Overlapping Community Detection Using Neighborhood-Inflated Seed Expansion

Joyce Jiyoung Whang, *Member, IEEE*, David F. Gleich, and Inderjit S. Dhillon, *Fellow, IEEE*

Abstract—Community detection is an important task in network analysis. A community (also referred to as a cluster) is a set of cohesive vertices that have more connections inside the set than outside. In many social and information networks, these communities naturally overlap. For instance, in a social network, each vertex in a graph corresponds to an individual who usually participates in multiple communities. In this paper, we propose an efficient overlapping community detection algorithm using a seed expansion approach. The key idea of our algorithm is to find good seeds, and then greedily expand these seeds based on a community metric. Within this seed expansion method, we investigate the problem of how to determine good seed nodes in a graph. In particular, we develop new seeding strategies for a personalized PageRank clustering scheme that optimizes the conductance community score. An important step in our method is the neighborhood inflation step where seeds are modified to represent their entire vertex neighborhood. Experimental results show that our seed expansion algorithm outperforms other state-of-the-art overlapping community detection methods in terms of producing cohesive clusters and identifying ground-truth communities. We also show that our new seeding strategies are better than existing strategies, and are thus effective in finding good overlapping communities in real-world networks.

Index Terms—Community detection, clustering, overlapping communities, seed expansion, seeds, personalized PageRank

1 Introduction

Community detection is one of the most important and fundamental tasks in network analysis with applications in functional prediction in biology [1] and sub-market identification [2] among others. Given a network, a community is defined to be a set of cohesive nodes that have more connections inside the set than outside. Since a network can be modelled as a graph with vertices and edges, community detection can be thought of as a graph clustering problem where each community corresponds to a cluster in the graph. In this manuscript, the terms *cluster* and *community* are used interchangeably.

The goal of traditional, exhaustive graph clustering algorithms (e.g., Metis [3], Graclus [4]) is to partition a graph such that every node belongs to exactly one cluster. However, in many social and information networks, nodes participate in multiple communities. For instance, in a social network, nodes represent individuals and edges represent social interactions between the individuals. In this setting, a node's communities can be interpreted as its social circles. Thus, it is likely that a node belongs to multiple communities, i.e., communities naturally overlap. To find these groups, we study the problem of overlapping community detection where communities are allowed to overlap with each other and some nodes are allowed not to belong to any cluster.

- *J.J. Whang is with the Department of Computer Engineering, Sungkyunkwan University, Suwon, South Korea. E-mail: jjwhang@skku.edu.*
- *D.F. Gleich is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907-2107. E-mail: dgleich@purdue.edu.*
- *I.S. Dhillon is with the Department of Computer Science, University of Texas at Austin, Austin, TX 78712-1757. E-mail: inderjit@cs.utexas.edu.*

Manuscript received 24 Mar. 2015; revised 19 Oct. 2015; accepted 25 Dec. 2015. Date of publication 18 Jan. 2016; date of current version 30 Mar. 2016. Recommended for acceptance by H. Zha.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2518687

The main contribution of our paper is a new overlapping community detection algorithm with performance that greatly exceeds the state-of-the-art. This contribution was accomplished by studying new ideas in the prototypical "seed-and-grow" meta-algorithm for overlapping communities. We study each step of the overall computational pipeline in detail on real-world networks to demonstrate the utility of each component of the algorithm. Our experimental results show that our overlapping community detection algorithm significantly outperforms other methods in terms of run time, cohesiveness of communities, and ground-truth accuracy.

These local seed expansion methods are among the most successful strategies for overlapping community detection [5]. However, principled methods to choose the seeds are few and far between. When they exist, they are usually computationally expensive (e.g., using maximal cliques as seeds [6]). Empirically successful strategies include exhaustively exploring all individual seeds and greedy methods that randomly pick a vertex, grow a cluster, and continue with any unassigned vertex.

To find a set of good seeds, we present two effective seeding strategies that we call "Graclus centers" and "Spread hubs." The "Graclus centers" seeding is based on the same distance kernel that underlies the equivalence between kernel k-means and graph clustering objectives [4]. Using this distance function, we can efficiently locate a good seed *within* an existing set of cohesive vertices of the graph. Specifically, we first compute many clusters using a multi-level weighted kernel k-means algorithm on the graph (the Graclus algorithm) [4], then use the corresponding distance function to compute the "centroid vertex" of each cluster. We use the neighborhood set of each centroid vertex as a seed region for community detection. The idea of "Spread hubs" seeding is to select an independent set of high degree vertices. This seeding strategy is inspired by recent observations that there should be

good clusters around high degree vertices in real-world networks with a power-law degree distribution [7], [8].

The algorithm we use to grow a seed set is based on personalized PageRank (PPR) clustering [9]. The high level idea of this expansion method is to first compute the PPR vector for each of the seeds, and then expand each seed based on the PPR score. It is important to note that we can have multiple nodes in the personalization vector, and indeed we use the entire vertex neighborhood of a seed node as the personalization vector for PPR. This *neighborhood inflation* plays a critical role in the success of our algorithm. The full algorithm to compute overlapping clusters from the seeds is discussed in Section 3. We name our algorithm NISE by abbreviating our main idea, Neighborhood-Inflated Seed Expansion.

Our experimental results show that our seeding strategies are better than existing seeding strategies, and effective in finding good overlapping communities in real-world networks. More importantly, we observe that NISE significantly outperforms other state-of-the-art overlapping community detection methods in terms of producing cohesive clusters and identifying ground-truth communities. Also, our method scales to problems with over 45 million edges, whereas other existing methods were unable to complete on these large datasets.

2 Preliminaries

We formally describe the overlapping community detection problem, and review some important concepts in graph clustering. Also, we introduce real-world networks which are used in our experiments.

2.1 Problem Statement

Given a graph $G = (\mathcal{V}, \mathcal{E})$ with a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$, we can represent the graph as an adjacency matrix A such that $A_{ij} = e_{ij}$ where e_{ij} is the edge weight between vertices i and j, or $A_{ij} = 0$ if there is no edge. We assume that graphs are undirected, i.e., A is symmetric. The goal of the traditional, exhaustive graph clustering problem is to partition a graph into k pairwise disjoint clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_k = \mathcal{V}$. On the other hand, the goal of the overlapping community detection problem is to find overlapping clusters whose union is not necessarily equal to the entire vertex set $\mathcal{V}$. Formally, we seek k overlapping clusters such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_k \subseteq \mathcal{V}$.

2.2 Measures of Cluster Quality

There are some popular measures for gauging the quality of clusters: cut, normalized cut, and conductance. Let us define $\texttt{links}(\mathcal{C}_p, \mathcal{C}_q)$ to be the sum of edge weights between vertex sets $\mathcal{C}_p$ and $\mathcal{C}_q$.

Cut. The cut of cluster $\mathcal{C}_i$ is defined as the sum of edge weights between $\mathcal{C}_i$ and its complement, $\mathcal{V}\backslash\mathcal{C}_i$:

$$\texttt{cut}(\mathcal{C}_i) = \texttt{links}(\mathcal{C}_i, \mathcal{V}\backslash\mathcal{C}_i). \tag{1}$$

Normalized Cut. The normalized cut of a cluster is defined by the cut with volume normalization as follows:

$$\texttt{ncut}(\mathcal{C}_i) = \frac{\texttt{cut}(\mathcal{C}_i)}{\texttt{links}(\mathcal{C}_i, \mathcal{V})}. \tag{2}$$

Conductance. The conductance of a cluster is defined to be the cut divided by the least number of edges incident on either set $\mathcal{C}_i$ or $\mathcal{V}\backslash\mathcal{C}_i$:

$$\texttt{cond}(\mathcal{C}_i) = \frac{\texttt{cut}(\mathcal{C}_i)}{\min\big(\texttt{links}(\mathcal{C}_i, \mathcal{V}), \texttt{links}(\mathcal{V}\backslash\mathcal{C}_i, \mathcal{V})\big)}.$$

By definition, $\texttt{cond}(\mathcal{C}_i) = \texttt{cond}(\mathcal{V}\backslash\mathcal{C}_i)$. The conductance of a cluster is the probability of leaving that cluster by a one-hop walk starting from the smaller set between $\mathcal{C}_i$ and $\mathcal{V}\backslash\mathcal{C}_i$. Notice that $\texttt{cond}(\mathcal{C}_i)$ is always greater than or equal to $\texttt{ncut}(\mathcal{C}_i)$.

2.3 Graph Clustering and Weighted Kernel k-means

The normalized cut objective of a graph G is defined:

$$\texttt{ncut}(G) = \min_{\mathcal{C}_1,\ldots,\mathcal{C}_k} \sum_{i=1}^{k} \frac{\texttt{links}(\mathcal{C}_i, \mathcal{V}\backslash\mathcal{C}_i)}{\texttt{links}(\mathcal{C}_i, \mathcal{V})}. \tag{3}$$

This objective is equivalent to a weighted kernel k-means objective with the weight of each data point set to the degree of a vertex, and the kernel matrix to be $K = \sigma D^{-1} + D^{-1}AD^{-1}$, where D is the diagonal matrix of degrees (i.e., $D_{ii} = \sum_{j=1}^{n} A_{ij}$ where n is the total number of nodes), and σ is a scalar typically chosen to make K positive-definite [4]. Then, we can quantify the kernel distance between a vertex $v \in \mathcal{C}_i$ and cluster $\mathcal{C}_i$, denoted $\texttt{dist}(v, \mathcal{C}_i)$, as follows:

$$\texttt{dist}(v, \mathcal{C}_i) = -\frac{2\texttt{links}(v, \mathcal{C}_i)}{\texttt{deg}(v)\texttt{deg}(\mathcal{C}_i)} + \frac{\texttt{links}(\mathcal{C}_i, \mathcal{C}_i)}{\texttt{deg}(\mathcal{C}_i)^2} + \frac{\sigma}{\texttt{deg}(v)} - \frac{\sigma}{\texttt{deg}(\mathcal{C}_i)} \tag{4}$$

where $\texttt{deg}(v) = \texttt{links}(v, \mathcal{V})$, and $\texttt{deg}(\mathcal{C}_i) = \texttt{links}(\mathcal{C}_i, \mathcal{V})$.

2.4 Datasets

We use ten different real-world networks including collaboration networks, social networks, and a product network from [10], [12], and [11]. The networks are presented in Table 1. All the networks are loop-less, connected, undirected graphs.

In a collaboration network, vertices indicate authors, and edges indicate co-authorship. If authors u and v wrote a paper together, there exists an edge between them. For example, if a paper is written by four authors, this is represented by a clique of size four in the network. HepPh, AstroPh, and CondMat networks are constructed based on the papers submitted to arXiv e-print service. The DBLP network is constructed based on the DBLP computer science bibliography website.

We use five social networks: Flickr, Myspace, LiveJournal, LiveJournal2 (a variation with ground-truth), and Orkut. Flickr is an online photo sharing application, Myspace is a social entertainment networking service, LiveJournal is a blogging application where users can publish their own journals, and Orkut was a social networking website operated by Google.

In the Amazon product network, vertices represent products and edges represent co-purchasing information. If products u and v are frequently co-purchased, there exists an edge between them. This network is constructed based on *Customers Who Bought This Item Also Bought* feature of the Amazon website.

TABLE 1
Summary of Real-World Networks

Category	Graph	No. of vertices	No. of edges	Max. Deg.	Avg. Deg.	Avg. CC	Ground-truth	Source
Collaboration	HepPh	11,204	117,619	491	21.0	0.6216	N/A	[10]
	AstroPh	17,903	196,972	504	22.0	0.6328	N/A	[10]
	CondMat	21,363	91,286	279	8.5	0.6417	N/A	[10]
	DBLP	317,080	1,049,866	343	6.6	0.6324	✓	[10]
Product	Amazon	334,863	925,872	549	5.5	0.3967	✓	[10]
Social	Orkut	731,332	21,992,171	6,933	60.1	0.2468	✓	[10]
	Flickr	1,994,422	21,445,057	27,908	21.5	0.1881	N/A	[11]
	Myspace	2,086,141	45,459,079	92,821	43.6	0.1242	N/A	[12]
	LiveJournal	1,757,326	42,183,338	29,771	48.0	0.2400	N/A	[12]
	LiveJournal2	1,143,395	16,880,773	11,495	29.5	0.2535	✓	[10]

In Table 1, we present the number of nodes/edges, the maximum degree, the average degree, and the average clustering coefficient (CC) of each of the networks. Fig. 1 shows the degree distributions of DBLP, Flicker and Amazon networks. We can see that the real-world networks have distinguishing characteristics: a power-law degree distribution [13] and a high clustering coefficient [14], [15].

As indicated in Table 1, we have ground-truth communities [10] on some of the datasets. In DBLP, each publication venue (i.e., journal or conference) can be considered as an individual ground-truth community. In the Amazon network, each ground-truth community can be defined to be a product category that Amazon provides. In LiveJournal2 and Orkut networks, there exists user-defined social groups. On LiveJournal2 and Orkut networks, the ground-truth communities do not cover a substantial portion of the graph, so we use a subgraph which is induced by the nodes that have at least one membership in the ground-truth communities. In Table 1, the statistics about LiveJournal2 and Orkut are based on the induced subgraphs we used in our experiments.

3 Overlapping Community Detection Using Neighborhood-Inflated Seed Expansion

We introduce our overlapping community detection algorithm, NISE which consists of four phases: filtering, seeding, seed expansion, and propagation. In the filtering phase, we remove regions of the graph that are trivially separable from the rest of the graph. In the seeding phase, we find good seeds in the filtered graph, and in the seed expansion phase, we expand the seeds using a personalized PageRank clustering scheme. Finally, in the propagation phase, we further expand the communities to the regions that were removed in the filtering phase. Fig. 2 shows the overview of the NISE algorithm.

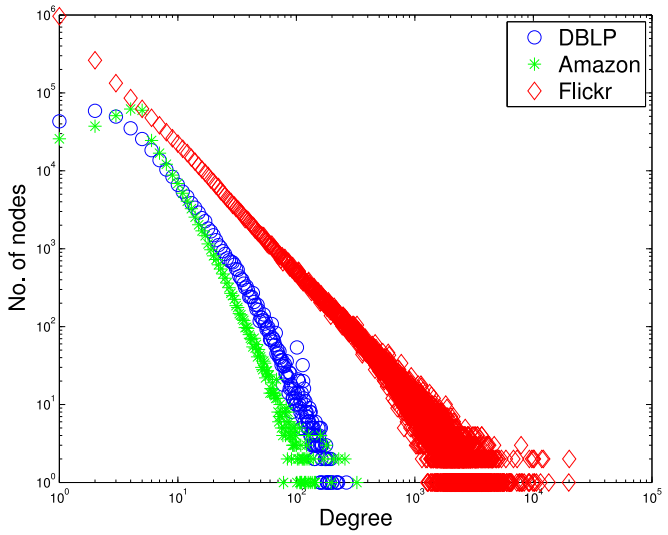


Fig. 1. Degree distributions of real-world networks – the degree distributions follow a power-law.

3.1 Filtering Phase

The goal of the filtering phase is to identify regions of the graph where an algorithmic solution is required to identify the overlapping clusters. To explain our filtering step, recall that almost all graph partitioning methods begin by assigning each connected component to a separate partition. Any other choice of partitioning for disconnected components is entirely arbitrary. The Metis procedure [3], for instance, may combine two disconnected components into a single partition in order to satisfy a balance constraint on the partitioning. For the problem of overlapping clustering, an analogous concept can be derived from biconnected components. We now review a series of definitions and formalizations of these ideas in order to analyze our filtering phase and prove new theorems about our propagation phase in Section 3.4. A biconnected component is defined as follows:

Definition 1. *Given a graph $G = (\mathcal{V}, \mathcal{E})$, a biconnected component is a maximal induced subgraph $G' = (\mathcal{V}', \mathcal{E}')$ that remains connected after removing any vertex and its adjacent edges in G'.*

Let us define the size of a biconnected component to be the number of edges in G'. Now, consider all the biconnected components of size one. Notice that there should be no overlapping partitions that use these edges because they bridge disjoint communities. Consequently, our filtering procedure is to find the largest connected component of the graph after we remove all single-edge biconnected components. We call this the "biconnected core" of the graph even though it may not be biconnected. Let $\mathcal{E}_S$ denote all the single-edge biconnected components. Then, the biconnected core graph is defined as follows:

Definition 2. *The biconnected core $G_C = (\mathcal{V}_C, \mathcal{E}_C)$ is the maximum size connected subgraph of $G'' = (\mathcal{V}, \mathcal{E} \setminus \mathcal{E}_S)$.*

Subgraphs connected to the biconnected core are called *whiskers* by Leskovec et al. [16] and we use the concept of a bridge to define them:

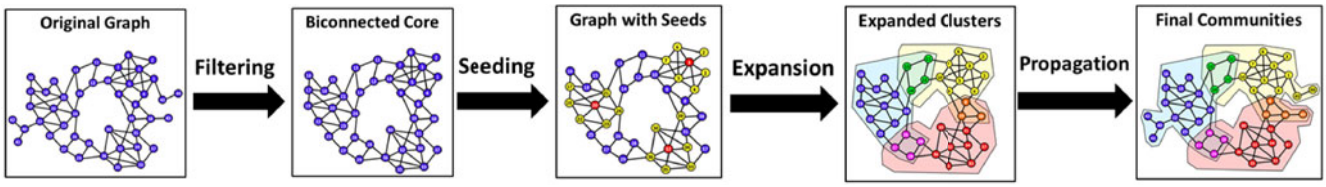


Fig. 2. Overview of NISE. NISE consists of four main phases: Filtering, seeding, seed expansion, and propagation.

Definition 3. *A bridge is a biconnected component of size one which is directly connected to the biconnected core.*

Whiskers are then defined as follows:

Definition 4. *A whisker $W = (\mathcal{V}_W, \mathcal{E}_W)$ is a maximal subgraph of G that can be detached from the biconnected core by removing a bridge.*

Let $\mathcal{E}_B$ be all the bridges in a graph. Notice that $\mathcal{E}_B \subseteq \mathcal{E}_S$. On the region which is not included in the biconnected core graph G_C, we define the detached graph G_D as follows:

Definition 5. *$G_D = (\mathcal{V}_D, \mathcal{E}_D)$ is the subgraph of G which is induced by $\mathcal{V} \setminus \mathcal{V}_C$.*

Finally, given the original graph $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ and $\mathcal{E}$ can be decomposed as follows:

Proposition 1. *Given a graph $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \mathcal{V}_C \cup \mathcal{V}_D$ and $\mathcal{E} = \mathcal{E}_C \cup \mathcal{E}_D \cup \mathcal{E}_B$.*

Proof. This follows from the definitions of the biconnected core, bridges, and the detached graph. □

Fig. 3 illustrates the biconnected core, whiskers, and bridges. The output of our filtering phase is the biconnected core graph where whiskers are filtered out (we remove regions that are clearly partitionable from the remainder). Note that there is no overlap between any of the whiskers. This indicates that there is no need to apply the overlapping community detection algorithm on the detached regions.

Table 2 shows the size of the biconnected core and the connectivity of the detached graph in our real-world networks. Details of these networks are presented in Table 1. We compute the size of the biconnected core in terms of the number of vertices and edges. The number reported in the parenthesis shows how many vertices or edges are included in the biconnected core, i.e., the percentages of $|\mathcal{V}_C|/|\mathcal{V}|$ and $|\mathcal{E}_C|/|\mathcal{E}|$, respectively. We also compute the number of connected components in the detached graph, and the size of the largest connected component (LCC in Table 2) in terms of the number of vertices. The number reported in the parenthesis indicates the relative size of the largest connected component compared to the number of vertices in the original graph.

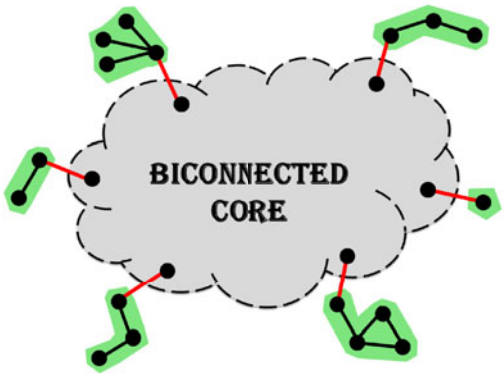


Fig. 3. Biconnected core, whiskers, and bridges – gray region indicates the biconnected core where vertices are densely connected to each other, and green components indicate whiskers. Red edges indicate bridges which connect the biconnected core and the whiskers.

We can see that the biconnected core contains a substantial portion of the edges. In terms of the vertices, the biconnected core contains around 80 or 90 percent of the vertices for all datasets except Flickr. In Flickr, the biconnected core only contains around 50 percent of the vertices while it contains 95 percent of edges. This indicates that the biconnected core is dense while the detached graph is quite sparse. Recall that the biconnected core is one connected component. On the other hand, in the detached graph, there are many connected components, which implies that the vertices in the detached graph are likely to be disconnected with each other. Notice that each connected component in the detached graph corresponds to a whisker. So, the largest connected component can be interpreted as the largest whisker. Based on the statistics of the detached graph, we can see that whiskers tend to be separable from each other, and there are no significant size whiskers. Also, the gap between the sizes of the biconnected core and the largest whisker is significant. All these statistics and observations support that our filtering phase creates a reasonable and more tractable input for an overlapping community detection algorithm.

3.2 Seeding Phase

Once we obtain the biconnected core graph, we find seeds in this filtered graph. The goal of an effective seeding strategy is to identify a diversity of vertices, each of which lies within a cluster of good conductance. This identification should not be too computationally expensive.

Graclus Centers. One way to achieve these goals is to first apply a high quality and fast graph partitioning scheme (disjoint clustering of vertices in a graph) in order to compute a collection of sets with fairly small conductance. Then, we select a set of seeds by picking the most central vertex from each set (cluster). The idea here is roughly that we want something that is close to the partitioning – which ought to be good – but that allows overlap to produce better boundaries between the partitions.

See Algorithm 1 for the full procedure. In practice, we perform top-down hierarchical clustering using Graclus [4] to get a large number of clusters. Then, we take the center of each cluster as a seed – the center of a cluster is defined to be the vertex that is closest to the cluster centroid (as discussed in Section 2.3, we can quantify the distance between a vertex and a cluster centroid by using the kernel that

TABLE 2
Biconnected Core and the Detached Graph (in the Last Column, LCC Refers to the Largest Connected Component)

	Biconnected core		Detached graph	
	No. of vertices (%)	No. of edges (%)	No. of components	Size of the LCC (%)
HepPh	9,945 (88.8%)	116,099 (98.7%)	1,123	21 (0.1874%)
AstroPh	16,829 (94.0%)	195,835 (99.4%)	957	23 (0.1285%)
CondMat	19,378 (90.7%)	89,128 (97.6%)	1,669	12 (0.0562%)
DBLP	264,341 (83.4%)	991,125 (94.4%)	43,093	32 (0.0101%)
Amazon	291,449 (87.0%)	862,836 (93.2%)	25,835	250 (0.0747%)
Flickr	954,672 (47.9%)	20,390,649 (95.1%)	864,628	107 (0.0054%)
Myspace	1,724,184 (82.7%)	45,096,696 (99.2%)	332,596	32 (0.0015%)
LiveJournal	1,650,851 (93.9%)	42,071,541 (99.7%)	101,038	105 (0.0060%)
LiveJournal2	1,076,499 (94.2%)	16,786,580 (99.4%)	59,877	91 (0.0080%)
Orkut	729,634 (99.8%)	21,990,221 (99.9%)	1,529	15 (0.0021%)

underlies the relationship between kernel k-means and graph clustering); see steps 5 and 7 in Algorithm 1. If there are several vertices whose distances are tied for the center of a cluster, we include all of them.

Algorithm 1. Seeding by Graclus Centers

Input: graph G, the number of seeds k.
Output: the seed set $\mathcal{S}$.
1: Compute exhaustive and non-overlapping clusters $\mathcal{C}_i$ ($i = 1, \ldots, k$) on G.
2: Initialize $\mathcal{S} = \emptyset$.
3: **for** each cluster $\mathcal{C}_i$ **do**
4: **for** each vertex $v \in \mathcal{C}_i$ **do**
5: Compute $\texttt{dist}(v, \mathcal{C}_i)$ using (4).
6: **end for**
7: $\mathcal{S} = \{\operatorname{argmin}_v \texttt{dist}(v, \mathcal{C}_i)\} \cup \mathcal{S}$.
8: **end for**

Spread Hubs. From another viewpoint, the goal is to select a set of well-distributed seeds in the graph, such that they will have high coverage after we expand the sets. We greedily choose an independent set of k points in the graph by looking at vertices in order of decreasing degree. For this heuristic, we draw inspiration from the distance function (4), which shows that the distance between a vertex and a cluster is inversely proportional to vertex's degree. Thus, high degree vertices are expected to have small distances to many other vertices. This also explains why we call the method *spread hubs*. It also follows from the recent results in [7], [8] which show that there should be good clusters around high degree vertices in power-law graphs with high clustering coefficients. We use an independent set in order to avoid picking seeds nearby each other.

Our full procedure is described in Algorithm 2. In the beginning, all the vertices are unmarked. Until k seeds are chosen, the following procedure is repeated: among unmarked vertices, the highest degree vertex is selected as a seed, and then the selected vertex and its neighbors are marked. As the algorithm proceeds exploring hubs in the network, if there are several vertices whose degrees are the same, we take an independent set of those that are unmarked. This step may result in more than k seeds, however, the final number of returned seeds typically does not exceed the input k too much because there usually are not too many high degree vertices.

Algorithm 2. Seeding by Spread Hubs

Input: graph $G = (\mathcal{V}, \mathcal{E})$, the number of seeds k.
Output: the seed set $\mathcal{S}$.
1: Initialize $\mathcal{S} = \emptyset$.
2: All vertices in $\mathcal{V}$ are unmarked.
3: **while** $|\mathcal{S}| < k$ **do**
4: Let $\mathcal{T}$ be the set of unmarked vertices with max degree.
5: **for each** $t \in \mathcal{T}$ **do**
6: **if** t is unmarked **then**
7: $\mathcal{S} = \{t\} \cup \mathcal{S}$.
8: Mark t and its neighbors.
9: **end if**
10: **end for**
11: **end while**

3.3 Seed Expansion Phase

Once we have a set of seed vertices, we wish to expand the clusters around those seeds. An effective technique for this task is using a personalized PageRank vector [17], also known as a random-walk with restart [18]. A personalized PageRank vector is the stationary distribution of a random walk that, with probability α follows a step of a random walk and with probability $(1 - \alpha)$ jumps back to a seed node. If there are multiple seed nodes, then the choice is usually uniformly random. Thus, nodes close by the seed are more likely to be visited. Recently, such techniques have been shown to produce communities that best match communities found in real-world networks [19]. In fact, personalized PageRank vectors have close relationships to graph cuts and clustering methods. Andersen et al. [9] show that a particular algorithm to compute a personalized PageRank vector, followed by a sweep over all cuts induced by the vector, will identify a set of good conductance within the graph. They prove this via a "localized Cheeger inequality" that states, informally, that the set identified via this procedure has a conductance that is not too far away from the best conductance of any set containing that vertex. Also, Mahoney et al. [20] show that personalized PageRank is, effectively, a seed-biased eigenvector of the Laplacian. They also show a limit to relate the personalized PageRank vectors to the Fiedler vector of a graph.

We briefly summarize the PPR-based seed expansion procedure in Algorithm 3 (each seed is expanded by this

Algorithm 3. Seed Expansion by PPR

Input: graph $G = (\mathcal{V}, \mathcal{E})$, a seed node $s \in \mathcal{S}$, PageRank link-following probability parameter $0 < \alpha < 1$, accuracy $\varepsilon > 0$
Output: low conductance set $\mathcal{C}$

1: Set $\mathcal{T} = \{s\} \cup \{\text{neighbors of s}\}$
2: Initialize $x_v = 0$ for $v \in \mathcal{V}$
3: Initialize $r_v = 0$ for $v \in \mathcal{V} \setminus \mathcal{T}$, $r_v = 1/|\mathcal{T}|$ for $v \in \mathcal{T}$
4: **while** any $r_v > \texttt{deg}(v)\varepsilon$ **do**
5: Update $x_v = x_v + (1-\alpha)r_v$.
6: For each $(v, u) \in \mathcal{E}$,
update $r_u = r_u + \alpha r_v/(2\,\texttt{deg}(v))$
7: Update $r_v = \alpha r_v/2$
8: **end while**
9: Sort vertices by decreasing $x_v/\texttt{deg}(v)$
10: For each prefix set of vertices in the sorted list, compute the conductance of that set and set $\mathcal{C}$ to be the set that achieves the minimum.

procedure). Please see Andersen et al. [9] for a full description of the algorithm. The high level idea of this expansion method is that given a set of restart nodes (denoted by $\mathcal{T}$ in Algorithm 3), we first compute the PPR vector, examine nodes in order of highest to lowest PPR score, and then return the set that achieves the minimum conductance.

It is important to note that we can have multiple nodes in $\mathcal{T}$ (which corresponds to nonzero elements in the personalization vector in PPR), and indeed we use the entire vertex neighborhood of a seed node as the restart nodes (see step 1 in Algorithm 3). Since we do not just use a singleton seed but also use its neighbors as the restart nodes in PPR, we call step 1 *neighborhood inflation*. We empirically observe that this neighborhood inflation plays a critical role in producing low conductance communities. See Section 5 for details. Recently, Gleich and Seshadhri [8] have provided some theoretical justification for why neighborhood-inflated seeds may outperform a singleton seed in PPR expansion on many real-world networks.

Steps 2-8 are closely related to a coordinate descent optimization procedure [21] on the PageRank linear system. Although it may not be apparent from the procedure, this algorithm is remarkably efficient when combined with appropriate data structures. The algorithm keeps two vectors of values for each vertex, $\mathbf{x}$ and $\mathbf{r}$. In a large graph, most of these values will remain zero on the vertices and hence, these need not be stored. Our implementation uses a hash table for the vectors $\mathbf{x}$ and $\mathbf{r}$. Consequently, the sorting step is only over a small fraction of the total vertices.

In the original PPR clustering [9], the PPR score is divided by the degree of each node (step 9) to remove bias towards high degree nodes. This step converts a PageRank vector, a left eigenvector of a Markov chain, into the right eigenvector of a Markov chain. Right eigenvectors are close relatives of the Fiedler vector of a graph, and so this degree normalization produces a vector that we call the *Fiedler Personalized PageRank vector* because of this relationship. Fiedler vectors also satisfy Cheeger inequalities, just like the Fiedler Personalized PageRank vectors. However, Kloumann and Kleinberg [22] recently reported that this degree normalization might slightly degrade the quality of the output clusters in terms of matching with ground-truth communities in some real-world networks. So, in our experiments, we also try

Algorithm 4. Propagation Procedure

Input: graph $G = (\mathcal{V}, \mathcal{E})$, biconnected core $G_C = (\mathcal{V}_C, \mathcal{E}_C)$, communities of $G_C : \mathcal{C}_i$ $(i = 1, ..., k) \in \mathcal{C}$.
Output: communities of G.

1: **for** each $\mathcal{C}_i \in \mathcal{C}$ **do**
2: Detect bridges $\mathcal{E}_{B_i}$ attached to $\mathcal{C}_i$.
3: **for** each $b_j \in \mathcal{E}_{B_i}$ **do**
4: Detect the whisker $w_j = (\mathcal{V}_j, \mathcal{E}_j)$ which is attached to b_j.
5: $\mathcal{C}_i = \mathcal{C}_i \cup \mathcal{V}_j$.
6: **end for**
7: **end for**

using the PPR score which we just call *PPR*. We compare the performance of the Fiedler PPR and PPR in Section 5.

In Algorithm 3, there are two parameters which are related to PPR computation: α and ε. We follow standard practice for PPR clustering on an undirected graph and set $\alpha = 0.99$ [16]. This value yields results that are similar to those without damping, yet have bounded computational time. The parameter ε is an accuracy parameter. As $\varepsilon \to 0$, the final vector solution $\mathbf{x}$ tends to the exact solution of the PageRank linear system. When used for clustering, however, this parameter controls the effective *size* of the final cluster. If ε is large (about 10^{-2}), then the output vector is inaccurate, incredibly sparse, and the resulting cluster is small. If ε is small, say 10^{-8}, then the PageRank vector is accurate, nearly dense, and the resulting cluster may be large. We thus run the PPR clustering scheme several times, with a range of accuracy parameters that are empirically designed to produce clusters with between 1 and 50,000 times the number of edges in the initial seed set (these values of ε are fixed and independent of the graph). The final community we select is the one with the best conductance score from these possibilities.

3.4 Propagation Phase

Once we get the personalized PageRank communities on the biconnected core, we further expand each of the communities to the regions detached in the filtering phase. Our assignment procedure is straightforward: for each detached whisker connected via a bridge, we add that piece to all of the clusters that utilize the other vertex in the bridge. This procedure is described in Algorithm 4. We show that our propagation procedure only improves the quality of the final clustering result in terms of the normalized cut metric. To do this, we need to fix some notation. Let $\mathcal{E}_{B_i}$ be a set of bridges which are attached to $\mathcal{C}_i$, and $W_{\mathcal{C}_i}$ be a set of whiskers which are attached to the bridges, i.e., $W_{\mathcal{C}_i} = (\mathcal{V}_{W_i}, \mathcal{E}_{W_i})$, where

$$w_j = (\mathcal{V}_j, \mathcal{E}_j) \in W_{\mathcal{C}_i}; \mathcal{V}_{W_i} = \bigcup_{w_j \in W_{\mathcal{C}_i}} \mathcal{V}_j; \text{ and } \mathcal{E}_{W_i} = \bigcup_{w_j \in W_{\mathcal{C}_i}} \mathcal{E}_j.$$

Finally, let $\mathcal{C}_i'$ denote the expanded $\mathcal{C}_i$, where $|\mathcal{C}_i'| \geq |\mathcal{C}_i|$. Equality holds in this expression when there is no bridge attached to $\mathcal{C}_i$. When we expand $\mathcal{C}_i$ using Algorithm 4, $\mathcal{C}_i'$ is equal to $\{\mathcal{C}_i \bigcup \mathcal{V}_{W_i}\}$. The following results show that we only decrease the (normalized) cut by adding the whiskers.

Theorem 1. *If a community $\mathcal{C}_i$ is expanded to $\mathcal{C}_i'$ using Algorithm 4,* $\texttt{cut}(\mathcal{C}_i') = \texttt{cut}(\mathcal{C}_i) - \texttt{links}(\mathcal{V}_{W_i}, \mathcal{C}_i)$.

TABLE 3
Average Normalized Cut Values Before & After Propagation

Graph	Before Propagation	After Propagation
HepPh	0.1383	0.1282
AstroPh	0.1764	0.1728
CondMat	0.1841	0.1717
DBLP	0.2329	0.2035
Amazon	0.1356	0.1159

Theorem 2 shows this should decrease.

Proof. Recall that $\texttt{cut}(\mathcal{C}_i)$ is defined as follows:

$$\begin{aligned}\texttt{cut}(\mathcal{C}_i) &= \texttt{links}(\mathcal{C}_i, \mathcal{V} \setminus \mathcal{C}_i).\\ &= \texttt{links}(\mathcal{C}_i, \mathcal{V}) - \texttt{links}(\mathcal{C}_i, \mathcal{C}_i).\end{aligned}$$

Let us first consider $\texttt{links}(\mathcal{C}'_i, \mathcal{V})$ as follows:

$$\begin{aligned}\texttt{links}(\mathcal{C}'_i, \mathcal{V}) = \texttt{links}(\mathcal{C}_i, \mathcal{V}) + \texttt{links}(\mathcal{V}_{W_i}, \mathcal{V})\\ - \texttt{links}(\mathcal{V}_{W_i}, \mathcal{C}_i).\end{aligned}$$

Notice that $\texttt{links}(\mathcal{V}_{W_i}, \mathcal{V}) = \texttt{links}(\mathcal{V}_{W_i}, \mathcal{V}_{W_i}) + \texttt{links}(\mathcal{V}_{W_i}, \mathcal{C}_i)$ by definition of whiskers. Thus, $\texttt{links}(\mathcal{C}'_i, \mathcal{V})$ can be expressed as follows:

$$\texttt{links}(\mathcal{C}'_i, \mathcal{V}) = \texttt{links}(\mathcal{C}_i, \mathcal{V}) + \texttt{links}(\mathcal{V}_{W_i}, \mathcal{V}_{W_i}). \quad (5)$$

On the other hand, $\texttt{links}(\mathcal{C}'_i, \mathcal{C}'_i)$ can be expressed as:

$$\begin{aligned}\texttt{links}(\mathcal{C}'_i, \mathcal{C}'_i) = \texttt{links}(\mathcal{V}_{W_i}, \mathcal{V}_{W_i}) + \texttt{links}(\mathcal{C}_i, \mathcal{C}_i)\\ + \texttt{links}(\mathcal{V}_{W_i}, \mathcal{C}_i).\end{aligned} \quad (6)$$

Now, let us compute $\texttt{cut}(\mathcal{C}'_i)$ which is defined by

$$\texttt{cut}(\mathcal{C}'_i) = \texttt{links}(\mathcal{C}'_i, \mathcal{V}) - \texttt{links}(\mathcal{C}'_i, \mathcal{C}'_i). \quad (7)$$

By rewriting (5) and (6), we can express $\texttt{cut}(\mathcal{C}'_i)$ as follows: $\texttt{cut}(\mathcal{C}'_i) = \texttt{cut}(\mathcal{C}_i) - \texttt{links}(\mathcal{V}_{W_i}, \mathcal{C}_i)$. □

Theorem 2. *If a community $\mathcal{C}_i$ is expanded to $\mathcal{C}'_i$ using Algorithm 4,* $\texttt{ncut}(\mathcal{C}'_i) \leq \texttt{ncut}(\mathcal{C}_i)$.

Proof. Recall that

$$\texttt{ncut}(\mathcal{C}_i) = \frac{\texttt{cut}(\mathcal{C}_i)}{\texttt{links}(\mathcal{C}_i, \mathcal{V})}.$$

On the other hand, by Theorem 1, we can represent $\texttt{ncut}(\mathcal{C}'_i)$ as follows:

$$\begin{aligned}\texttt{ncut}(\mathcal{C}'_i) &= \frac{\texttt{cut}(\mathcal{C}'_i)}{\texttt{links}(\mathcal{C}'_i, \mathcal{V})}.\\ &= \frac{\texttt{cut}(\mathcal{C}_i) - \texttt{links}(\mathcal{V}_{W_i}, \mathcal{C}_i)}{\texttt{links}(\mathcal{C}_i, \mathcal{V}) + \texttt{links}(\mathcal{V}_{W_i}, \mathcal{V}_{W_i})}.\end{aligned}$$

Therefore, $\texttt{ncut}(\mathcal{C}'_i) \leq \texttt{ncut}(\mathcal{C}_i)$. Equality holds when there is no bridge attached to $\mathcal{C}_i$, i.e., $\mathcal{E}_{B_i} = \emptyset$. □

Table 3 shows the average normalized cut values before and after the propagation phase. As predicted by the theorem, these values decrease after the propagation on this set of graphs.

TABLE 4
Time Complexity of Each Phase

Phase		Time complexity
Filtering		$O(\lvert\mathcal{V}\rvert + \lvert\mathcal{E}\rvert)$
Seeding	Graclus centers	$O(\lceil \log k \rceil (\lvert\mathcal{V}_C\rvert + \lvert\mathcal{E}_C\rvert))$
	Spread hubs	$O(\lvert\mathcal{V}_C\rvert)$
Seed expansion		$O(\sum_i^k \max_\varepsilon \texttt{links}(\mathcal{C}_i(\varepsilon), \mathcal{V}_C))$
Propagation		$O(\sum_i^k (\mathcal{E}_{B_i} + \mathcal{V}_{W_i} + \mathcal{E}_{W_i}))$

3.5 Time Complexity Analysis

We summarize the time complexity of our overall algorithm in Table 4. The filtering phase requires computing biconnected components in a graph, which takes $O(|\mathcal{V}| + |\mathcal{E}|)$ time. The complexity of "Graclus centers" seeding strategy is determined by the complexity of hierarchical clustering using Graclus. Recall that "Spread hubs" seeding strategy requires nodes to be sorted according to their degrees. Thus, the complexity of this strategy is bounded by the sorting operation (we can use a bucket sort). Expanding each seed requires solving multiple personalized PageRank clustering problems. The complexity of this operation is complicated to state compactly [9], but it scales with the output size of each cluster, $\texttt{links}(\mathcal{C}_i, \mathcal{V}_C)$. We do evaluate the seed expansion multiple times for the various values of ε. So the total runtime of this step involves a summation of all the cluster sizes for each ε. Practically, this is only a small amount larger than the largest cluster output by any step, so that $\max_\varepsilon \texttt{links}(\mathcal{C}_i(\varepsilon), \mathcal{V}_C)$ is a realistic estimate. Finally, our simple propagation procedure scans the regions that were not included in the biconnected core and attaches them to the final communities.

4 Related Work

For overlapping community detection, many different approaches have been proposed [5] including clique percolation, line graph partitioning, eigenvector methods, ego network analysis, and low-rank models. Clique percolation methods look for overlap between fixed size cliques in the graph [23]. Line graph partitioning is also known as link communities. Given a graph $G = (\mathcal{V}, \mathcal{E})$, the line graph of $L(G)$ (also called the dual graph) has a vertex for each edge in G and an edge whenever two edges (in G) share a vertex. For instance, the line graph of a star is a clique. A partitioning of the line graph induces an overlapping clustering in the original graph [24]. Even though these clique percolation and line graph partitioning methods are known to be useful for finding meaningful overlapping structures, these methods often fail to scale to large networks like those we consider.

Eigenvector methods generalize spectral methods and use a soft clustering scheme applied to eigenvectors of the normalized Laplacian or modularity matrix in order to estimate communities [25]. Ego network analysis methods use the theory of structural holes [26], and compute and combine many communities through manipulating ego networks [27], [28]. We compare against the Demon method [28] that uses this strategy. We also note that other low-rank methods such as non-negative matrix factorizations identify overlapping communities as well. We compare against the Bigclam method [29] that uses this approach.

TABLE 5
Returned Number of Clusters and Graph Coverage of Each Algorithm

Graph		oslom	demon	bigclam	nise-sph-fppr	nise-grc-fppr
HepPh	coverage (%)	100	88.83	84.37	100	100
	no. of clusters	608	5,147	100	99	90
AstroPh	coverage (%)	100	94.15	91.11	100	100
	no. of clusters	1,241	8,259	200	212	246
CondMat	coverage (%)	100	91.16	99.96	100	100
	no. of clusters	1,534	10,474	200	201	249
Flickr	coverage (%)	N/A	N/A	52.13	93.60	100
	no. of clusters	N/A	N/A	15,000	15,349	16,347
LiveJournal	coverage (%)	N/A	N/A	43.86	99.78	99.79
	no. of clusters	N/A	N/A	15,000	15,058	16,271
Myspace	coverage (%)	N/A	N/A	N/A	99.87	100
	no. of clusters	N/A	N/A	N/A	15,324	16,366
DBLP	coverage (%)	100	84.89	100	100	100
	no. of clusters	17,519	174,560	25,000	26,503	18,477
Amazon	coverage (%)	100	79.16	100	100	100
	no. of clusters	17,082	105,685	25,000	27,763	20,036
Orkut	coverage (%)	N/A	N/A	82.13	99.99	100
	no. of clusters	N/A	N/A	25,000	25,204	32,622
LiveJournal2	coverage (%)	N/A	N/A	56.64	99.95	99.99
	no. of clusters	N/A	N/A	25,000	25,065	32,274

The approach we employ is called local optimization and expansion [5]. Starting from a seed, such a method greedily expands a community around that seed until it reaches a local optima of the community detection objective. Determining how to seed a local expansion method is, arguably, a critical problem within these methods. Strategies to do so include using maximal cliques [6], prior information [30], or locally minimal neighborhoods [8]. The latter method was shown to identify the vast majority of good conductance sets in a graph; however, there was no provision made for total coverage of all vertices.

Different optimization objectives and expansion methods can be used in a local expansion method. For example, Oslom [31] tests the statistical significance of clusters with respect to a random configuration during community expansion. Starting from a randomly picked node, the Oslom method greedily expands the cluster by checking whether the expanded community is statistically significant or not, which results in detecting a set of overlapping clusters and outliers in a graph. We compare our method with the Oslom method in our experiments (see Section 5).

In our algorithm, we use a personalized PageRank based cut finder [9] for the local expansion method. Abrahao et al. [19] observe that the structure of real-world communities can be well captured by the random-walk-based algorithms, i.e., personalized PageRank clusters are topologically similar to real-world clusters. More recently, Kloumann and Kleinberg [22] propose to use pure PageRank scores instead of the Fiedler PageRank scores to get a higher accuracy in terms of matching with ground-truth communities.

A preliminary version of this work has appeared in [32]. In this paper, we provide technical details about neighborhood inflation in our seed expansion phase, and include additional experimental results to show the importance of the neighborhood inflation step. Also, we test and compare the performance of the Fiedler PageRank and the standard PPR in our expansion phase. We also improve the implementation of our algorithm in that we try expanding seeds in parallel using multiple threads.

5 Experimental Results

We compare our algorithm, NISE, with other state-of-the-art overlapping community detection methods: Bigclam [29], Demon [28], and Oslom [31]. For these three methods, we used the software which is provided by the authors of [28], [29], and [31] respectively. While Demon and Oslom only support a sequential execution, Bigclam supports a multithreaded execution. NISE is written in a mixture of C++ and MATLAB. In NISE, seeds can be expanded in parallel, and this feature is implemented using the parallel computing toolbox provided by MATLAB. We compare the performance of each of these methods on ten different real-world networks which are presented in Section 2.4. Within NISE, we also compare the performance of different seeding strategies and some variants of expansion methods. We use four different seeding strategies: "graclus centers" (denoted by "nise-grc-*") and "spread hubs" (denoted by "nise-sph-*") which are proposed in this manuscript, "locally minimal neighborhoods" (denoted by "nise-lcm-*") which has been proposed in [8], and random seeding strategy (denoted by "nise-rnd-*") where we randomly take k seeds. Andersen and Lang [2] have provided some theoretical justification for why random seeding also should be competitive. We also compare two different expansion methods: the Fiedler Personalized PageRank (denoted by "nise-*-fppr"), and the standard Personalized PageRank (denoted by "nise-*-ppr").

5.1 Graph Coverage and Community Sizes

We first report the returned number of clusters and the graph coverage of each algorithm in Table 5. The graph coverage indicates how many vertices are assigned to clusters (i.e., the number of assigned vertices divided by the total number of vertices in a graph). Note that we can control the number of seeds k in NISE and the number of clusters k in Bigclam. We set k (in our methods and Bigclam) as 100 for HepPh, 200 for AstroPh and CondMat, 15,000 for Flickr, Myspace, and LiveJournal, and 25,000 for DBLP, Amazon, LiveJournal2, and Orkut networks without any tuning and using the guidance

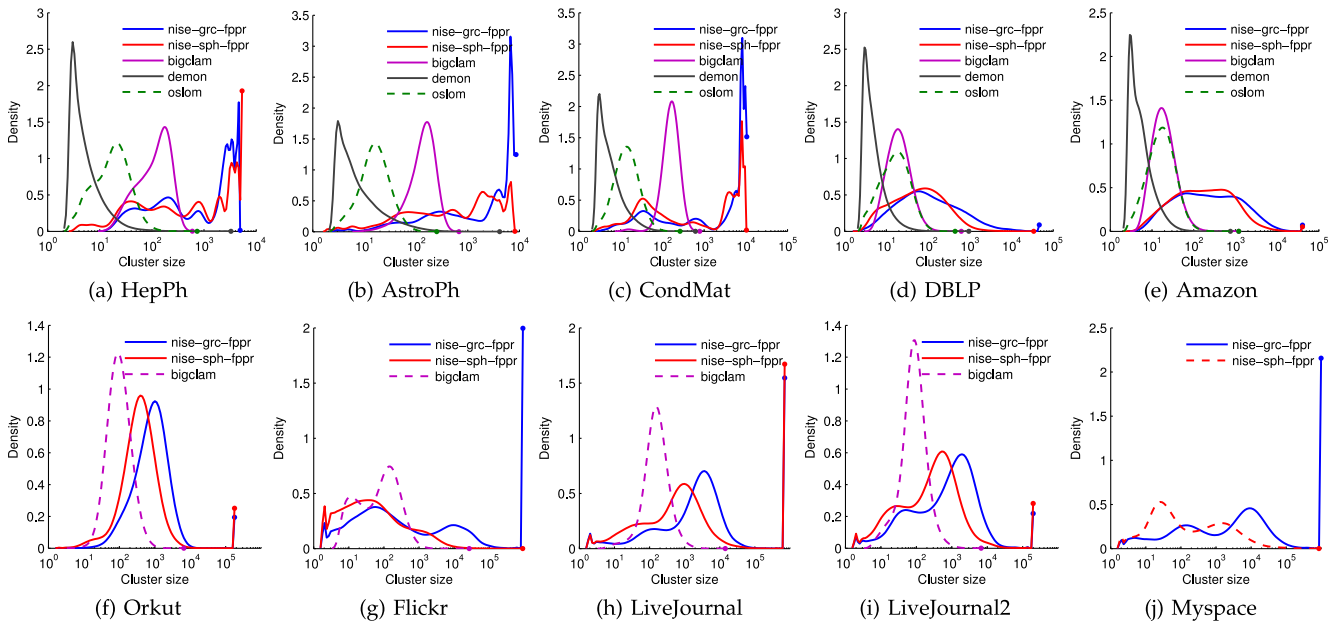


Fig. 4. Distributions of cluster sizes from the methods. These plots show a kernel density smoothed histogram of the cluster sizes from each method. The horizontal axis is the cluster size and the vertical axis is proportional to the number of clusters of that size.

that larger graphs can have more clusters. (Section 5.6 discusses varying k.) For the networks where we have ground-truth communities, we slightly overestimate the number of clusters k since there usually exists a large number of ground-truth communities. Since we remove duplicate clusters after the PageRank expansion in NISE, the returned number of clusters can be smaller than k. Also, since we choose all the tied seeds in "graclus centers" and "spread hubs", the returned number of clusters of these algorithms can be slightly larger than k. Recall that we use a top-down hierarchical clustering scheme in the "graclus centers" strategy. So, in this case, the returned number of clusters before filtering the duplicate clusters is slightly greater than or equal to $2^{\lceil \log k \rceil}$. Demon and Oslom determine the number of clusters based on datasets themselves, although these methods fail on Flickr, Myspace, LiveJournal, LiveJournal2, and Orkut. Bigclam does not finish on the Myspace network (using 4 threads) after running for one week.

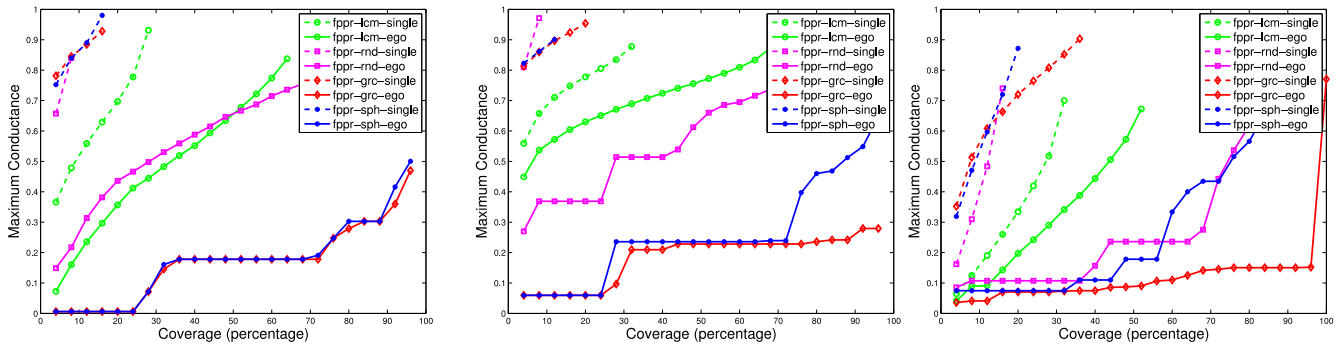
Fig. 4 shows distributions of cluster sizes. These figures show that the NISE method tends to find larger clusters than the other methods, usually about 10 to 100 times as large. Also, the NISE method often finds a number of large clusters—these are the spikes on the right for subfigures (f)-(j). This tends to happen slightly more often for the "graclus centers" seeding strategy. The other observation is that NISE tends to produce more variance in the sizes of the clusters than the other methods and the resulting histograms are not as sharply peaked.

5.2 Importance of Neighborhood-Inflation

We evaluate the quality of overlapping communities in terms of the maximum conductance of any cluster. A high quality algorithm should return a set of clusters that covers a large portion of the graph with small maximum conductance. This metric can be captured by a conductance-versus-coverage curve. That is, for each method, we first sort the clusters according to the conductance scores in ascending order, and then greedily take clusters until a certain percentage of the graph is covered. The x-axis of each plot is the graph coverage, and the y-axis is the maximum conductance value among the clusters we take. We can interpret this plot as follows: we need to use clusters whose conductance scores are less than or equal to y to cover x percentage of the graph. Note that lower conductance indicates better quality of clusters, i.e., a lower curve indicates better clusters.

First, we verify the importance of *neighborhood inflation* in our seed expansion phase. Recall that when we compute the personalized PageRank score for each seed node, we use the seed node's entire vertex neighborhood (the vertex neighborhood is also referred to as "ego network") as the restart region in PPR (details are in Section 3.3). To see how this affects the overall performance of the seed expansion method, we compare the performance of singleton seeds and neighborhood-inflated seeds. Fig. 5 shows the conductance-versus-coverage plot for singleton seeds and neighborhood-inflated seeds. "*-single" indicates singleton seeds, i.e., each seed is solely used as the restart region in PPR. "*-ego" indicates neighborhood-inflated seeds. We also use four different seeding strategies: "graclus centers" (denoted by "grc-*"), "spread hubs" (denoted by "sph-*"), "locally minimal neighborhoods" (denoted by "lcm-*"), and "random" (denoted by "rnd-*").

We can see that the performance significantly degrades when singleton seeds are used for all the seeding strategies. This implies that neighborhood inflation plays a critical role in the success of our method. Even though we only present the results on LiveJournal, Myspace, and Flickr in Fig. 5 for brevity, we consistently observed that neighborhood-inflated seeds are much better than singleton seeds on all other networks. We also notice that when neighborhood-inflated seeds are used, both "graclus centers" and "spread hubs" seeding strategies significantly outperform other seeding strategies. "spread hubs" and "graclus centers" seeding strategies produce similar results on LiveJournal whereas "graclus centers" is better than "spread hubs" on Myspace and Flickr. We use the conventional Fiedler PPR for the



(a) LiveJournal (b) Myspace (c) Flickr

Fig. 5. Importance of neighborhood inflation – there is a large performance gap between singleton seeds and neighborhood-inflated seeds for all the seeding strategies. Neighborhood inflation plays a critical role in the success of NISE. When neighborhood-inflated seeds are used, "graclus centers" and "spread hubs" seeding strategies significantly outperform other seeding strategies.

expansion phase in Fig. 5, but we also got the same conclusion using the standard PPR.

5.3 Community Quality Using Conductance, Modularity, and Association

We compare the performance of NISE with other state-of-the-art methods. Within NISE, we also compare four different seeding strategies and two different expansion methods. To evaluate the quality of communities, we use three metrics: conductance, modularity [33], and average association [34].[1] Lower conductance, higher modularity, and higher average association indicate better communities. Similar to how we draw the maximum conductance-versus-coverage plot, we draw minimum modularity-versus-coverage and minimum average association-versus-coverage plots. Then, we compute AUC (Area Under the Curve) of the metric-versus-coverage. For the conductance measure, the AUC scores are normalized such that they are between zero and one, and then, we report 1-AUC as the AUC score of the conductance metric in order to keep the property that *higher AUC is better*. Thus, in Table 6, higher AUC scores indicate better communities for all the three metrics.

Table 6 shows AUC scores on the six networks where we do not have ground-truth community information (see Table 1 for details about these networks). We can see several patterns in Table 6. First, NISE outperforms Demon, Oslom, and Bigclam. There is a significant performance gap between NISE and these methods. Second, within NISE, "graclus centers" and "spread hubs" seeding strategies outperform the other two seeding strategies. As a result, "nise-grc-fppr" or "nise-grc-ppr" show the best performance for all networks. Third, "fppr" leads to better conductance values than "ppr" whereas "ppr" leads to better average association values than "fppr".

We also compare NISE with a non-overlapping clustering to study the benefit of overlap. If we use the clusters produced by Graclus [4], a graph partitioning method which is used within our "graclus centers" seeding strategy, we observe that NISE also significantly outperforms Graclus in terms of all the three metrics on all the networks. For example, in terms of conductance measure, the AUC of Graclus is 0.4691 while the AUC of NISE is 0.8981. Due to the large performance gap between these clusterings, we omit more comprehensive numerical evaluation.

1. The modularity of an individual cluster $\mathcal{C}_i$ is defined as $(1/\texttt{links}(\mathcal{V},\mathcal{V}))(\texttt{links}(\mathcal{C}_i,\mathcal{C}_i) - \texttt{links}(\mathcal{C}_i,\mathcal{V})^2/\texttt{links}(\mathcal{V},\mathcal{V}))$, and the average association of $\mathcal{C}_i$ is defined as $\texttt{links}(\mathcal{C}_i,\mathcal{C}_i)/|\mathcal{C}_i|$.

5.4 Community Quality via Ground-truth

We have ground-truth communities for the DBLP, Amazon, LiveJournal2, and Orkut networks, thus, for these networks, we compare against the ground-truth communities. Given a set of algorithmic communities $\mathcal{C}$ and the ground-truth communities $\mathcal{S}$, we compute the F_1 measure and the F_2 measure to evaluate the relevance between the algorithmic communities and the ground-truth communities. In general, F_β measure is defined as follows:

$$F_\beta(\mathcal{S}_i) = (1+\beta^2)\frac{precision(\mathcal{S}_i)\cdot recall(\mathcal{S}_i)}{\beta^2\cdot precision(\mathcal{S}_i) + recall(\mathcal{S}_i)}$$

where β is a non-negative real value, and the $precision$ and $recall$ of $\mathcal{S}_i \in \mathcal{S}$ are defined as follows:

$$precision(\mathcal{S}_i) = \frac{|\mathcal{C}_j \bigcap \mathcal{S}_i|}{|\mathcal{C}_j|},$$

$$recall(\mathcal{S}_i) = \frac{|\mathcal{C}_j \bigcap \mathcal{S}_i|}{|\mathcal{S}_i|},$$

where $\mathcal{C}_j \in \mathcal{C}$, and $F_\beta(\mathcal{S}_i) = F_\beta(\mathcal{S}_i, \mathcal{C}_{j^*})$ where $j^* = \text{argmax}_j F_\beta(\mathcal{S}_i, \mathcal{C}_j)$. Then, the average F_β measure is defined to be

$$\bar{F_\beta} = \frac{1}{|\mathcal{S}|}\sum_{\mathcal{S}_i \in \mathcal{S}} F_\beta(\mathcal{S}_i).$$

Given an algorithmic community, $precision$ indicates how many vertices are actually in the same ground-truth community. Given a ground-truth community, $recall$ indicates how many vertices are predicted to be in the same community in a retrieved community. By definition, the precision and the recall are evenly weighted in F_1 measure. On the other hand, the F_2 measure puts more emphasis on recall than precision. The authors in [29] who provided the datasets argue that it is important to quantify the recall since the ground-truth communities in these datasets are partially annotated, i.e., some vertices are not annotated to be a part of the ground-truth community even though they actually belong to that community. This indicates that it would be reasonable to weight recall higher than precision, which is done by the F_2 measure.

TABLE 6
AUC of Metric-versus-Coverage – We Use Three Metrics: Conductance, Modularity, And Average Association

Graph	Metric	oslom	demon	bigclam	nise-lcm-fppr	nise-rnd-fppr	nise-grc-fppr	nise-sph-fppr	nise-grc-ppr	nise-sph-ppr
HepPh	conductance	0.5349	0.4970	0.3752	0.5655	0.7767	**0.8981**	0.8952	0.8403	0.8266
	modularity	0.0066	0.0391	0.0085	0.0198	0.1404	**0.1886**	0.1751	0.1647	0.1615
	association	15.030	24.623	16.585	6.279	35.019	39.043	36.622	**50.806**	45.734
AstroPh	conductance	0.4202	0.4304	0.3545	0.4590	0.7758	**0.8466**	0.8323	0.8111	0.8063
	modularity	0.0010	0.0224	0.0051	0.0298	0.1336	**0.1638**	0.1576	0.1537	0.1491
	association	11.641	18.336	15.202	10.602	22.928	26.147	25.454	**32.669**	29.392
CondMat	conductance	0.5715	0.4299	0.5125	0.8494	0.8295	**0.8945**	0.8882	0.8487	0.8426
	modularity	0.0007	0.0012	0.0059	0.1580	0.1704	**0.1964**	0.1925	0.1770	0.1738
	association	5.575	5.993	6.341	9.079	9.135	10.579	10.527	**11.549**	11.222
Flickr	conductance	N/A	N/A	0.1201	0.3736	0.6597	**0.8922**	0.6794	0.8828	0.7365
	modularity	N/A	N/A	0.00001	0.0075	0.1147	**0.1945**	0.1204	0.1882	0.1335
	association	N/A	N/A	2.152	8.488	23.013	76.807	26.792	**79.901**	27.483
LiveJournal	conductance	N/A	N/A	0.0465	0.3447	0.3682	**0.8223**	0.8150	0.8176	0.8093
	modularity	N/A	N/A	0.00004	0.0006	0.0014	0.1334	0.1318	**0.1336**	0.1329
	association	N/A	N/A	7.202	13.160	20.068	52.835	45.702	**54.822**	47.958
Myspace	conductance	N/A	N/A	N/A	0.2168	0.3795	**0.8052**	0.7301	0.8008	0.7144
	modularity	N/A	N/A	N/A	0.00007	0.0322	0.1278	0.1013	**0.1317**	0.1091
	association	N/A	N/A	N/A	4.478	21.888	53.125	35.347	**62.141**	31.908

Higher AUC indicates better communities. NISE *outperforms Oslom, Demon, and Bigclam. Within* NISE*, "graclus centers" and "spread hubs" seeding strategies are better than other seeding strategies.*

TABLE 7
F1 and F2 Measures

	DBLP		Amazon		LiveJournal2		Orkut	
	F_1	F_2	F_1	F_2	F_1	F_2	F_1	F_2
bigclam	15.1%	13.0%	27.1%	25.6%	11.3%	13.7%	43.0%	47.4%
demon	13.7%	12.0%	16.5%	15.3%	N/A	N/A	N/A	N/A
oslom	13.4%	11.6%	32.0%	30.2%	N/A	N/A	N/A	N/A
nise-lcm-fppr	13.9%	15.4%	46.3%	56.5%	11.3%	13.8%	40.9%	46.8%
nise-rnd-fppr	17.7%	20.5%	48.9%	58.8%	12.1%	16.5%	54.6%	62.9%
nise-sph-fppr	18.1%	21.4%	49.2%	**59.5%**	12.7%	**18.1%**	55.1%	64.2%
nise-sph-ppr	**19.0%**	**22.6%**	**49.7%**	58.7%	**12.8%**	**18.1%**	**57.4%**	**65.2%**
nise-grc-fppr	17.6%	21.7%	46.7%	57.1%	12.2%	17.6%	51.1%	61.4%
nise-grc-ppr	17.6%	22.0%	47.3%	56.0%	**12.8%**	17.6%	53.5%	62.4%

NISE *with "spread hubs" seeding strategy achieves the highest F1 and F2 scores.*

TABLE 8
Running Times of Different Methods on Our Test Networks

Graph	oslom	demon	bigclam	nise-sph-fppr	nise-grc-fppr
HepPh	19 mins. 16 secs.	27 secs.	11 mins. 23 secs.	22 secs.	2 mins. 48 secs.
AstroPh	38 mins. 3 secs.	42 secs.	48 mins. 1 secs.	36 secs.	2 mins. 26 secs.
CondMat	20 mins. 39 secs.	50 secs.	7 mins. 21 secs.	36 secs.	1 min. 14 secs.
DBLP	5 hrs. 50 mins.	3 hrs. 53 mins.	7 hrs. 13 mins.	18 mins. 20 secs.	29 mins. 44 secs.
Amazon	2 hrs. 55 mins.	1 hr. 55 mins.	1 hr. 25 mins.	37 mins. 36 secs.	42 mins. 43 secs.
Flickr	N/A	N/A	69 hrs. 59 mins.	43 mins. 55 secs.	3 hrs. 56 mins.
Orkut	N/A	N/A	13 hrs. 48 mins.	1 hrs. 16 mins.	4 hrs. 16 mins.
LiveJournal	N/A	N/A	65 hrs. 30 mins.	2 hrs. 36 mins.	4 hrs. 48 mins.
LiveJournal2	N/A	N/A	21 hrs. 35 mins.	2 hrs. 15 mins.	6 hrs. 37 mins.
Myspace	N/A	N/A	>7 days	5 hrs. 27 mins.	9 hrs. 42 mins.

In Table 7, we report the average F_1 and F_2 measures on DBLP, Amazon, LiveJournal2, and Orkut networks. A higher value indicates better communities. We see that NISE outperforms Bigclam, Demon, and Oslom in terms of both F_1 and F_2 measures on these networks. Within NISE, "spread hubs" seeding is better than "graclus centers" seeding, and the standard PPR is slightly better than the Fiedler PPR in most of the cases. So, we see that the standard PPR is useful for identifying ground-truth communities. This result is also consistent with the recent observations in [22].

5.5 Comparison of Running Times

We compare the running times of the different algorithms in Table 8. To do a fair comparison, we run the single thread versions of Bigclam and NISE on the HepPh, AstroPh, CondMat, DBLP, and Amazon networks. On

TABLE 9
Running Times (Minutes) of Different Methods on Flickr Networks Of Different Sizes

nodes	edges	bigclam	nise-sph-fppr	nise-grc-fppr
475,621	3,693,728	506	13	65
1,396,462	11,722,538	1,260	24	100
1,994,422	21,445,057	$>4{,}000$	30	97

TABLE 10
F_1 Measures with Different Numbers of Communities

	DBLP		Amazon	
	$k = 20{,}000$	$k = 30{,}000$	$k = 20{,}000$	$k = 30{,}000$
bigclam	16.6%	14.0%	31.9%	22.9%
demon	13.7%	13.7%	16.5%	16.5%
oslom	13.4%	13.4%	32.0%	32.0%
nise-sph-fppr	**17.5%**	**18.8%**	**47.6%**	**49.2%**
nise-grc-fppr	17.2%	17.6%	45.6%	46.7%

TABLE 11
AUC of Conductance-versus-Coverage with Different Numbers of Communities

	CondMat		LiveJournal	
	$k = 150$	$k = 250$	$k = 10{,}000$	$k = 20{,}000$
oslom	0.5715	0.5715	N/A	N/A
demon	0.4299	0.4299	N/A	N/A
bigclam	0.5238	0.5112	0.0472	0.0465
nise-grc-fppr	**0.8910**	**0.8945**	**0.8161**	**0.822**
nise-sph-fppr	0.8882	0.8884	0.8110	0.8166

Higher AUC indicates better communities.

larger networks Demon and Oslom fail to complete. So, we switch to the multi-threaded version of Bigclam and NISE with four threads for Flickr, Orkut, LiveJournal, LiveJournal2, and MySpace. We see that NISE is the only method which can process the largest dataset (Myspace) in a reasonable time. On small networks (HepPh, AstroPh, and CondMat), "nise-sph-fppr" is faster than Demon, Oslom and Bigclam. On medium size networks (DBLP and Amazon), both "nise-grc-fppr" and "nise-sph-fppr" are faster than other methods. On large networks (Flickr, Orkut, LiveJournal, LiveJournal2, Myspace), NISE is much faster than Bigclam.

Table 9 shows the running times (in minutes) on Flickr networks [11] of three different sizes. Demon and Oslom fail on these three networks. We set $k = 10{,}000$ for Bigclam and NISE. We observe that when the input size increases by a factor of 3, the runtime of NISE increases by a factor of 1.5 or 1.8 depending on the seeding strategies (comparison of the first row and the second row of Table 9). By comparing the second row and the third row, we infer that if the differences in sizes of two graphs are not larger than a factor of 2, the inherent structure of the network has more impact on the run time of NISE than the input size itself. This occurs because the most time-consuming step in our computation is the seed expansion phase. The run time of this depends more strongly on the *output clusters* than the input network size. Finally, we observe that NISE is much faster than Bigclam on all of these networks.

5.6 Varying the Number of Communities

We need to specify the number of communities for NISE and Bigclam whereas Demon and Oslom automatically identify the number of communities. Thus, we also conduct experiments using different numbers of communities to ensure that our results are not an extremal case. Table 10 shows the F_1 scores of each method with different values of k, the number of communities. The outputs of Demon and Oslom are not affected by different k, but we include these results for reference. Also, Table 11 shows the AUC of conductance-versus-coverage with different k. We see that NISE consistently outperforms other methods with a reasonable range of k in terms of both F1 measures and AUC scores even with untuned values of the number of communities.

6 Discussion and Conclusion

We now discuss the results from our experimental investigations. First, we note that NISE is the only method that worked on all of the problems. Also, our method is faster than other state-of-the-art overlapping community detection methods. Perhaps surprisingly, the major difference in cost between using "graclus centers" for the seeds and the other seed choices does not result from the expense of running Graclus. Rather, it arises because the personalized PageRank expansion technique takes longer for the seeds chosen by Graclus. When the PageRank expansion method has a larger input set, it tends to take longer, and the "graclus centers" seeding strategy is likely to produce larger input sets because of the neighborhood inflation and because the central vertices of clusters are likely to be high degree vertices.

We wish to address the relationship between our results and some prior observations on overlapping communities. The authors of Bigclam found that the dense regions of a graph reflect areas of overlap between overlapping communities. By using a conductance measure, we ought to find only these dense regions – however, our method produces much larger communities that cover the entire graph. The reason for this difference is that we use the entire vertex neighborhood as the restart for the personalized PageRank expansion routine. We avoid seeding exclusively inside a dense region by using an entire vertex neighborhood as a seed, which grows the set beyond the dense region. Thus, the communities we find likely capture a combination of communities given by the ego network of the original seed node.

Overall, NISE significantly outperforms other state-of-the-art overlapping community detection methods in terms of run time, cohesiveness of communities, and ground-truth accuracy. Also, our new seeding strategies, "graclus centers" and "spread hubs", are superior than existing methods, thus play an important role in the success of our seed set expansion method.

Acknowledgments

This research was supported by US NSF grants CCF-1117055 and CCF-1320746 to ID, and by NSF CAREER award CCF-1149756 to DG.

REFERENCES

[1] J. Lee, S. P. Gross, and J. Lee, "Improved network community structure improves function prediction," *Sci. Rep.*, vol. 3, p. 2197, Jul. 2013.

[2] R. Andersen and K. J. Lang, "Communities from seed sets," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 223–232.

[3] G. Karypis and V. Kumar, "Multilevel K-way partitioning scheme for irregular graphs," *J. Parallel Distrib. Comput.*, vol. 48, pp. 96–129, 1998.

[4] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors: A multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[5] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state of the art and comparative study," *ACM Comput. Surveys*, vol. 45, no. 4, pp. 43:1–43:35, 2013.

[6] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *Physica A*, vol. 388, no. 8, pp. 1706–1712, 2009.

[7] J. J. Whang, X. Sui, and I. S. Dhillon, "Scalable and memory-efficient clustering of large-scale social networks," in *Proc. 12th IEEE Int. Conf. Data Mining*, 2012, pp. 705–714.

[8] D. F. Gleich and C. Seshadhri, "Vertex neighborhoods, low conductance cuts, and good seeds for local community methods," in *Proc. 18th ACM Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 597–605.

[9] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using PageRank vectors," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci.*, 2006, pp. 475–486.

[10] (2014). Stanford Network Analysis Project [Online]. Available: http://snap.stanford.edu/

[11] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Growth of the Flickr social network," in *Proc. 1st Workshop Online Social Netw.*, 2008, pp. 25–30.

[12] H. H. Song, B. Savas, T. W. Cho, V. Dave, Z. Lu, I. S. Dhillon, Y. Zhang, and L. Qiu, "Clustered embedding of massive social networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 331–342, 2012.

[13] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[14] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'Small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[15] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[16] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Math.*, vol. 6, no. 1, pp. 29–123, 2009.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd, (1999, Nov.). The PageRank citation ranking: Bringing order to the web, Stanford Univ., Stanford, CA, US, Tech. Rep. 1999-66 [Online]. Available: http://dbpubs.stanford.edu:8090/pub/1999-66

[18] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proc. 10th ACM Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 653–658.

[19] B. Abrahao, S. Soundarajan, J. Hopcroft, and R. Kleinberg, "On the separability of structural classes of communities," in *Proc. 18th ACM Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 624–632.

[20] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi, "A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally," *J. Mach. Learning Res.*, vol. 13, no. 1, pp. 2339–2365, 2012.

[21] F. Bonchi, P. Esfandiar, D. F. Gleich, C. Greif, and L. V. Lakshmanan, "Fast matrix computations for pairwise and columnwise commute times and Katz scores," *Internet Math.*, vol. 8, nos. 1-2, pp. 73–112, 2012.

[22] I. M. Kloumann and J. M. Kleinberg, "Community membership identification from small seed sets," in *Proc. 18th ACM Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1366–1375.

[23] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814–818, 2005.

[24] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, pp. 761–764, 2010.

[25] S. Zhang, R.-S. Wang, and X.-S. Zhang, "Identification of overlapping community structure in complex networks using fuzzy C-means clustering," *Physica A*, vol. 374, no. 1, pp. 483–490, 2007.

[26] R. S. Burt, *Structural Holes: The Social Structure of Competition*. Cambridge, MA, US: Harvard Univ. Press, 1995.

[27] B. S. Rees and K. B. Gallagher, "Overlapping community detection by collective friendship group inference," in *Proc. Int. Conf. Adv. Social Netw. Anal. Mining*, 2010, pp. 375–379.

[28] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Demon: A Local-first discovery method for overlapping communities," in *Proc. 18th ACM Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 615–623.

[29] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 587–596.

[30] U. Gargi, W. Lu, V. Mirrokni, and S. Yoon, "Large-scale community detection on YouTube for topic discovery and exploration," in *Proc. 5th Int. AAAI Conf. Weblogs Social Media*, 2011, pp. 486–489.

[31] A. Lancichinetti, F. Radicchi, J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PLoS ONE*, vol. 6, no. 4, p. e18961, 2011.

[32] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using seed set expansion," in *Proc. 22nd ACM Int. Conf. Inform. Knowl. Manage.*, 2013, pp. 2099–2108.

[33] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci.*, vol. 103, no. 23, pp. 8577–8582, 2006.

[34] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

Joyce Jiyoung Whang received the BS degree in computer science and engineering from Ewha Womans University (Seoul, South Korea), and the PhD degree in computer science from the University of Texas at Austin. She is an assistant professor of computer engineering at Sungkyunkwan University. Her main research interests include data mining, big data, social network analysis, and machine learning with specific interests in community detection, overlapping clustering, and graph partitioning. She is a member of the IEEE.

David F. Gleich received the BS degree from Harvey Mudd College and the PhD degree from Stanford University. He is an assistant professor of computer science at Purdue University. His research is on matrix computations, network and graph algorithms, and parallel and distributed computing. He has been awarded a Microsoft Research Graduate fellowship, the John von Neumann postdoctoral fellowship, and an US NSF CAREER award.

Inderjit S. Dhillon received the BTech degree from IIT Bombay, and the PhD degree from UC Berkeley. He is the Gottesman Family Centennial professor of computer science and mathematics at UT Austin, where he is also the director of the ICES Center for Big Data Analytics. His main research interests include big data, machine learning, network analysis, linear algebra, and optimization. He has received several prestigious awards, including the ICES Distinguished Research Award, the SIAM Outstanding Paper Prize, the Moncrief Grand Challenge Award, the SIAM Linear Algebra Prize, the University Research Excellence Award, and the US NSF Career Award. He has published over 120 journal and conference papers, and has served on the Editorial Board of the *Journal of Machine Learning Research*, *the IEEE Transactions of Pattern Analysis and Machine Intelligence*, *Foundations and Trends in Machine Learning*, and *the SIAM Journal for Matrix Analysis and Applications*. He is an IEEE fellow, a SIAM fellow, and an ACM fellow.