside C_v, and $|\mathcal{E}|$ is the total number of edges in the network.

Modularity can also be expressed (as discussed in previous section) in the following form 4:

$$Q = \frac{1}{2|\mathcal{E}|} \sum_{v,u} \left(A_{vu} - \frac{k^{in}_v k^{out}_u}{2|\mathcal{E}|}\right)\delta_{C_v, C_u} \tag{4}$$

where k_v is the degree of node v, A_{vu} is an element of the adjacency matrix, δ_{C_v, C_u} is the Kronecker delta symbol (The symbol is 1 if the nodes are in same group, and 0 otherwise), and C_v is the label of the community to which node v is assigned. Since larger Q means a stronger community structure, several algorithms which we will discuss in the next section, are based on modularity optimization However, this definition can be naturally extended to apply to directed networks as well as to weighted networks. Weighted and directed networks contain more information than undirected and unweighted ones and are therefore often viewed as more valuable but also as more difficult to analyze than their simpler counterparts. Since the structure of adjacency matrix is independent of the edge weights, it is possible to adjust all the methods developed for unweighted networks to the weighted ones. It is necessary to point out that the notion of degree of a node should also be extended for the weighted graphs. In this case degree of a node is defined as the sum of weights of all edges incident to this node.

2.1.2 Modularity Optimization Approaches . Optimization techniques have received the greatest attention in the literature. The goal is finding an extremum, usually the maximum, of a function indicating the quality of a clustering, over the space of all possible clusterings. Quality functions can express the goodness of a partition or of single clusters. The idea behind these methods is that a good community structure must present high values of modularity[10] measure. Therefore, it is natural to discover communities by maximizing modularity over all possible partitions of a network. However, it is computationally prohibitively expensive to exhaustively search all such partitions for the optimal value of modularity since modularity optimization is known to be NP hard. However, many heuristic methods were introduced to find high-modularity partitions in a reasonable time.Here, we will review some of these modularity optimization heuristics methods.

Fast Greedy (FG) [19] performs a greedy modularity optimization with the agglomerative hierarchical approach. Initially, every

node belongs to its own community, creating altogether $|\mathcal{V}|$ communities. Then, at each step, the algorithm repeatedly merges pairs of communities together and chooses the merger for which the resulting modularity is the largest.Limitation of this method is that update of the adjacent matrix at each step contains a large number of unnecessary operations when the network is sparse and therefore its matrix has a lot of zero entries. Another type of greedy modularity optimization algorithm different from those above was proposed by Blondel et al., and it is usually referred to as Multi level Louvain [3]. It is fast modularity optimization methods that have a two-step process i.e. network collapse and greedy optimization. The Louvain algorithm (ML) is one of the first scalable methods to build on Newman-Girvan modularity maximization. It is a hierarchical agglomerative method that takes a greedy approach to local optimization. The algorithm is based on two steps. In the first step, the algorithm iterates over the nodes in the graph and assigns each node to a community if the assignment will lead to an increase in modularity. In the second step, the algorithm creates super-nodes out of the clusters found in the first step. The process repeats iteratively, always using the base-graph to compute the gains in modularity. Although the underlying computational problem is NP-hard, the Louvain algorithm relies on an efficient and effective heuristic that balances solution quality, measured by modularity, and computational complexity, which, although not precisely known, scales roughly linearly with the number of edges. However, it has been proved that optimizing the modularity yields to the problem of resolution limit, making modularity based methods unable to detect communities of small sizes despite those real-world complex systems inherent small groups most of the time. Also, the results of Louvain are impacted by the order in which the nodes in the first phase are considered for merging. Another recent work includes Fine-tuned Qds [6] community detection algorithm which optimizes modularity density for the given network. Most recent work is proposed by Sandefinitionkar et al. [40], that is based on CEIL score optimization. Similarly, Huang et al. [20] proposed a two-stage parameter-free extension of density-based clustering called DenShrink [20] which first finds smaller communities using the highest local structural similarity for a pair of nodes and then iteratively optimizes the modularity measure upon joining these smaller communities.

Spectral modularity optmization methods: There are two categories of spectral algorithms for maximizing modularity: one is based on the modularity matrix [27], [31], [36]; the other is based on the Laplacian matrix of a network [44],[39] (out of scope)

Modularity (Q) can be expressed as [27]

$$Q = \frac{1}{4|\mathcal{E}|}\sum_{v,u}\left(A_{vu} - \frac{k_v^{in}k_u^{out}}{2|\mathcal{E}|}\right)(s_u s_v + 1) = \frac{1}{4|\mathcal{E}|}s^T B s \quad (5)$$

where A_{vu} are the elements of adjacent matrix $A$and s is the column vector representing any division of the network into two groups. Its elements are defined as $s_v = +1$ if node v belongs to the first group and $s_v = -1$ if it belongs to the second group. B is the modularity matrix with elements

$$B = A_{vu} - \frac{k_v^{in}k_u^{out}}{2|\mathcal{E}|} \quad (6)$$

Representing s as a linear combination of the normalized eigenvectors x_v of B: $s = \sum_{v=1}^{|\mathcal{V}|} a_v x_v$ with $a_v = x_v^T.s$ and then plugging the result into Equation 5 yield

$$Q = \frac{1}{4|\mathcal{E}|}\sum_v a_v x_v^T B \sum_v a_v x_v = \frac{1}{4|\mathcal{E}|}\sum_v a_v^2 \beta_v \quad (7)$$

where β_v is the eigenvalue of B corresponding to eigenvector x_v. To maximize Q above, Newman proposed a spectral approach to choose s proportional to the leading eigenvector u_1 corresponding to the largest (most positive) eigenvalue β_1. Moreover, Newman proposed to divide network into more than two communities by repeatedly dividing each of the communities obtained so far into two until the additional contribution ΔQ to the modularity made by the subdivision of a community c.

$$Q = \frac{1}{4|\mathcal{E}|}s^T B^{(c)} s \quad (8)$$

$B^{(c)}$ in the formula above is the generalized modularity matrix. Although [31] explored using multiple leading eigenvectors of the modularity matrix, it did not pursue it in detail beyond a two-eigenvector approach for bipartitioning [27], [31]. Richardson et al. [36] provided a extension of these recursive bipartitioning methods by considering the best two-way or three-way division at each recursive step to more thoroughly explore the promising partitions. To reduce the number of partitions considered for the eigenvector-pair tripartitioning, the authors adopted a divide-and-conquer method and as a result yielded an efficient approach whose computational complexity is competitive with the twoeigenvector bipartitioning method.

However, the spectral algorithm described above has two drawbacks. First, it divides a network into more than two communities by repeated division instead of getting all the communities directly in a single step. Second, it only uses the leading eigenvector of the modularity matrix and ignores all the others, losing all the useful information contained in those eigenvectors. These methods scale poorly to large networks because of running k-means partitioning up to K times.

Other optmization methods: Infomap [38], which is based on random walk dynamics and optimize map equation [37] to detect communities. Infomap characterizes the problem of finding the optimal clustering of a graph as the problem of finding a description of minimum information of a random walk on the graph. The algorithm maximizes an objective function called the Minimum Description Length, and in practice an acceptable approximation to the optimal solution can be found quickly. Previous studies have found Infomap's performance to remain stable for networks with up to 100,000 nodes. The limitation of this method is that it has high computational complexity in massive networks. OSLOM [23] method searches for clusters via a local optimization of a significance score [23] and capable to detect clusters in networks by accounting the edge directions, edge weights, overlapping communities, hierarchies and community dynamics. However, experimentally this method also show high computational complexity in large graphs.

2.1.3 Resolution limit. Since its inception, the modularity has been used extensively as the measure of the quality of partitions

produced by community detection algorithms. In fact, if we adopt modularity as a quality measure of communities, the task of discovering communities is essentially turned into the task of finding the network partitioning with an optimal value of modularity. However as properties of the modularity were studied, it was discovered that in some cases it fails to detect small communities. There is a certain threshold,such that a community of the size below it will not be detected even if it is a complete subgraph connected to the rest of the graph with a single edge. This property of modularity has become known as the resolution limit. Although the resolution limit prevents detection of small communities, the actual value of the threshold depends on the total number of edges in the network and on the degree of interconnectedness between communities. In fact, the resolution limit can reach the values comparable to the size of the entire network causing formation of a few giant communities (or even a single community) and failing to detect smaller communities within them. It makes interpreting the results of community detection very difficult because it is impossible to tell beforehand whether a community is well-formed or if it can be further split into subcommunities.

Considering modularity as a function of the total number of edges, $|\mathcal{E}|$ and the number of communities, K, makes it possible to find the values of m and $|\mathcal{E}|$ which maximize this function. It turns out that setting $K = \sqrt{|\mathcal{E}|}$ yields the absolute maximal value of modularity. Consequently, modularity has a resolution limit of order $\sqrt{|\mathcal{E}|}$ which bounds the number and size of communities. In fact, if for a certain community the number of edges inside it is smaller than $q = \sqrt{|\mathcal{E}|}/2$, such community cannot be resolved through the modularity optimization. It is also possible for modularity optimization to fail to detect communities of larger size if they have more edges in common with the rest of the network. Therefore, by finding the optimal value of the modularity we are generally not obtaining the best possible structure of communities.

There have been extensive studies done on how to mitigate the consequences of the modularity resolution limit. The widely used approach is given by Arenas et al. as described next. The modularity resolution limit depends on the total weight $2|\mathcal{E}|$. By varying the total weight, it is possible to control the resolution limit, effectively performing community detection at different granularity levels. Changing the sum of weights of edges adjacent to every node by some value r results in rescaling topology by a factor of r. Further studies revealed that it suffers from two major issues. First, when the value of the resolution parameter r is low it tends to group together small communities. Second, when the resolution is high, it splits large communities. These trends are opposite for networks with a large variation of community sizes. Hence, it is impossible to select a value of the resolution parameter such that neither smaller nor larger communities are adversely affected by the resolution limit.

2.2 Review of other community detection approaches

2.2.1 Methods based on statistical inference. Statistical inference provides a strong set of tools to handle the problem of community detection. The standard approach is to fit a generative network model on the data. The SBM (stochastic block model)[19] are famous generative model of graphs with communities. These are divided into three caregories i.e. Labelled SBMs [22, 50] allowing for edges to carry a label, Degree-corrected SBMs [23], allowing for a degree parameter for each vertex to scales the edge probabilities, and Overlapping SBMs [1, 32] is based on the mixed-membership SBM. The most important drawback of this type of approach is the need to specify the number K of groups beforehand, which is usually unknown for real networks.

2.2.2 Methods based on spectral clustering. Spectral graph clustering is an approach to detect clusters using spectral properties of the graph [9, 24]. Spectral clustering consists of generating a projection of the graph vertices in a metric space, by using the entries of those eigenvectors as coordinates. Spectral clustering algorithms typically start from local information encoded in a weighted graph on the data and cluster according to the global eigenvectors of the corresponding (normalized) similarity matrix. Spectral clustering that uses the first K eigenvectors for finding K clusters also suffers from the unsuitability of normalized cut for partitioning [29, 30]. Hence, spectral clustering is not always reliable.

2.2.3 Methods based on consensus clustering. Consensus clustering is based on the ideas of combining the information of the different outputs of partition into a new partition [12, 25]. The goal is searching for a consensus partition, that is better fitting than the input partitions. Consensus clustering is a difficult combinatorial optimisation problem and dealing with large number of data items can be problematic because of time complexity and uncertainty of these methods [12].

2.2.4 Methods based on label propagation. Raghavan et al. [35] proposed label propagation methods that simulate the diffusion of information (labels) through the network. Initially, each vertex is assigned a unique label and it uses an iterative process to find stable communities in a graph. The method begins by giving each node in the graph a unique label. Then, the algorithm iteratively simulates a process in which each node in the graph adopts the label most common amongst its neighbors. The process repeats until the label of every node in the graph is the same as the label of maximum occurrence amongst its neighbors. The outcome of these methods is sometimes unpredictable and it is possible that the whole network might end up having the same label. Other recent work is Attractor [41], which computes the distance on edges, based on Jaccard similarity and apply interaction patterns to propagate the labels. To control the resolution problem, the author introduced cohesion parameter λ to detect suitable clusters. The author [46] proposed SLPA a general speaker-listener based label propagation process, which spreads a label at a time between nodes according to interaction rules. Same line work is clustering via node embeddings [14–16, 34], these methods require prior knowledge of labels and comparing them is out of the scope of this paper.

2.2.5 Methods based on matrix/tensor factorization. More recently, graph clustering methods that rely on the Non-negative Matrix Factorization (NMF) [26] have been proposed e.g., [5, 11, 47]. Their goal is to approximate an adjacency matrix of a given network by a product of two non-negative low-rank matrices, such that they have clustering interpretation, i.e., they can be used for

assigning nodes to communities. Another line of work is done with tensor factorization (TF) [2, 13, 17, 33, 43], these techniques find communities in multi-layer graphs. Various Heterogeneous Information Network (HIN) [4, 18] method for community detections is proposed to solve this problem.

3 PRELIMINARIES

Graphs are an effective way to represent a large variety of data and relations between data entities. Each entity represented by a node or vertex ($v \in \mathcal{V}$) and the relation between entities are defined by weighted or unweighted edges ($\mathcal{E}, w$), where $\mathcal{E}$ is set the edges connecting a pair of vertices $(v, u) \in \mathcal{E}$ and w is the corresponding weight on the edge. The graph can be represented using an adjacency matrix, a square node-by-node matrix that indicates an edge (weight associated with it) between two nodes. Table 1 contains the symbols used throughout the paper.

Symbols	Definition
A, **a**, a	Matrix, Column vector, Scalar
$\leftarrow$	A direction of the link between nodes
$N[v]$	Neighbors of node v
k_v^{in}, k_v^{out}	in-degree and out-degree of the node v
C,C_i,$C(v)$	Set of communities, i^{th} community in C, community membership of node v
$k_{C(v)}^{in}$, $k_{C(v)}^{out}$	total in-degree and out-degree of the partition where node v belongs

Table 1: Table of symbols and their description

3.1 Problem Definition

Before we conceptualize the problem that our paper deals with, we define certain terms which are necessary to set up the problem.

Definition 3.1. **Node Potential Score:** Given a partition or community, total degree (we call it *vol*) and the incoming weights corresponding to nodes $u \in N[v]$ that belong to community $C(v)$ is referred as node potential score for node v, which help to encode the similarity as well as the strength of staying in its existing community $C(v)$.

$$\phi(v) = vol_{C_v}(v) + \sum_{\substack{(u,v)\in C(v) \\ (u \to v)}} W_{u \to v} \tag{9}$$

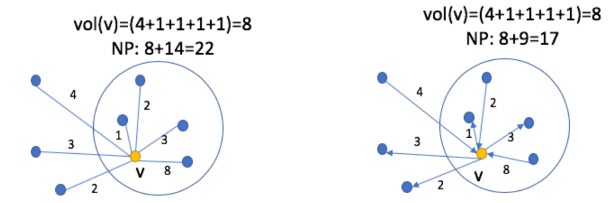

Figure 4: The node potential score of a node shows how many highly informative nodes are pointing to this node.

Definition 3.2. **Edge Potential Score:** Connections corresponding to node $v \in \mathcal{V}$ and its neighbors $u \in N[v]$ referred as edge potential score which captures the sum of weights on edges connected to/from observed node v, or more general prior knowledge about the node relationship.

$$\psi(v) = \sum_{\substack{(v,u)\in\mathcal{E} \\ u\in N[v]}} W_{vu} \tag{10}$$

Figure 5: The edge potential score or weighted degree of a node shows the amount of valuable information that this node holds.

Definition 3.3. **Membership Score ($\mathcal{M}(v)$):** We define *membership score* as the full or partial sum of node and edge potential score to measure the overall individual or group of node's strength to stay in its assigned community. This is a scalar quantity and $\mathcal{M}$ has a superscript referred to the corresponding node and a subscript to the category on which it is computed.

Definition 3.4. **Merit Function $\mathcal{M}(v, N[v])$:** Given single or individual node membership score ($\mathcal{M}_{snm}^{v}$), neighboring node membership score ($\mathcal{M}_{nnm}^{N[v]}$) and adjacent community neighboring score ($\mathcal{M}_{ncm}^{v\in C(N[u])}$). Our goal is to find maximum $\mathcal{M}(v, N[v])$ while enforcing it should be strictly positive for each community.

$$\mathcal{M}(v, N[v]) = \max_{u\in N[v]} \left(\mathcal{M}_{nnm}^{N[v]} - \frac{\mathcal{M}_{snm}^{v} * \mathcal{M}_{ncm}^{v\in C(N[u])}}{2|\mathcal{E}| + \min\{vol(C_{C(v)}), vol(C/C_{C(v)})\}}\right) \tag{11}$$

Intuition: Our merit function uses the idea that if a natural division of a network exists, connections within a community should be stronger, and the opposite should hold true for connections between communities. If an edge ($\mathcal{E}(u, v)$) is stronger than its expected membership score ($\mathcal{M}_{snm}$), it contributes positively to the merit function, provided that the two nodes (u, v) connected by the edge belong to the same community. Divisions that increase merit value are preferred because they lead to communities with high community structure.

Definition 3.5. **Membership propagation:** This is a message passing method that iteratively computes the membership score $\mathcal{M}$ of node v with respect to a neighborhood system $u \in N[v]$ and pass the community membership when $\mathcal{M}(v, N[v])$ is strictly positive.

$$C(v) \leftarrow C(u),\ s.t.\ \mathcal{M}(v, u) > 0 \tag{12}$$

Problem. Given a graph or network $G = (\mathcal{V}, \mathcal{E})$, with vertex or node set $\mathcal{V}$ and edge set $\mathcal{E}$, we can represent the graph as an adjacency matrix **A** such that $\mathbf{A}_{uv} = W_{uv}$, where W_{uv} is edge weight between vertices u and v, or $\mathbf{A}_{uv} = 0$, if there is no edge. The graph partitioning problem can be defined as creating K disjoint set of vertices i.e. clusters $C_1, C_2, \ldots C_K$ such that $\{C_1 \cup C_2 \cup \cdots \cup C_K\} = V$ and $\{C_1 \cap C_2 \cap \cdots \cap C_K\} = \{\}$. The objective of partitioning is to maximize the merit function

$\mathcal{M}(v, N[v])$ which uses the local connectivity structure of the network in our *best* clustering.

4 PROPOSED METHOD

The first part of this section describes the cluster formation and optimization of the merit function, which is a key contribution of this work, the second subsection elaborates the merging or splitting of nodes to form communities that identify the most relevant cluster for given vertices in the graph. We also present the termination condition and provides a complexity analysis of the proposed method in this section.

Summary: Given a graph, the HACD algorithm is summarized as follows:

- For each node ($v \in \mathcal{V}$) in given graph G , compute single node membership score $\mathcal{M}^v_{snm}$, its corresponding neighbourhood membership score $\mathcal{M}^u_{nnm}$ s.t. $u \in N[v]$ and neighboring community membership score $\mathcal{M}^{(v,u)}_{ncm}$, when placing the v in $C(u)$.
- Find the set of nodes with a maximum positive change in membership score (i.e., merit function) $\mathcal{M}(v, N[v])$ given above described membership score based on a node potential $\phi(v)$ and edge potential $\psi(v)$ $s.t.$ $v \notin N[v]$; we denote these sets as $C = \{C_1, C_2, \ldots, C_K\}$, where $C_i = \{v_1, v_2, v_3, \ldots, v_m\}$ and $C_i \subset \mathcal{V}$.
- Detect the best partition in the graph in the propagation phase and propose the clustering of nodes.

4.1 Cluster Detection Phase

The intuition of clustering is to separate nodes in different groups according to their similarities. Step(i) of phase employs the membership score as quantity to identify the largest merit among the neighborhood nodes. Our algorithm allocates different communities to each node in the graph, which means that initially we have total number of communities same as the number of nodes in the graph i.e. $\#C = \#\mathcal{V}$. Next, for each node, we calculate the single node membership score given as :

$$\mathcal{M}^v_{snm} = \phi(v) + \psi(v) = \Big(vol(v) + \sum_{\substack{(u,v)\in C(v)\\(u\to v)}} W_{u\to v}\Big) + \sum_{u\in N[v]} W_{uv} \tag{13}$$

where former part $\phi(v)$ is a node potential score, and later part $\psi(v)$ represents an edge potential score(weights on edge) for nodes $u \in N[v]$ in any cluster C and $vol(v)$ represents the total degree of the nodes in set $C_{C(v)}$ where node v belongs initially, commonly called its volume in analogy with geometric objects and $W_{u\to v}$ is incoming weights of the nodes in set $C_{C(v)}$.

Similarly, for each neighbor $u \in N[v]$, we calculate its neighboring node (NN) membership score i.e sum of the volume of u and the weights of all the incoming and outgoing links in the network to/from node $u \in C$, given as:

$$\mathcal{M}^u_{nnm} = vol(u) + \sum_{\substack{u\in N[v]\\w\in N[u]}} W_{uw} \tag{14}$$

In the next step, we remove the node v from its community $C_{C(v)}$ and place it in its neighbor communities $C_{C(u)}$ and calculate the neighbor community membership score as:

$$\mathcal{M}^u_{ncm} = \begin{cases} \sum\limits_{\substack{v\leftrightarrow u\\u\in N[v]}} W_{vu} + \sum\limits_{\substack{u\in N[v]\\w\in N[u]}} W_{uw} & v \neq u \\ \sum\limits_{\substack{v\leftrightarrow u\\u\in N[v]}} W_{vu} + \sum\limits_{\substack{u\in N[v]\\w\in N[u]}} W_{uw} - M^v_{snm} & v = u \end{cases} \tag{15}$$

Here we focus on a local perspective on community detection that gives us two appealing properties: First, clusters do not depend on the global network structure but only depend on the relative local density. Second, only a subgraph of the network needs to be accessed, which is advantageous in case of computational constraints (time and memory) in using massive graphs, or we are only interested in a specific group or sub-system to analyze. In such cases, we would prefer to not applying a method to the whole network in order to find, for example, the cluster containing a particular node in the network. Finally, we compute our merit or objective function that helps us to decide whether we keep the selected node in the same community or not. It is defined as :

$$\mathcal{M}(v, N[v]) = \max_u \big(\mathcal{M}^u_{nnm} - \frac{\mathcal{M}^v_{snm} * M^u_{ncm}}{2|\mathcal{E}| + \min\{vol(C_{C(v)}), vol(C/C_{C(v)})\}}\big) \tag{16}$$

where $\mathcal{M}(v, N[v])$ are set of merit values computed for each neighbor node u of observed node v. If the $\mathcal{M}(v, N[v])$ is negative, node v stays in same community and for all other positive values, we find the node u that correspond to maximum merit value i.e. $\max(\mathcal{M}(v, N[v]))$ and node v is then assigned to the community same as node u i.e u passing the membership or label to node v and update the network by removing it from unobserved nodes. We repeat the process for all the nodes in the network until we did not see any further improvement in merit function. This preprocessing not only helps in reducing the number of vertices needed to be considered during each iteration, but it also allows the vertices that contain multiple neighbors to be the main drivers of community migration decisions. It is observed that the equation 16 is a variant of the modularity (Equ. 4) based methods that include only edge potential when partitioning the graph and highly dependent on the eigenvectors of the adjacency matrix. However, HACD is able to successfully exploit node and edge strength together to find hierarchical communities and does not depend on the eigenvectors of the adjacency matrix or graph. Also, modularity based methods are not able to find small clusters but HACD is easily able to detect the smaller communities which have a strong similarity of nodes in the group based on membership score as explained above and provides strong cohesiveness of groups with high internal density.

4.2 Propagation Phase

Once we get the set of vertices or nodes forming intermediate communities, we further expand each of community to the regions of neighboring communities found in the detection phase. Our assignment method is straightforward: first, we update the weights of edges of nodes in the community by the sum of the weights of the connections between nodes in the corresponding communities. Next, we find the bridge nodes between community C_i and network and detach it from the community C_i. Finding the bridge node is not an expensive process. When we update the network, we have

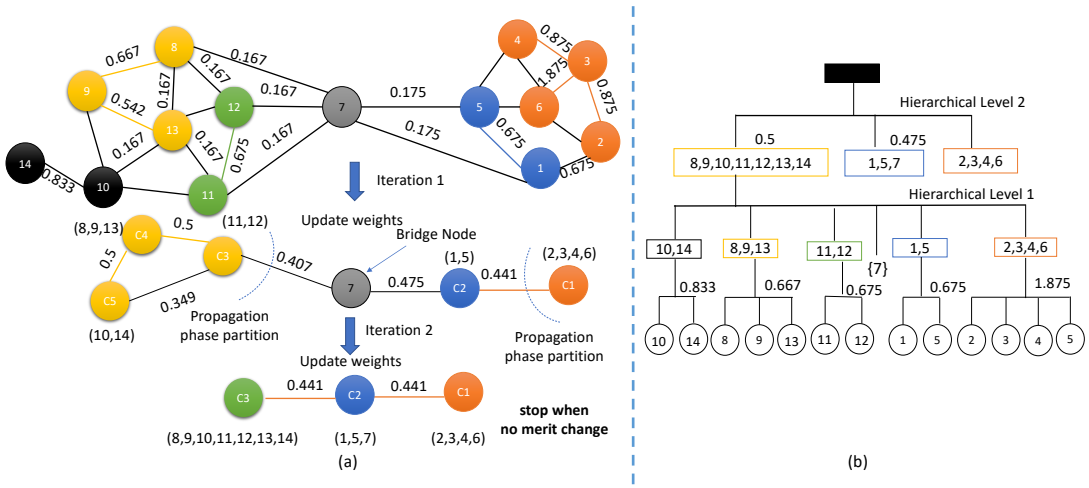

Figure 6: Illustration of the procedure and result of the hierarchical network clustering algorithm HACD: (a) the process of the hierarchical clustering per iteration, (b) the hierarchical structure of the detected communities at different level. Numerical values indicates the maximum merit $\Delta\mathcal{M}$ value found between nodes.

nodes those are connecting the two communities as shown in figure 6. These nodes are considered as a bridge in the graph. Next, we add that piece to all the communities $C_j \subset C$ that utilize the node in the bridge. We propagate the membership if it improves the merit function in the new network formed by merging two communities. This is described in Algorithm 1. Let us denote $C_i^{'}$ as the expanded community C_i, $G_b = (\mathcal{V}_b, \mathcal{E}_b)$ as bridge edges or small subgraph that connect the C_i and C_j. When we expand community using Algorithm 1, we can say that $C_i^{'} = C_i \cup C(\mathcal{V}_b)$ and $\mathcal{M}(C_i^{'}) \geq \mathcal{M}(C_i)$. In case of large communities, the algorithm will split a community $C_i$into two or more meta-communities C_{i_1} and C_{i_2}, if the split improves the value of the $\mathcal{M}(v, N[v])$. This density cut [42] finds the partitioning that minimizes the ratio of edge and node potential scores as given by $\frac{\sum \psi(C_i, C_j)}{\min(\phi(C_i), \phi(C_j))}$. This minimization penalizes the large communities in which either of the splitted communities are small and thus favors balanced partitioning over unbalanced ones. If there is no improvement, the community C_i stays the same. We do not need to run the splitting multiple times. Once the assignment is done with the detection phase and re-creation of a new network, we perform splitting over generated communities only. This will save the computation time for large graphs. As this phase of our algorithm highly involved in moving nodes from one community to other communities, it is able to identify smaller communities and does not suffer from the resolution limit problem. We show that our merging/splitting phase improves the quality of final clustering results in terms of qualitative measure such as conductance etc. of the network.

Finally, we analyze the computational complexity of our algorithm HACD. The running time of HACD is mainly consumed by finding positive change in membership score via merit function while moving the nodes from its own community to another community and propagating the membership scores to nodes in each iteration. The time complexity of the detection phase is $O(VlogV)$ for the network with V nodes. The propagation phase requires to process all communities, results in the time complexity of this phase is $O(K * V)$. If there are n steps for the algorithm to terminate,

Algorithm 1: HACD framework

Input: $G = (\mathcal{V}, \mathcal{E})$
Output: Communities C of G

Detection phase
for $v \in \mathcal{V}$ **do**
  Compute $\mathcal{M}^v_{snm}, \mathcal{M}^u_{nnm}, \mathcal{M}^u_{ncm}$ using equation 13,14 and 15
  Find $u \in N[v]$ s.t $\max_u \mathcal{M}(v,u)$ using equation 16
  $C(v) \leftarrow C(u),\ s.t.\ \mathcal{M}_{uv} > 0$
end for
Update $\mathcal{E}(v,u) \leftarrow \sum_{u \in C_i} \mathcal{E}(C_i)$
for each $C_i \in C$ **do**
  Detect bridges $e \in \mathcal{E}_b$ attached to C_i
  for each $e \in \mathcal{E}_b$ **do**
    Update $C(\mathcal{V}_{e_i}) \leftarrow C(\mathcal{V}_{e_j})$
    Compute $m = \mathcal{M}(e_j, \mathcal{V}/e_j)$
    if $m > 0$ **then**
      $C_i \leftarrow C_i \cup C(\mathcal{V}_{b_j})$
    else
      $C_i \leftarrow C_i \cap C(\mathcal{V}_{b_j})$
    end if
  end for
end for

which results in an overall time complexity of $O(n(VlogV + KV)) \approx O(VlogV)$ given that $K << N$.

4.3 Terminating Condition

We have found the local maximum of merit if any of the following three conditions are satisfied:

- We have visited all nodes in $G = (\mathcal{V}, \mathcal{E})$ and no improvement in merit score of nodes while moving from assigned community C_i to another community C_j where $C_i, C_j \in C$.
- The product of $\mathcal{M}^v_{snm}$ and $\mathcal{M}^u_{nnm}$, divided by sum of weights of all edges in graph, is always greater than $\mathcal{M}_{ncm}$.

- The number of iterations in the detection phase exceeds the user-defined hierarchical depth. The default limit is 20.

4.4 Filling the gap

Our proposed method is variant of the modularity optimization (Equ. 4). HACD contains two phases. The first, detection phase starts from forming each node as a singleton community by calculating the changes of membership scores after moving each node to its neighbor's communities. If the change in membership score is positive, it move the node to the community with max change score. Otherwise, node stays in its own community. Once it reaches the maximal membership score changes, the detection phase ends. Significnace of membership score is that it reveals the intra community strength of node or group of nodes on which it is computed which is missing in the literature. For measureing node membership scores, we consider (i) intra community strength in terms of node potential , (ii) sum of incoming and outgoing edges between nodes $\sum W_{uv} \approx |E_v^{in+out}|$, that reveals its edge potential. Considering equation 4, while computing change in modularity , state of art method considers only product of number of incoming or outgoing edges (k_v^{in} , k_u^{out}) of nodes in community C_i but do not consider contribution of node in terms of its weights between nodes. It is quite possible that nodes have same incoming and outgoing degree in the network, so does not indicate any intra community structure. Our membership score consists of intra community strength that forces the node to stay in its own community until the neightbouring node's membership score is more that its own score. For example, consider an author-author relation , certain authors are co-author with-in university department authors more than authors from other universities. The weight reveals there intra connection strength which is important factor while consideing the community membership. This type of analysis give us more information about local structure of the sub-graph.

State of art methods like [3, 19, 20, 23], compute the global (considering entire network) modularity gain score with respect to clusters detected after moving nodes from one cluster to another. Our method moves the nodes to certain cluster or community if and only if change in membership score is positive , result help to reduce unnecessary movements of nodes and optimize the objective in local prospective. Our algorithm provides a decomposition of the network into communities for different levels of organization. For instance, when applied on the DBLP network, unlike state-of-art methods, the small sized semi-cliques subgroups are indeed merged in the final step but they are distinct after the first detection phase. This result suggests that the intermediate solutions found by our algorithm may also be meaningful and that the uncovered hierarchical structure may allow the end-user to zoom in the network and to observe its structure with the desired resolution.

As discussed above, finding optimal value of the modularity (being a global measure since it assumes that edges between any pairs of nodes are equally likely) does not necessarily leads to obtaining the best possible structure of communities. Here, in our method we optimize the change in membership in local prospective way that helps to find the communities of small size. The main idea underlying our method is based on performing multiple resolution community detection on essential parts of the network, thus analyzing each part independently. After re-creating the network in propagation phase, method operates iteratively by first finding bridge nodes and placing corresponding nodes in a singe community, and compute density cut (based on the average degree of the community structure) ratio. Finally, it runs the same algorithm on each community that was found. The method terminates when no more split of communities is necessary, which usually takes just a few steps.

4.5 What are the advantages of our solution over the prior art?

- Highly efficient method to control graph's local information for membership propagation that helps to reduce the overhead when user want to analyize sub-graph in large graphs.
- Paramter free, Scaleble and Resulution limit free method.
- Method provides local maximum modularity at the same time finds low conductance cut in the network.
- Uncovered important network hierarchical structures in intermediate steps.

5 EXPERIMENTAL EVALUATION

In this section, we extensively evaluate the performance of HACD on seven synthetic and six real datasets, and compare its performance with state-of-the-art approaches for community detection. We implemented HACD in python 3.5 which supports efficient computations. We used Intel(i7) @2.7GHz machine with 8 CPU cores and 16GB RAM.

5.1 Datasets

- **LFR Hierarchical Benchmark Data**: LFR networks serve as benchmark networks in which certain real-world properties, namely power-law distribution for nodal degree and community sizes. The specifications of each synthetic dataset are given in Table 2.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	d_{avg}	$\#C$	(γ_1, γ_2)	μ
Net1	1, 000	7, 647	15	30	(20 , 50)	0.10
Net2	5, 000	38, 521	15	136	(20 , 60)	0.20
Net3	10, 000	177, 386	35	256	(30 , 50)	0.20
Net4	50, 000	494, 249	20	788	(50 , 80)	0.25
Net5	100, 000	980, 648	20	2568	(30, 50)	0.30
Net6	500, 000	1, 376, 865	10	4012	(100, 500)	0.35
Net7	1, 000, 000	1, 937, 776	8	7290	(500, 1000)	0.40

Table 2: Table of Datasets analyzed, $|N|$ = number of Nodes, $|E|$ = number of Edges, d= average degree, $\#C$ = number of ground truth communities γ_1 , γ_2 = power-law distribution exponents, μ controls the mixing of communities.

- **Real Data**: In order to truly evaluate the effectiveness of HACD, we test its performance against six real datasets that have been used in the literature. Those datasets are summarized in Table 3 and are publicly available at [28].

5.2 Baselines

We selected a set of state-of-the-art methods to be compared with HACD. The selected methods are OSLOM [23] , Multi-Level (ML)[3], Label Propagation (SLPA) [38], CEIL [40], Fine-tuned Qds [6], Den-Shrink [20] and BigCLAM [47]. We used the publicly available

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Description
KDD Publication	11, 176	24, 976	KDD co-authorship network
Gowalla	196, 591	950, 327	Friendship network of Gowalla users
Stanford	281, 903	2, 312, 497	Stanford web graph
Dblp	317, 080	1, 049, 866	DBLP co-authorship network
Google	875, 713	5, 105, 039	Google programming contest web graph
Youtube	1, 134, 890	2, 987, 624	Youtube social network,

Table 3: Large real-world web and social network [28].

implementations (Python or Matlab). The selected algorithms are state-of-the-art and representative in community detection because of their high-quality results. All comparisons were carried out over 5-10 iterations, and each number reported is a mean with a standard deviation attached to it.

5.3 Measures of Cluster Quality

Yang and Leskovec [48] compared the performance of several scoring functions using perturbation experiments and reported that Conductance is one of the best performers. Modularity is another popular definition. So we evaluate the community detection performance in terms of three different quality measures: Conductance [48], Coverage[43], and Modularity[40] when ground-truth communities are not available.

- **Conductance**:Measure of quality of the cut $(C, \hat{C})$. Given a graph G with adjacency matrix A the conductance of set of nodes C is defined as :

$$\Phi(C) = \frac{\sum_{i \in C, j \notin C} A_{ij}}{min\{A(C), A(\hat{C})\}} \tag{17}$$

  where $A(C) = \sum_{i \in C} \sum_{j \in V} A_{i,j} = \sum_{i \in C} d(i)$, where $d(i)$ is degree of ndoe i in G. Hence the conductance of graph G is given by $\Phi_G = \min_{C \subset V} \Phi(C)$. It provides the goodness of a community.
- **Coverage**: The graph coverage indicates how many vertices are as- signed to clusters (i.e., the number of assigned vertices divided by the total number of vertices in a graph.

$$Coverage = 100 * \frac{\text{Intra-community edges}}{\text{Total number of edges}} \tag{18}$$

  If clusters are disjoint, then $Coverage$ = 100%.
- **Modularity**: It measure the strength of sets of a network into modules i.e. communities. It is defined as :

$$Q = \sum_{i=1}^{C} (e_{ij} - a_i^2) \tag{19}$$

  where $e_{ij} = \sum_{vw} \frac{A_{vw}}{2\mathcal{E}} 1_{v \in C_i} 1_{w \in C_j}$ is the fraction of edges with one end vertices in community C_i and the other in community C_j and $a_i = \frac{k_i}{2\mathcal{E}}$ is the fraction of ends of edges that are attached to vertices in community i.

In addition to above measures, Adjusted normalized mutual information (NMI) [43], Average F1-score [43] and Adjusted Random Index [17] are measures for assessing the performance of community identification when ground-truth communities are available.

- **NMI**: Given set of true communities $C = \{C_1, C_2, \ldots, C_R\}$ and predicted $\hat{C} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_{R'}\}$, NMI is defined as

$$NMI(C, \hat{C}) = \frac{2I(C, \hat{C})}{H(C) + H(\hat{C})} \tag{20}$$

  where $H(C)$ denotes the entropy of community C and the mutual information $I(C, \hat{C})$ reflects a measure of similarity between the two communities.
- **Average F1-score**: F1-score is a measure of binary classification accuracy. Average F1-score for detected communities $\hat{C}$ is given by

$$F_{1_{avg}} = \frac{1}{2|C|} \sum_i \frac{2|C_i \cap \hat{C}_{I(i)}|}{|C_i| + |\hat{C}_{I(i)}|} + \frac{1}{2|\hat{C}|} \sum_i \frac{2|C_{I'(i)} \cap \hat{C}_i|}{|C_{I'(i)}| + |\hat{C}_i|} \tag{21}$$

  where $I(i) = \max_j \left(\frac{2|C_i \cap \hat{C}_j|}{|C_i| + |\hat{C}_j|}\right)$ and $I'(i) = \max_j \left(\frac{2|C_j \cap \hat{C}_i|}{|C_j| + |\hat{C}_i|}\right)$
- **Random Index**: Random Index is defined as the ratio of number of nodes correctly extracted to total number of nodes. Formally,

$$RI(C, \hat{C}) = \frac{1}{|V|} \sum_{r=0}^{R} |C_r \cap \hat{C}_r| \tag{22}$$

  where C_r and $\hat{C}_r$ are the set of verticies in the actual and predicted community r , respectively. $|C_r \cap \hat{C}_r|$ is the number of same objects between C_r and $\hat{C}_r$. R is total number of communities.

5.4 Empirical Analysis

5.4.1 Synthetic data. We experiment on LFR hierarchical benchmark with the parameters listed in Table 2. For this experiment, we have run the methods for 5 times, and the mean, as well as standard deviation of quality metric i.e., Adjusted NMI, Avg F1 score, and Adjusted RI, were computed as shown in Figure 8 . As we can notice from Figure 8, the critical point on the performance, for the majority of the algorithms, arrives when $|\mathcal{V}| > 10^5$ with high mixing parameter $\mu >= 0.3$. For $Net1$ with actual 30 ground truth communities, ML, SLPA, CEIL, DenShrink and HACD were able to detect the all 30 communities, whereas OSLOM, Fine-tuned, and BigCLAM detected average 28.5, 37, 39 communities, respectively. For $Net2$ and $Net3$ with actual 136 and 256 ground truth communities,respectively, we see similar behavior. We observed that ML, SLPA, and OSLOM finds a number of communities much lower than the ground truth communities. This number reduces as the $|\mathcal{V}|$ and μ mixing parameter increases; thus when the community structure becomes more difficult to uncover, these algorithms have the tendency to join communities. The average NMI for ML, SLPA and OSLOM are 0.57, 0.83 and 0.71, respectively. CEIL finds the correct partitioning for $\mu = 0.3$, but, for $\mu > 0.3$, the number of clusters is 2, 568 for $Net5$, and then it drastically splits the networks by returning an average of 2, 584, 44, 423, and 199, 341 communities for $Net5$, $Net6$ and $Net7$, respectively and yields average Adj. NMI = 0.71. Fine-tuned and BigCLAM show similar behavior, both split the networks into several communities with $\mu < 0.3$ and $|\mathcal{V}| \leq 5 * 10^4$, but as the $|\mathcal{V}| > 10^5$, both start merging cluster to form large communities and average Adj. NMI is 0.56 and 0.47, respectively. DenShrink is the best performing algorithm offering almost perfect and stable results till $\mu <= 0.3$ and achieves average Adj. NMI equals to 0.87 ± 0.1. However, HACD performs well on all synthetic data and returning an average of $< \pm 3\%$ deviation in the true number of communities and yields average Adj. NMI score = 0.94 ± 0.05. It is worth to point out that when the number

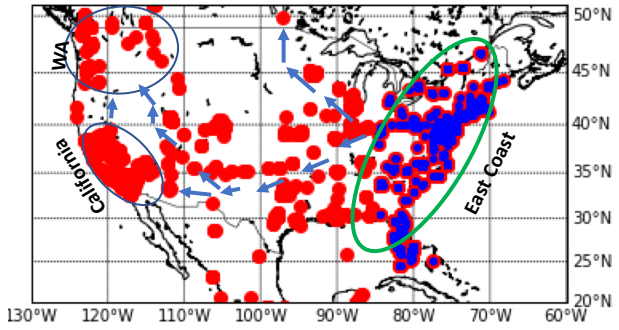

Figure 7: Analysis of community #141, the red color indicates the community member's all visited locations and the blue color indicates start location. Arrows indicate the movement of the users.

of communities obtained by a method is very close to the ground truth consequently, the NMI values are high.

5.4.2 Real data: We have selected some large-scale real-world (Tbl 3) networks to test the performance of HACD. Due to unavailability of ground-truth communities, Adj. NMI , Adj. RI and Avg. F1-score could not be evaluated, thus performance is assessed using the conductance and modularity. As low values of conductance and high values of modularity correspond to more cohesive communities, generally implies better performance. For *KDD publication*, we observed that HACD provides the high-quality communities with respect to the the conductance and modularity score. We further analyze the outcome of baseline methods and observe that DenShrink is able to find few strongly connected communities but fails to merge the two hubs even those share a strong connection. CEIL, SLPA and ML outcome is very surprising, for example, the researcher *"Jiawei Han"* shares a large group of tightly connected researchers like *"Philip S. Yu", "Wei Fan"*, but surprisingly these methods predicted researchers in very small separate communities of size average < 10. BigCLAM, OSLOM, and Fine-tuned merge most of the communities and results in only 160, 148 and 394, respectively. Overall, HACD outperforms with modularity improvement $> 10\%$ and conductance improvement $> 20\%$ with respect to baseline methods.

Gowalla is a location-based social networking website where users share their locations by checking-in and form friendship network. HACD drastically outperforms the baseline methods with respect to finding people with similar visited places. We found that the largest community (#141) is of size $25,857$ consists of $1,536$ location instances, and each location instance is associated with average 123 users. The users from East coast visited places in California, Nevada, Washington state from late September till October end as shown in figure 7. Baseline methods mostly cluster the users together who shares just one common location with the exception of DenShrink. HACD achieves conductance improvement $> 13\%$ with respect to baseline methods. Table 4 describes the overall performance of all methods and Figure 11 present the time taken by each method for the real datasets. For *Stanford* and *DBLP*, most of the methods performed same. BigCLAM reported 2000 communities and rest all reported communities ranging from 3123 to 4528 for both datasets. DenShrink and Fine-tuned outperform with respect to modularity but HACD provides the lowest conductance (12.2% and 14.3% improvement) among all methods. For *Google*, we found 3280 clusters of web pages links. Conspicuously, we note that the baselines are almost on par with HACD, which can be rooted in the fact that Google network structure is easy to capture because of it is highly dense and corresponds to the strong connected components. For *Youtube*, HACD outperforms the baseline methods. Remarkably, it surpasses the baselines most when the data is sparse ($10^{-4}\%$) by average 25% with respect to the conductance and modularity score with the exception of DenShrink that perform similar. However, the resolution of the communities is another important metric which must be considered in drawing conclusions. We provide analysis for this in section 5.5. Interestingly, our algorithm also follows the rule of the power-law distribution (Figure 9) that reveals the granularity of the detected communities and ensures that our algorithm does not suffer from the resolution limit problem.

5.4.3 Multi-Scale Community Detection. Figure 9 shows the community size power law distribution obtained by HACD on DBLP and Youtube network. This outcome indicates that HACD is able to detect communities of any scale varies from small to large scaled communities, independent of network size. This is something that baseline algorithms (ML, SLPA, OSLOM), which suffer from the resolution limit, are lacking.

5.4.4 Hierarchical Community Detection. Another relevant feature of HACD is to find the hierarchical structure of the network. The results are very good on hierarchical benchmarks. HACD generally finds different depths in hierarchical branches. In fact, when the algorithm is applied, not all vertices are grouped, as some of them are group-less. The coexistence of group-less vertices with proper clusters yields a hierarchical structure with branches of different depths. Table 5 shows the hierarchical levels of KDD co-authorship data. Due to the limited space, we can not show all the extracted communities in KDD network. We then select three representative communities based on size and list no more than seven community members along with three representative researcher who are responsible for merging into a large community in Table 5 . Each community represents a group of scientists with the same research interests, such as Data Mining community (3) and Information Retrieval and Web Mining community (1131) in Table 5. Researchers like "Christos Faloutsos" and "Hanghang Tong", have published a large number of papers in collaboration with people from various research communities. These nodes should be considered as tightly related to the same community, forming hierarchies in the network. A scholar who only publishes one paper with these researchers in the community group should not be considered as a member of the group, and meanwhile, it is better to be regarded as a group-less member.

5.5 Analysis of the Resolution Limit Problem

Despite the good performance of the algorithm on many practical networks, it may lead to apparently unreasonable partitions in some cases. We present the clustering results on the two datasets that consist of N (5-100) cliques with 10 vertices each forming a Ring and Pairwise like structure. Every clique has two neighboring cliques, connected to it via a single edge. In case of pairwise, we connect one of the nodes of each clique to the root node. Intuition

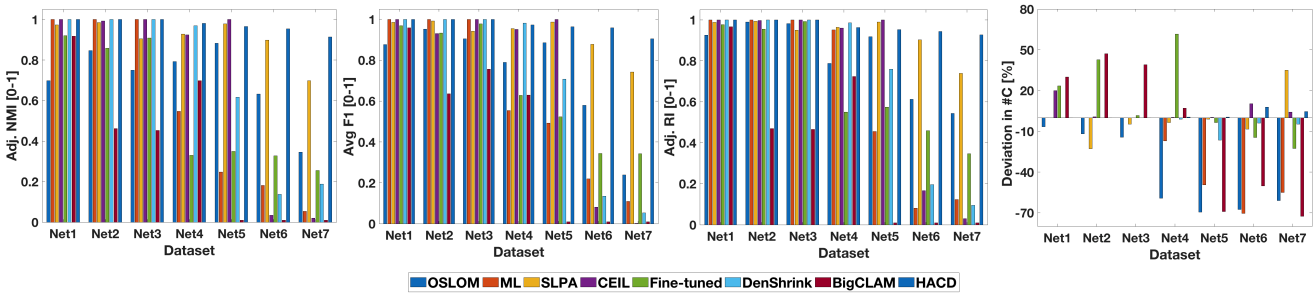

Figure 8: Comparison of HACD 's quality with that of competing techniques for varying number of nodes $|\mathcal{V}|$ as specified in Tbl 2. Higher the value, better the cluster quality.

Dataset	Measure	OSLOM	Multi-Level	SLPA	CEIL	Fine-tuned	BigCLAM	DenShrink	HACD
KDD Publication	Conductance	0.222 ± 0.002	0.151 ± 0.001	0.134 ± 0.002	0.265 ± 0.001	0.59 ± 0.012	0.290 ± 0.001	0.416 ± 0.000	**0.0468 ± 0.004**
	Modularity	0.791 ± 0.002	0.817 ± 0.002	0.829 ± 0.002	0.684 ± 0.001	0.392 ± 0.211	0.541 ± 0.001	0.52 ± 0.001	**0.914 ± 0.004**
	#C	148.0 ± 6.671	1265.4 ± 8.313	1810.2 ± 9.032	**3074.2 ± 7.040**	394 ± 4.000	160 ± 0.000	1737 ± 13.634	1553.4 ± 1.945
Gowalla	Conductance	0.243 ± 0.012	0.367 ± 0.002	0.383 ± 0.004	0.328 ± 0.006	0.599 ± 0.023	0.375 ± 0.014	0.232 ± 0.003	**0.213 ± 0.012**
	Modularity	0.683 ± 0.007	0.627 ± 0.042	0.598 ± 0.035	0.706 ± 0.011	0.271 ± 0.006	0.517 ± 0.004	0.689 ± 0.001	**0.711 ± 0.014**
	#C	**2942.4 ± 72.411**	1202.4 ± 52.934	1081.8 ± 44.523	1219.0 ± 61.854	950.0 ± 29.828	573.7 ± 6.363	2364.4 ± 24.468	2304.4 ± 5.334
Stanford	Conductance	0.415 ± 0.017	0.141 ± 0.003	0.145 ± 0.001	0.153 ± 0.000	0.141 ± 0.000	0.129 ± 0.001	0.139 ± 0.004	**0.124 ± 0.02**
	Modularity	0.546 ± 0.032	0.790 ± 0.002	0.867 ± 0.003	0.867 ± 0.000	0.882 ± 0.000	0.865 ± 0.002	**0.911 ± 0.002**	0.897 ± 0.002
	#C	1550.5 ± 23.3834	1930.5 ± 36.934	1902 ± 1.000	1976 ± 0.000	1923 ± 0.000	**2000 ± 0.000**	1955 ± 0.571	1947 ± 2.8
DBLP	Conductance	0.159 ± 0.003	0.182 ± 0.001	0.356 ± 0.000	0.183 ± 0.001	0.223 ± 0.000	0.186 ± 0.003	0.178 ± 0.001	**0.149± 0.004**
	Modularity	0.753 ± 0.003	0.762 ± 0.002	0.592 ± 0.000	0.754 ± 0.001	0.674 ± 0.000	0.783 ± 0.001	**0.838± 0.001**	0.816 ± 0.008
	#C	2183 ± 42.426	2852.6 ± 2.88	2225 ± 0.000	3475 ± 2.983	2387 ± 0.000	2000 ± 0.000	**4357.2 ± 241.01**	3173.2 ± 55.91
Google	Conductance	0.194 ± 0.000	0.358 ± 0.020	0.412 ± 0.000	0.364 ± 0.018	0.348 ± 0.000	0.342 ± 0.000	0.135 ± 0.000	**0.015 ± 0.003**
	Modularity	0.724 ± 0.000	0.721 ± 0.002	0.519 ± 0.000	0.709 ± 0.007	0.532 ± 0.000	0.673 ± 0.000	0.812 ± 0.000	**0.962 ± 0.015**
	#C	2474 ± 0.000	1271.5 ± 21.90	872 ± 0.000	1460.5 ± 4.50	1145 ± 0.000	**5000 ± 0.000**	2842 ± 0.000	3280 ± 18.4
Youtube	Conductance	0.438 ± 0.000	0.575 ± 0.010	0.383 ± 0.000	0.381 ± 0.002	0.417 ± 0.000	0.482 ± 0.000	0.341 ± 0.000	**0.334 ± 0.011**
	Modularity	0.593 ± 0.000	0.441 ± 0.010	0.623 ± 0.000	0.580 ± 0.002	0.574 ± 0.000	0.489 ± 0.000	0.559 ± 0.000	**0.623 ± 0.019**
	#C	945 ± 0.000	640.5 ± 72.830	956 ± 0.000	1493 ± 30.0	1587 ± 0.000	**5000 ± 0.000**	1230 ± 0.000	1234 ± 15.5

Table 4: Experimental results for Conductance (*low value better*) and Modularity (*high value better*) of detected communities on real-world datasets. We see that HACD gives comparable performance to baseline. It is observed that all baseline methods yields 100% coverage with the exception of BigCLAM achieved 50-80% coverage only. Bold values in #C show the highest number of communities among methods.

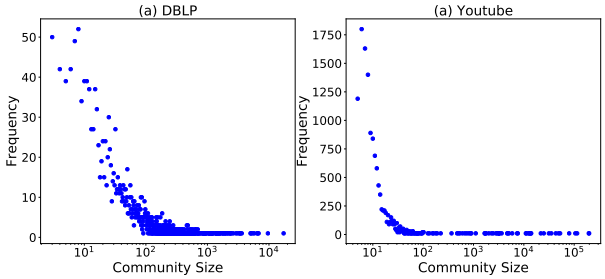

Figure 9: Communities size distribution obtained from HACD on DBLP and Youtube dataset. The results show that HACD is able to follow the power-law distribution.

Community[#3]	Community[#1099]	Community[#1131]
Jimeng Sun Jian Pei Bing Liu Bhavani M. Thuraisingham Longbing Cao Tanya Y. Berger Wolf Xindong Wu	B. Aditya Prakash Spiros Papadimitriou Abhay Harpale Stephan Gunnemann Danai Koutra Leman Akoglu Tao Li	Josh Attenberg Anitha Kannan Sreenivas Gollapudi Kamal Ali Sunandan Chakraborty Rui Cai
Wei Fan Philip S. Yu Charu C. Aggarwal	Tom M. Mitchell Tina Eliassi Rad Hanghang Tong	Panayiotis Tsaparas Ramakrishnan Srikant Lei Zhang
Jiawei Han	Christos Faloutsos	Rakesh Agrawal

Table 5: Top three communities (based on size) discovered by HACD on KDD Co-authorship network. The last row indicates authors that are detected at the top level and others rows indicate the people tightly connected with the associated authors found at the top level.

suggests that the graph has a natural community structure, with N communities, each corresponding to one clique, which appears to be the natural partition of the network. It turns out that baseline methods (ML, SLPA, OSLOM, BIGCLAM) merge multiple clique vertices at the final level, specifically in the pairwise network which has highly mixed nodes in each hierarchy. However, HACD is able to detect all N cliques in separate clusters as shown in Figure 10 below. In most cases, the numbers of communities obtained by our algorithms are the most accurate results, which are very close to the ground truth. The reason that our algorithms can overcome the resolution limit is that it combines the density (merging/splitting cluster) and the variant of modularity measure.

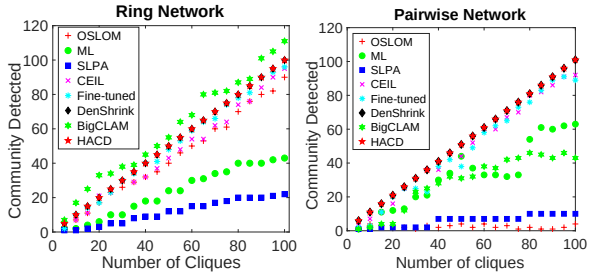

Figure 10: Test of the resolution limit. (a) The number of detected communities on ring network composed of identical cliques of different size connected by a single edge. (b) the number of detected communities on the pairwise network composed of identical cliques connected by a single edge with each other and another edge to the root node forming a complex network.

5.6 Scalability of HACD

HACD scales similarly to DenShrink and SLPA, better than Fine-tuned, OSLOM and BigCLAM, and it is slower than CEIL and ML baselines for increasing $\mathcal{V}$ while keeping the density of graph almost same Fig. 11. As expected, since HACD performs clustering via membership propagation on the detected communities, its run time increases faster than that of CEIL and ML. However, computational time is invested to enable its superior quality performance Tbl 4. It is worth noting that since it is a heuristic method, it is possible to parallelize the implementations, which can enable its feasible adoption for analysis of even larger graphs.

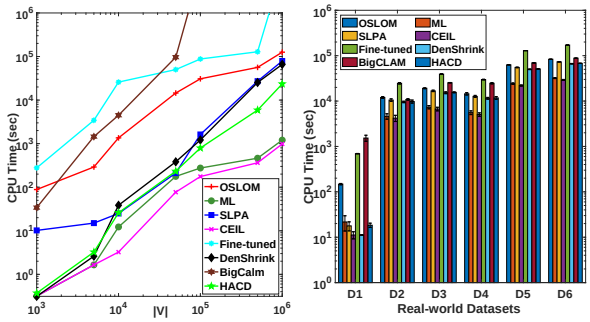

Figure 11: (a) Running time for HACD with varying network sizes. (b) Runtime of different algorithms on various datasets denoted on the x-axis as (D1) KDD, (D2) Gowalla, (D3) Stanford, (D4) DBLP, (D5) Google, and (D6) Youtube.

6 CONCLUSIONS

This work dealt with the identification of hierarchical communities via HACD, a bottom-to-up (agglomerative) detection framework. Specifically, a novel formulation of merit for a network was introduced and utilized in a recreating the networks, whose sub-network subsequently reveal the underlying communities. To provide the detected communities with desirable resolution, this algorithm was applied continuously in a bottom-to-up fashion, where the network is converted into sub-graph of communities per iteration. Empirical demonstrations show that HACD outperforms baseline techniques by up to 25% with respect to conductance and 10% with respect to modularity. It also shows that our proposed method is able to find local maximum modularity at the same time finds low conductance score in the network that dramatically improves the interpretability and quality of detected communities. In our future work, we will extend our method for finding overlapping communities for a single as well as multi-edge networks. Parallelization of HACD is among our future directions, through which memory as well as, computational requirements of the algorithm can be reduced.

REFERENCES

[1] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. 2008. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* 9, Sep (2008), 1981–2014.

[2] Zilong Bai, Peter Walker, and Ian Davidson. 2018. Mixtures of Block Models for Brain Networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 46–54.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[4] Deng Cai, Zheng Shao, Xiaofei He, Xifeng Yan, and Jiawei Han. 2005. Mining hidden community in heterogeneous social networks. In *Proceedings of the 3rd international workshop on Link discovery*. ACM, 58–65.

[5] Yulong Pei1 Nilanjan Chakraborty and Katia Sycara. 2015. Nonnegative matrix tri-factorization with graph regularization for community detection in social networks. In *AAAI*. AAAI Press, 2083–2089.

[6] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. 2014. Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems* 1, 1 (2014), 46–65.

[7] Thang N Dinh and My T Thai. 2013. Community detection in scale-free networks: approximation algorithms for maximizing modularity. *IEEE Journal on Selected Areas in Communications* 31, 6 (2013), 997–1006.

[8] Nicole Eikmeier and David F Gleich. 2017. Revisiting Power-law Distributions in Spectra of Real World Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 817–826.

[9] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[10] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[11] Vladimir Gligorijevic, Yannis Panagakis, and Stefanos P Zafeiriou. 2018. Non-Negative Matrix Factorizations for Multiplex Network Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[12] Andrey Goder and Vladimir Filkov. 2008. Consensus clustering algorithms: Comparison and refinement. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*. Society for Industrial and Applied Mathematics, 109–117.

[13] Alexander Gorovits, Ekta Gujral, Evangelos E Papalexakis, and Petko Bogdanov. 2018. LARC: Learning Activity-Regularized Overlapping Communities Across Time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1465–1474.

[14] Steve Gregory. 2010. Finding overlapping communities in networks by label propagation. *New Journal of Physics* 12, 10 (2010), 103018.

[15] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[16] Qiong Gui, Rui Deng, Pengfei Xue, and Xiaohui Cheng. 2018. A community discovery algorithm based on boundary nodes and label propagation. *Pattern Recognition Letters* 109 (2018), 103–109.

[17] Ekta Gujral and Evangelos E Papalexakis. 2018. SMACD: Semi-supervised Multi-Aspect Community Detection. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 702–710.

[18] Manish Gupta, Jing Gao, and Jiawei Han. 2013. Community distribution outlier detection in heterogeneous information networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 557–573.

[19] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks* 5, 2 (1983), 109–137.

[20] Jianbin Huang, Heli Sun, Jiawei Han, and Boqin Feng. 2011. Density-based shrinkage for revealing hierarchical and overlapping community structure in networks. *Physica A: Statistical Mechanics and its Applications* 390, 11 (2011), 2160–2171.

[21] Guanbo Jia, Zixing Cai, Mirco Musolesi, Yong Wang, Dan A Tennant, Ralf JM Weber, John K Heath, and Shan He. 2012. Community detection in social and biological networks using differential evolution. In *Learning and Intelligent Optimization*. Springer, 71–85.

[22] Varun Jog and Po-Ling Loh. 2015. Information-theoretic bounds for exact recovery in weighted stochastic block models using the Renyi divergence. *arXiv preprint arXiv:1509.06418* (2015).

[23] Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical review E* 83, 1 (2011), 016107.

[24] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. 2013. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences* 110, 52 (2013), 20935–20940.

[25] Andrea Lancichinetti and Santo Fortunato. 2012. Consensus clustering in complex networks. *Scientific reports* 2 (2012), 336.

[26] Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.). MIT Press, 556–562. http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[27] Elizabeth A Leicht and Mark EJ Newman. 2008. Community structure in directed networks. *Physical review letters* 100, 11 (2008), 118703.

[28] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

[29] Ulrike V Luxburg, Olivier Bousquet, and Mikhail Belkin. 2005. Limits of spectral clustering. In *Advances in neural information processing systems*. 857–864.

[30] Boaz Nadler and Meirav Galun. 2007. Fundamental limitations of spectral clustering. In *Advances in neural information processing systems*. 1017–1024.

[31] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104.

[32] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (2005), 814.

[33] Evangelos E Papalexakis, Leman Akoglu, and Dino Ienco. Do more views of a graph help? Community detection and clustering in multi-graphs. Citeseer.

[34] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[35] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.

[36] Thomas Richardson, Peter J Mucha, and Mason A Porter. 2009. Spectral tripartitioning of networks. *Physical Review E* 80, 3 (2009), 036111.

[37] Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. 2009. The map equation. *The European Physical Journal Special Topics* 178, 1 (2009), 13–23.

[38] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 105, 4 (2008), 1118–1123.

[39] Jianhua Ruan and Weixiong Zhang. 2007. An efficient spectral algorithm for network community discovery and its applications to biological and social networks. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. IEEE, 643–648.

[40] Vishnu Sankar, Balaraman Ravindran, and S Shivashankar. 2015. Ceil: A scalable, resolution limit free approach for detecting communities in large networks. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[41] Junming Shao, Zhichao Han, and Qinli Yang. 2014. Community detection via local dynamic interaction. *arXiv preprint arXiv:1409.7978* (2014).

[42] Junming Shao, Qinli Yang, Jinhu Liu, and Stefan Kramer. 2016. Graph Clustering with Density-Cut. *arXiv preprint arXiv:1606.00950* (2016).

[43] Fatemeh Sheikholeslami and Georgios G Giannakis. 2018. Robust Overlapping Community Detection via Constrained Egonet Tensor Decomposition. (2018).

[44] Scott White and Padhraic Smyth. 2005. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 274–285.

[45] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. 2013. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)* 45, 4 (2013), 43.

[46] Jierui Xie and B. K. Szymanski. 2013. LabelRank: A Stabilized Label Propagation Algorithm for Community Detection in Networks. In *Proc. IEEE Network Science Workshop*. West Point, NY.

[47] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 587–596.

[48] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

[49] Yang Yang, Yizhou Sun, Saurav Pandit, Nitesh V Chawla, and Jiawei Han. 2011. Is objective function the silver bullet? A case study of community detection algorithms on social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE, 394–397.

[50] Se-Young Yun and Alexandre Proutiere. 2016. Optimal cluster recovery in the labeled stochastic block model. In *Advances in Neural Information Processing Systems*. 965–973.