Scalable Spectral Algorithms for Community Detection in Directed Networks

Sungmin Kim KIM.2774@OSU.EDU
Tao Shi TAOSHI@STAT.OSU.EDU
Department of Statistics
The Ohio State University
1958 Neil Avenue, Columbus, OH 43210-1247, USA

Abstract

Community detection has been one of the central problems in network studies and directed network is particular challenging due to asymmetry among its links. In this paper, we found that incorporating the direction of links reveals new perspective on communities regarding to two different roles, source and terminal, that a node plays in each community. Intriguingly, such communities appear to be connected with unique spectral property of the graph Laplacian of the adjacency matrix and we exploit this connection by using regularized SVD methods. We propose harvesting algorithms, coupled with regularized SVDs, that are linearly scalable for efficient identification of communities in huge directed networks. The algorithm showed great performance and scalability on benchmark networks in simulations and successfully recovered communities in real networks applications (with ~ 2 million nodes and ~ 50 million edges).

Keywords: Community extraction, Graph Laplacian, Regularized SVD, Scalable algorithm, Social networks

1. Introduction

Many real world problems can be effectively modeled as complex relationship in networks where nodes represent entities of interest and links mimic the interactions or relationships among those nodes. The study of such complex relationship networks, recently referred to as *network science*, can provide insight into their structures and properties (Newman, 2006). One particularly interesting area in studies of network structures is searching for important sub-networks which are called communities, modules or groups. A community in a network is typically characterized by a group of nodes that have more links connected within the community than connected to out of the community (Fortunato, 2010). Community detection is in growing attention not only because it leads to understanding of the complex network structure, but also it allows further analysis such as studies on information flows on in networks, evolution of networks and visualization of networks.

Upon the advancement of technology, the size of the network we encounter today is way beyond what we traditionally are able to handle. Modern networks such as social networks, citation networks, Internet networks simply exceed millions of nodes. Consequently, recent developments of community detection algorithms pay considerable attention to the scalability of the algorithms. A survey paper, Parthasarathy et al. (2011), summarizes recent developments in scalable algorithms such as Meits (Karypis and Kumar, 1998), Graclus (Dhillon et al., 2007), MLR-MCL (Satuluri and Parthasarathy, 2009) and local graph clustering of Andersen et al. (2006). However, those scalable community detection algorithms are mainly designed for undirected networks.

In many practical applications, there is a large number of networks that are directed in nature, such as the World Wide Web, tweeter's follower-followee network, and paper citation networks. Even though a few algorithms developed for undirected networks can be extended to apply on directed networks (Danon et al., 2005; Fortunato, 2010; Coscia et al., 2011), the notion of community in undirected networks cannot be simply translated to the directed ones. A common approach to handle directed networks is symmetrizing the adjacency matrix and treating the resulted matrix as from an undirected network. However, it is not uncommon to see that ignoring the direction of edges results in abnormal communities (Leicht and Newman, 2008).

The aim of the present work is to develop scalable community detection algorithms that explicitly incorporate the direction of links. The nodes and links in a directed network are often presented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} \equiv \{v_1, \ldots, v_n\}$ and $\mathcal{E} \equiv \{e_1 \ldots, e_m\}$ denoting the vertex set and edge set respectively. For an existing edge e in the network, we denote its source node and terminal node as $v^s(e)$ and $v^t(e)$ respectively. Let W be the $|\mathcal{V}| \times |\mathcal{V}|$ adjacency matrix in which $W(i,j) = 1$ indicates the existence of an edge originated from v_i and pointed to v_j and $W(i,j) = 0$ otherwise.

Compared with in-depth studies of community structures in undirected networks (Danon et al., 2005; Fortunato, 2010; Coscia et al., 2011), community detection in directed networks has not been as fruitful. One particular difficulty in studying the structure of directed networks is the lack of a clear definition of the connectivity between each pair of nodes. As a result, it is hard to define communities due to the asymmetry nature of the edges. Existing works (Kleinberg, 1999) pointed out the importance of recognizing the dual roles, as source and terminal of edges, that nodes play in a directed network. In this paper, we concentrate on directly incorporating directional edges in analysis and we start with defining a new type of community, *Directional Component*, which contains a source part and a terminal part based on the connectivity between nodes following a path of the edges in alternative directions.

From a theoretical point of view, we study the connection between the directional components in a directed network and the spectrum (singular values and singular vectors), of the graph Laplacian of the network. We prove that the number of directional components in a directed network equals to the multiplicity of the top singular values of the graph Laplacian. In addition, nodes corresponding to nonzero elements in each pair of top singular vectors form the source part and terminal part of one of the directional components. With these theoretical insights, we observe that the successful DI-SIM algorithm proposed in Rohe and Yu (2012) is able to find the source parts and the terminal parts of the directional components. Even with small number of external edges connecting the directional components, the DI-SIM algorithm still stands a good chance of finding approximate directional components due to the stability of the top spectrum of the graph Laplacian.

To build scalable community detection algorithms, we explore the idea of extracting one community, like an approximate directional component, at a time. We propose to use regularized SVD on the graph Laplacian to achieve this goal. Two types of penalizations,

L_0 and Elastic-net, are studied in this paper. The L_0 regularized SVD leads to hard-thresholding of the vectors in the power methods iteration and Elastic-net regularization leads to soft-thresholding. We show that the corresponding threshold level in both methods can be calculated through direct deterministic searching algorithms, other than calling optimization routines. This direct searching step dramatically reduces the computation time and improves the scalability of the regularized SVD methods. Using the regularized SVD as building blocks, we design a community harvesting algorithm that extracts communities from a large network one at a time.

We test the proposed community harvesting algorithms and compare them with existing methods on simulated networks and real networks. In our simulation study, we find that L_0 harvesting and Elastic-net harvesting algorithms perform well when the network is sparse or the communities are tight, which is the case for most social networks. Through simulations, we also find the harvesting algorithm scale well to big networks, even with a large number of communities. We apply these algorithms to a citation network (30,228 nodes and 110,654 edges) with manually assigned categories on papers and found the results from harvesting algorithms are largely consistent with the categories and outperform other methods. In addition, we check the scalability of our algorithms on a social network (1,944,589 nodes and 50,655,143 edges) and found the algorithms are applicable for such a large network and the results reveal interesting structures as well.

The remainder of the paper is organized as follows. In Section 2, we review recent developments in community detection in directed networks and propose a new concept of community, *Directional Component*. Section 3 exploits the connections between directional components and the spectrum of the graph Laplacian of the adjacency matrix. Based on this connection, we propose regularized SVD algorithms for community extraction. Section 4 addresses the problem of selecting parameters for regularized SVD and develops community harvesting algorithm. Section 5 contains simulation studies in various community structures and Section 6 shows the results of harvesting algorithms in two real-world networks. Finally, we conclude the paper with conclusions and discussion in Section 7. Theory proofs are detailed in Appendix.

2. Directed Networks and Directional Components

In this section, we first review existing efforts made in incorporating directional information for community detections in directed networks. Following the review, we extend those concepts to a new definition of community, *Directional Component*. We further discuss the properties of directional components and present a simple algorithm to identify them.

2.1 Review: Community Detection in the Directed Networks

A common approach to handle a directed network is symmetrizing the asymmetric adjacency matrix and treating the resulted matrix as undirected. This approach may work reasonably well in cases where a directed network has large portion of symmetric edges. However, ignoring the direction of edges usually results in abnormal community detection results (Leicht and Newman, 2008).

In the literature of community detection in directed networks, several authors has attempted to directly incorporate directional information into their algorithms. They can be categorized as probability model based (Newman and Leicht, 2007), spectral based (Capocci et al., 2005; Andersen and Lang, 2006; Rohe and Yu, 2012), modularity based (Guimerà et al., 2007; Arenas et al., 2007; Leicht and Newman, 2008), and information based (Rosvall et al., 2009). In particular, several methods, Capocci et al. (2005); Leicht and Newman (2008); Satuluri and Parthasarathy (2011); Rohe and Yu (2012), are based on concepts of in-link similarity and out-link similarity. The in-link (out-link) similarity between a pair of nodes is defined as the number of common nodes having edges pointing to (from) them. A modularity of a directed network was proposed based on these two similarity measures in Leicht and Newman (2008) and they suggested to find communities by maximizing this modularity. Satuluri and Parthasarathy (2011) proposed a degree-discounted symmetrization method that combines the in-link similarity and the out-link similarity of a directed network so that existing community detection algorithms for undirected networks can be applied.

It is also suggested in Rohe and Yu (2012) to detect communities by investigating the spectrum of graph Laplacian of the adjacency matrix W. The graph Laplacian Q of a directed network is defined as

$$Q = D_r^{-\frac{1}{2}} W D_c^{-\frac{1}{2}}, \tag{1}$$

where D_r is the diagonal matrix of out-degrees (or row sum of W) and D_c is the diagonal matrix of in-degrees (or column sum of W)[1]. The proposed algorithm, DI-SIM algorithm, clusters nodes in two different ways (co-clustering) by running the k-means algorithm on the leading left singular vectors and right singular vectors separately. They showed that the DI-SIM algorithm may recover stochastically equivalent sender-nodes and receiver-nodes under Stochastic Co-Block model, which is a relaxed version of Stochastic Block model of Holland et al. (1983).

A close look at these methods reveals that a key to the success of these methods is to recognize the dual roles that nodes play in a directed network, as source nodes and terminal nodes. Actually, Kleinberg (1999) differentiated these two different roles that webpages play in the World Wide Web network and proposed a *hyperlink-induced topic search* (HITS) algorithm to find important websites. The influence of all websites is characterized by a hub score vector ($\mathbf{u}$) and an authority score vector ($\mathbf{v}$) that are solutions of

$$\begin{cases} \mathbf{u} &= \alpha W \mathbf{v}, \\ \mathbf{v} &= \beta W^t \mathbf{u}, \end{cases} \qquad \text{subject to} \quad \mathbf{u}^t\mathbf{u} = \mathbf{v}^t\mathbf{v} = 1. \tag{2}$$

for fixed constants α, β. An interesting property of the two scores is that this pair of scores actually define each other. In his words (Kleinberg, 1999), there are *reinforcement relationship of nodes*, which means that good hubs are the nodes that refer to good authorities and good authorities are the nodes that are referred by good hubs, as reflected in (2). Even though the HITS algorithm originally aims the global reinforcement relationship, it gives us a clue to a concept of community that is based on the local reinforcement relationship between hub nodes and authority nodes.

1. We define $\frac{0}{0} = 0$ for convenient notations.

2.2 Directional Components

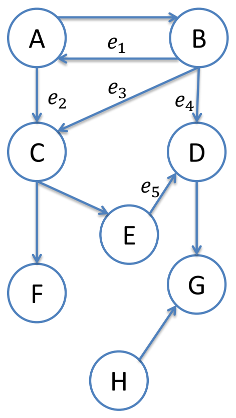
As we have discussed so far, understanding the two different roles that a node plays in a directed network is crucial in incorporating the direction of edges into the concept of communities. We start with exploring different definition of connected nodes in this section. Two types of connectivity between nodes have been studied in the literature of directed networks. *Weak connectivity* defines two nodes s and t $(s, t \in \mathcal{V})$ as weakly connected if they reach each other through a path $e_1, e_2, \ldots, e_l$, regardless of the directions of edges in the path. Meanwhile, *Strong connectivity* takes in account of the directions of the edges in the path and claims nodes s and t are strongly connected only if the path $(e_1, e_2, \ldots, e_l)$ also satisfies $v^s(e_1) = s, v^t(e_l) = t, v^t(e_k) = v^s(e_{k+1}), k = 1, \ldots, l-1$. In this paper, we define a new type of connectivity, *D-connectivity*, as

Definition 1 *Two nodes s and t $(s, t \in \mathcal{V})$ are **D-connected**, denoted by $s \to t$, if there exists a path of edges $(e_1, \ldots, e_{2m-1})$, $m \in \mathbb{R}^+$, satisfying $v^s(e_1) = s, v^t(e_{2m-1}) = t$ and*

$$\begin{cases} v^t(e_{2k-1}) = v^t(e_{2k}) & \textit{(common terminal nodes)} \\ v^s(e_{2k}) = v^s(e_{2k+1}) & \textit{(common source nodes)} \end{cases}$$

for $k = 1, 2, \ldots, \max\{m-1, 1\}$.

D-connectivity follows the edges in alternative directions, one forward and then backward. We call this sequence of edges a D-connected path. Figure 1 provides an illustration of D-connectivity. For example, we observed that $A \to D$ through the sequence of edges (e_2, e_3, e_4) and $E \to A$ through the sequence of edges (e_5, e_4, e_1).



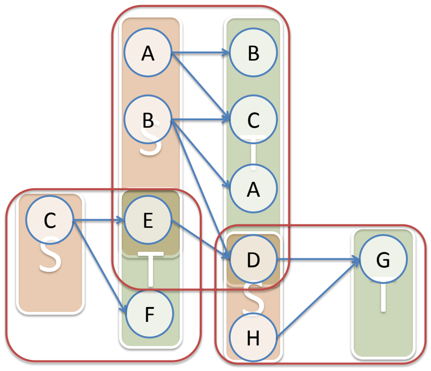
	A	B	C	D	E	F	G	H
A	0	1	1	0	0	0	0	0
B	1	0	1	1	0	0	0	0
C	0	0	0	0	1	1	0	0
D	0	0	0	0	0	0	1	0
E	0	0	0	1	0	0	0	0
F	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	1	0

Figure 1: An example directed network and its adjacency matrix.

The definition of D-connectivity is a restricted version of a concept called *alternating connectivity* that was introduced by Kleinberg (1999), in the context of analyzing the centrality of webpages in a sub-network of World Wide Web using the HITS algorithm. The difference is that the alternating connectivity allows two nodes to be any pair of nodes on a D-connected path being alternatively connected. Kleinberg (1999) also pointed out the difficulty of extending the alternating connectivity between a pair of nodes to a concept that characterizes a group of tightly connected nodes (a community), because transitive relation does not hold in alternating connectivity. However, the D-connectivity bypasses this problem by recognizing the two different roles, sources and terminals. Next we define a new type of community structure, *Directional Component*, based on the D-connectivity.

Definition 2 *A* ***Directional Component*** *(DC) consists a source node set S and a terminal node set T ($S, T \subset \mathcal{V}$) and they are the maximal subset of nodes such that any pair of nodes $(s, t), s \in S, t \in T$, are D-connected ($s \rightarrow t$). We call S and T the source part and terminal part of the directional component and denote $DC \equiv (S, T)$.*

The concept of directional component provides a potential way to partition a network into smaller communities. First, a node v in a directional component may belong to either the source part S or the terminal part T or both. Second, in a directed network that contains multiple directional components $DC_1, DC_2, \ldots, DC_K$, any node $v \in \mathcal{V}$ can only belong to one of the source part S. In other words, the source parts $S_1, \ldots, S_K$ are disjoint and the same holds for $T_1, \ldots, T_K$. However, it is possible that $v \in S_i$ and $v \in T_j$ for any i and j.



	A	B	C	D	E	F	G	H
A	0	1	1	0	0	0	0	0
B	1	0	1	1	0	0	0	0
E	0	0	0	1	0	0	0	0
C	0	0	0	0	1	1	0	0
D	0	0	0	0	0	0	1	0
H	0	0	0	0	0	0	1	0
F	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0

Figure 2: The decomposition of the network in Figure 1 and rearranged adjacency matrix.

This two-way partition of nodes respects the asymmetric property of the directed network. Figure 2 illustrates the decomposition of the directed network shown in Figure 1. Three directional components are found and the source and terminal parts of each directional component are displayed in boxes. As we discussed above, node A appear in the source part and the terminal part of the same component, but node D belongs to the source part and the terminal part of two different directional components. Meanwhile, all source (terminal) parts are disjoint. After reorganizing the nodes by directional components, it is clear that there is no edges between the source part and terminal part of different directional components, as shown in the right panel of Figure 2. Therefore, this two-way partition of nodes results in a re-ordered adjacency matrix that exhibits clear block-wise structure.

Finding directional components can be achieved through a simple searching algorithm of computation complexity $O(|\mathcal{V}| + |\mathcal{E}|)$. Identification of one directed component is done by iterations of adding nodes into the source part and the terminal part. Staring with picking one node with a positive out-degree as a member of a source part, the algorithm iterates between steps:

1. For each node in source part, add all nodes pointed by this node to the terminal part;
2. For each node in terminal part, add all nodes pointing this node to the source part;

The iteration runs until no more nodes need be added in either source part or terminal part. Once the iteration terminates, one may remove all edges of the identified directional component from the original network and repeat the iteration to identify another directional component on the reduced network.

One drawback of this direct searching algorithm is that it usually detect only few large directional components, even when the network has much finer community structures. This phenomenon is due to fact that it is unrealistic to expect absolutely no links between those small components. For example, a direct search on the Cora citation network [2], which presents bibliographic citation between papers in computer science, leads to one giant directional component that covers all nodes. However, a close look at the dataset suggests that there are much more citations between papers in a similar field and much less between papers in different ones. In other words, many smaller sized communities exist, but the algorithm is too strict to detect them. Therefore we need consider more flexible algorithms that may detect a directional component with existence of a small number of external edges.

3. Regularized SVD Algorithms for Community Extraction

We first explore the connection between the directional components in a directed network and the leading spectrum of its graph Laplacian Q, as defined in (1). These spectral properties motivate us to propose using regularized SVD algorithms to extract communities that are close to directional components. We also show that the computation of this class of regularized SVD algorithms is in linear order of the number of nodes and edges in the network, so it is scalable.

3.1 Spectrum of Graph Laplacian and Directional Components

We start with investigating the spectral properties of the graph Laplacian of a network with several DCs. It is well known that the spectrum of graph Laplacian of an undirected network is strongly tied with its connected components (Von Luxburg, 2007). More specifically, the multiplicity of the largest eigenvalue (one) is the same as the number of connected components in the network. The next theorem shows similar connections in directed networks, in which the concept of connected components is replaced by directional components.

For a subset of nodes, $A \subset \mathcal{V}$, in a network of $|\mathcal{V}| = n$, we define $\mathbf{1_A}$ as an indicator vector of length n with each element $\mathbf{1}_{\mathbf{A}(i)} = I(v_i \in A)$. Recall that S_k and T_k represent the source part and terminal part of the k-th directional component respectively. We have,

Theorem 3 *The principal singular value of Q in a directed network is* **one** *and its multiplicity, K, equals to the number of directional components in the network. In addition, the principal left (or right) singular vector space is spanned by $\{D_r^{\frac{1}{2}}\mathbf{1_{S_1}}, \ldots, D_r^{\frac{1}{2}}\mathbf{1_{S_K}}\}$ (or $\{D_c^{\frac{1}{2}}\mathbf{1_{T_1}}, \ldots, D_c^{\frac{1}{2}}\mathbf{1_{T_K}}\}$) respectively.*

2. More details of this dataset will be provided in Section 6

The complete proof is included in Appendix A.1. The main idea of the proof is behind the connection between the spectrum of a directed network and that of the bipartite graph generated from it. The nonzero entries of each $D_r^{\frac{1}{2}}\mathbf{1}_{\mathbf{S_k}}$ correspond to the source part of one directional component and similar result holds for $\{D_c^{\frac{1}{2}}\mathbf{1}_{\mathbf{T_k}}\}_{k=1,\dots,K}$ and terminal part.

Theorem 3 indicates that the space spanned by the top singular vectors contains almost complete information about all directional components. However, the output space calculated from any SVD algorithms is up to a rotation due to the multiplicity of the top singular value. The DI-SIM algorithm (Rohe and Yu, 2012) address this problem by applying the k-means algorithm on the left and right singular vectors separately. In addition, even with the presence of a small fraction of external edges between directional components, the space spanned by top singular vectors should still be stable when a large eigen-gap exists.

Although it seems attractive to apply the DI-SIM algorithm to find directional components in the presence of external edges, a few limitations of the algorithm make it less effective in practice, especially for large networks. First, the two sets of clustering results from k-means on left and right singular vectors are not paired and it is not obvious how one may connect these two parts of the solution. Second, the DI-SIM algorithm needs to pre-specify the number of communities, which is unknown in practice. If the pre-specified value is too small, the leading spectrum of SVD will be unstable. As a result, the community detection results will be less accurate. More importantly, the computation of DI-SIM is very demanding for large networks with many communities, since it tries to detect all communities at the same time. SVD computation increases quadratically as the number of singular vectors increases and the k-means algorithm slows down dramatically when either the number of nodes or the number of groups is large.

In response to these limitations, we consider direct-searching algorithms to identify one community at a time rather than attempting to discover all components by the division of nodes. This strategy leads to algorithms that do not need pre-specification of the number of communities and in turn provides stable results and scalable computation.

3.2 Regularized SVD with L_0 penalty

To design methods that extract directional components from a network one at a time, we start with defining the size of a community $C(S,T)$ that consists a source part S and terminal part T. In directed networks, such as citation network or social network, the distribution of in-degrees is usually very different from that of out-degrees. Therefore, it is nature to treat S and T differently. We define the size of a community $C(S,T)$ as

$$SZ_\omega(C) = SZ_\omega(S,T) \equiv |S| + \omega|T|, \tag{3}$$

where the constant $\omega > 0$ balances the in-degree and the out-degree.

With this definition, we may order all directional components of a directed network in an increasing order of their sizes such that $SZ_\omega(DC_1) \leq SZ_\omega(DC_2) \leq \ldots \leq SZ_\omega(DC_K)$. Since each directional component corresponds to a pair of vectors $(D_r^{\frac{1}{2}}\mathbf{1}_{\mathbf{S_k}}, {}^{\frac{1}{2}}\mathbf{1}_{\mathbf{T_k}})$ with the largest singular value 1, we should be able to identify the smallest directional component by penalizing the SVD by the size of the component corresponding to non-zero elements of the singular vectors. This is achieved by adding L_0 penalty to vectors $\mathbf{u}$ and $\mathbf{v}$ in SVD as

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^t Q\mathbf{v} - \eta(\|\mathbf{u}\|_0 + \omega\|\mathbf{v}\|_0), \qquad \|\mathbf{u}\|_2 \leq 1, \quad \|\mathbf{v}\|_2 \leq 1. \tag{4}$$

The solution $(\mathbf{u}, \mathbf{v})$ from (4) leads to a community $C(S, T)$ with $S = \{v : \mathbf{u}(v) \neq 0\}$ and $T = \{v : \mathbf{v}(v) \neq 0\}$. The following proposition connects the smallest directional component with the solution of the L_0 regularized SVD.

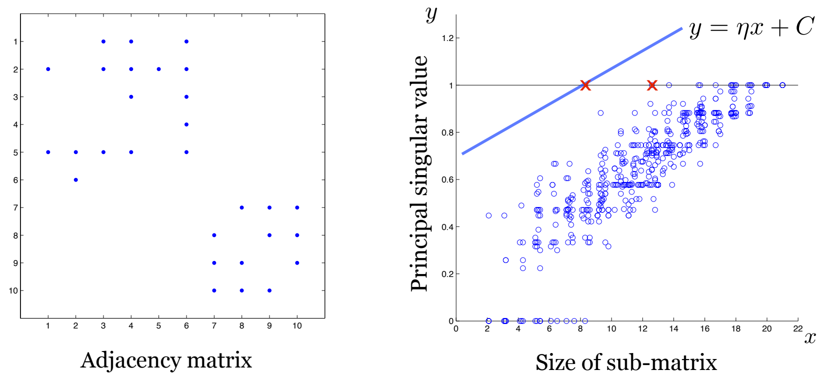
Proposition 4 *There exists a cutoff value $\epsilon > 0$ such that for any $0 < \eta < \epsilon$, the community defined by the solution $(\mathbf{u}, \mathbf{v})$ of the optimization problem (4) is the same as the smallest directional component in the directed network.*

With its proof given in Appendix A.2, the proposition implies that one of the smallest directional components can be found by solving the optimization problem (4) with a sufficiently small η. Details in the proof also reveal that the optimization essentially searches for one of the smallest sub-matrices of Q that has singular value 1 in this case.

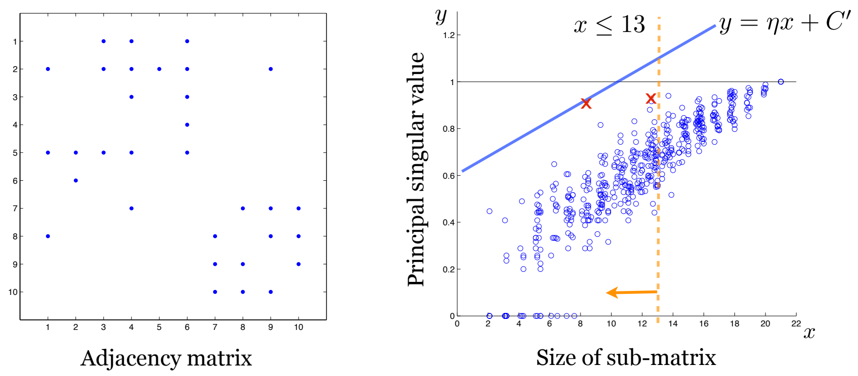
We provide a simple example to illustrate the ideas behind this approach. The left panel of Figure 3a shows the adjacency matrix of a network with ten nodes. The network has two directional components with different sizes. The parameter ω in defining the size of communities is set to 1.1 in (3). We randomly select two subsets (S, T) of nodes to generate a sub-matrix $Q_s(S, T)$ from the full graph Laplacian matrix Q, considering S, T as indexes of rows and columns of Q respectively. For each selected $Q_s(S, T)$, we calculate its principal singular value $\lambda_1(Q_s(S, T))$ and its size $SZ_\omega(S, T)$. In addition to 500 randomly chosen sub-matrices, we also include those two directional components as testing communities.

The right panel of Figure 3a show paired values $(SZ_\omega(S, T), \lambda_1(Q_s(S, T)))$, with o, for each sampled sub-matrix Q_s, and those two directional components are marked as Xs. Let us denote the value of objective function in (4) as C. This figure shows that there exist a line with slope $\eta > 0$ whose intercept C is maximized at the point corresponding to the smallest directional component as Proposition 4 describes. Therefore, the optimization (4) leads to identification of the smaller of the two directional components in this network. To summarize the result, both Theorem 3 and the example show that directional components, if there is any, can be identified sequentially by L_0 regularized SVD approach.

Recall that we encountered the problem that the existence of a small number of external edges tends to connect smaller directional components together as one giant directional component. The root of the problem is the too strict requirement on finding exact directional components, the maximal set of node satisfying D-connectivity. The L_0 regularized SVD limits the number of non-zero entries of the singular vectors, so it may find a component that is smaller and almost separated from other components. To illustrate the advantage of the regularized approach, we randomly add three external edges in the example. As a result, those two directional components merge together as one, as shown in the left panel of Figure 3b. The right panel of Figure 3b plots paired values $(SZ_\omega(S, T), \lambda_1(Q_s(S, T)))$ of the same 500 pairs of (S, T) shown in Figure 3a. The principle singular values of two true directional components (X marks) have decreased because of the added external edges, but the line with the same slope is still capable to identify the smallest directional component.



(a) No external edges



(b) After adding three external edges

Figure 3: Left panel of (a): The adjacency matrix of an example network having two directional components. Right panel of (a): Scatter plots of $SZ_\omega(S,T)$ and $\lambda_1(Q_s(S,T))$ of sub-matrix $Q_s(S,T)$ of the graph Laplacian matrix Q of a directed graph having two directional components. Left panel of (b): The adjacency matrix of the example network of Figure 3a after adding three external edges. Right panel of (b): Scatter plots of $SZ_\omega(S,T)$ and $\lambda_1(Q_s(S,T))$ of sub-matrix $Q_s(S,T)$ of the graph Laplacian matrix Q of the directed graph corrupted by external edges.

3.2.1 L_0 Regularized SVD Algorithm

In this section, we show the solution the of L_0 regularized SVD (4) can be found through iterative hard-thresholding. Similar with approaches taken in Shen and Huang (2008) and d'Aspremont et al. (2008), we start with exploiting the bi-linearity of the optimization problem (4). For a fixed vector $\mathbf{v}$, we show how to solve the maximization problem with respect to $\mathbf{u}$. Here we first introduce a notation:

Definition 5 *Given a vector $\mathbf{z} = (z_1, \ldots, z_n)' \in \mathcal{R}^n$, we denote the l-th largest absolute value of $\mathbf{z}$ as $|z|_{(l)}$. Consequently, we define $\mathbf{z}_l^h (\in \mathcal{R}^n)$ as the vector resulted from hard thresholding $\mathbf{z}$ by its $(l+1)$-th largest absolute entry, e.g. the i-th element of $\mathbf{z}_l^h$*

$$\mathbf{z}_l^h(i) = z_i \, I(|z_i| > |z|_{(l+1)}) = z_i \, I(|z_i| \geq |z|_{(l)}),$$

while the superscript "h" stands for hard-thresholding.

For a fixed $\mathbf{v}$, we may treat $Q\mathbf{v}$ as a generic vector $\mathbf{z}$ and find the solution $\mathbf{u}$ that maximizes (4) through the following result:

Theorem 6 *For a given vector $\mathbf{z}$ and a fixed constant $\rho > 0$, the solution of*

$$\max_{\|\mathbf{u}\|_2 \leq 1} \mathbf{u}^t \mathbf{z} - \rho \|\mathbf{u}\|_0 \tag{5}$$

is

$$\mathbf{u} = \mathbf{z}_l^h / \|\mathbf{z}_l^h\|_2,$$

where the integer l is the minimum number that satisfies

$$|z|_{(l+1)} \leq \sqrt{\rho^2 + 2\,\rho\, \|\mathbf{z}_l^h\|_2} \tag{6}$$

Proof When $\|\mathbf{u}\|_0 = l$, $\max_{\mathbf{u}} \mathbf{u}^t \mathbf{z}$ is obtained when $\mathbf{u} = \mathbf{z}_l^h / \|\mathbf{z}_l^h\|_2$. Thus the objective function (5) can be written as

$$F(l) = \|\mathbf{z}_l^h\|_2 - \rho\, l.$$

Now we try to maximize $F(l)$ over l. Notice that $\|\mathbf{z}_l^h\|_2$ monotonically increases as l increases. The value of $F(l)$ keeps increasing until

$$\sqrt{\|\mathbf{z}_l^h\|_2^2 + |z|_{(l+1)}^2} - \|\mathbf{z}_l^h\|_2 \leq \rho,$$

which is equivalent to (6). After l goes beyond this point, since $|z|_{(l)}^2$ decreases and $\|\mathbf{z}_l^h\|_2$ increases, $F(l)$ starts to decrease and keep decreasing. Therefore, The solution to (5) is found at the minimum l that satisfies (6). We note that when vector $\mathbf{z}$ contains tied absolute values, one may just select the first ones following the original order. ■

Theorem 6 leads a computationally efficient algorithm to solve the L_0 regularized SVD in (5). We first sort the entries of $\mathbf{z}$ by their absolute values and then sequentially search from the largest to smallest while testing if condition (6) has been met. As soon as the (6) is satisfied, we obtain the solution. The computation complexity of this direct searching algorithm is $O(n \log(n))$.Consequently, the solution of regularized SVD problem (4) is obtained by using the searching algorithm for a fixed $\mathbf{v}$ and for a fixed $\mathbf{u}$ alternatively. Algorithm 1 lists the details.

Algorithm 1 L_0 regularized SVD

```
Require: $Q, \eta, \omega$
    initialize $\mathbf{v}$
    repeat
        $\mathbf{z} \leftarrow Q\mathbf{v}$ , $\rho \leftarrow \eta$
        $\mathbf{u} \leftarrow \mathbf{z}_l^h / \|\mathbf{z}_l^h\|_2$ , where $l$ is the minimum integer such that $|z|_{(l+1)} \leq \sqrt{\rho^2 + 2\,\rho\,\|\mathbf{z}_l^h\|_2}$
        $\mathbf{z} \leftarrow Q^t\mathbf{u}$ , $\rho \leftarrow \eta\,\omega$
        $\mathbf{v} \leftarrow \mathbf{z}_l^h / \|\mathbf{z}_l^h\|_2$, , where $l$ is the minimum integer such that $|z|_{(l+1)} \leq \sqrt{\rho^2 + 2\,\rho\,\|\mathbf{z}_l^h\|_2}$
    until $\mathbf{u}, \mathbf{v}$ converged
    return $\mathbf{u}, \mathbf{v}$
```

3.3 SVD with Elastic-net Penalty

In the previous section, we find that the L_0 regularized SVD may detect smaller and tighter communities in direct networks and it can be solved by an efficient algorithm based on the power method combined with hard thresholding. When the number of external edges is relatively small, we found the L_0 regularized SVD performs well. However, when external edges introduce larger perturbation to the spectrum of Q, it may lead to difficulties for (4) to identify the communities, for example, the line $y = \eta x + C$ may hit other o's before finding the second X in the right panel of Figure 3b.

We may consider a slight modification of (4) in the following form:

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^t Q \mathbf{v} \qquad \text{subject to} \qquad \|\mathbf{u}\|_0 + \omega\|\mathbf{v}\|_0 \leq \gamma, \qquad \|\mathbf{u}\|_2 \leq 1, \quad \|\mathbf{v}\|_2 \leq 1, \tag{7}$$

which searches for a sub-matrix of Q that has the largest singular value with a strict constraint on its size. The yellow vertical line in the right panel of Figure 3b shows the constraint when $\gamma = 13$. In this case, the solution of (7) corresponds to the second directional component. This formulation provides another option for recovering the directional components when extra edges present. However, finding a solution of (7) is challenging due to the discrete nature of the constraint.

A typical approach to overcome the computational difficulty related with the non-convexity of the L_0 constraint is to relax L_0 penalty to L_1 penalty. Replacing L_0 penalty to L_1 penalty and separating the penalty for $\mathbf{u}$ and $\mathbf{v}$ result in

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^t Q \mathbf{v} \qquad \text{subject to} \qquad \|\mathbf{u}\|_1 \leq C_1, \|\mathbf{v}\|_1 \leq C_2, \quad \|\mathbf{u}\|_2 \leq 1, \|\mathbf{v}\|_2 \leq 1. \tag{8}$$

This optimization problem is identical to one version of the sparse matrix decomposition method proposed in Witten et al. (2009), in which the authors provided an algorithm to find a local solution. The algorithm uses the power method for SVD combined with soft-thresholding on singular vectors over iterations (Shen and Huang, 2008; Witten et al., 2009; Lee et al., 2010). However we found that the solution of (8) did not report significantly better solutions than L_0 regularized SVD solution from (4) in our simulation studies. One of possible reason is that L_1 constraint may fail to give sufficiently sparse solution of $\mathbf{u}$ and $\mathbf{v}$, as pointed out by Yang et al. (2011).

As an alternative, we propose SVD with Elastic-net type of penalty (Zou and Hastie, 2005),

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^t Q\mathbf{v}, \qquad (9)$$
$$\text{subject to} \quad (1-\alpha)\|\mathbf{u}\|_2^2 + \alpha\|\mathbf{u}\|_1 \leq c_1, \quad (1-\beta)\|\mathbf{v}\|_2^2 + \beta\|\mathbf{v}\|_1 \leq c_2,$$

where the sparsity level is controlled by parameters $\alpha \in [0,1)$ and $\beta \in [0,1)$. Note that $\alpha = \beta = 0$ leads to the regular SVD problem. When $\alpha \in (0,1), \beta \in (0,1)$, the optimization problem becomes non-convex. We show that local optimal solutions of (9) can be found by the power method with soft-thresholding.

3.3.1 Elastic-net Regularized SVD Algorithm

Similar to the calculation of the L_0 regularized SVD, we take advantage of the bi-linearity of the optimization problem. For fixed $\mathbf{v}$ and α, (or $\mathbf{u}$ and β), the optimization becomes convex,

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^t Q\mathbf{v}, \quad \text{subject to} \quad (1-\alpha)\|\mathbf{u}\|_2^2 + \alpha\|\mathbf{u}\|_1 \leq c_1. \qquad (10)$$

whose global solution can be obtained through simple soft-thresholding. We note that Witten et al. (2009) and Lee et al. (2010) showed similar results under slightly different constraints. To find the solution of (10), we first introduce a notation:

Definition 7 *For a vector $\mathbf{z} = (z_1, \ldots, z_n)' \in \mathcal{R}^n$, recall the l-th largest absolute value of $\mathbf{z}$ was defined as $|z|_{(l)}$. Denoting $|z|_{(n+1)} = 0$ for connivence and we define*

$$G_{\mathbf{z}}(x) = \frac{1}{4x^2}\sum_{i=1}^{k(x)-1}(|z|_{(i)} - x)^2 + \frac{1}{2x}\sum_{i=1}^{k(x)-1}(|z|_{(i)} - x) \qquad (11)$$

where $k(x) \in \{1, \ldots, n+1\}$ satisfies $|z|_{(k(x))} \leq x < |z|_{(k(x)-1)}$.

From Witten et al. (2009) we borrow a notation, $S(\mathbf{z}, d)$, as the result of soft-thresholding a vector $\mathbf{z}$ by a scaler d. Soft-thresholding is defined by $S(\mathbf{z}, d) = \text{sign}(\mathbf{z})(|\mathbf{z}| - d)_+$, where $d > 0$ and $x_+ = \max\{x, 0\}$. Again treating $Q\mathbf{v}$ as a generic vector $\mathbf{z}$, we find the solution $\mathbf{u}$ that maximizes (10) by the following result:

Theorem 8 *For a fixed vector $\mathbf{z}$, the solution of the optimization problem,*

$$\max_{\mathbf{u}} \quad \mathbf{u}^t\mathbf{z}, \quad \textit{subject to } (1-\alpha)\|\mathbf{u}\|_2^2 + \alpha\|\mathbf{u}\|_1 \leq c_1$$

is

$$\mathbf{u} = \frac{2d(1-\alpha)}{\alpha} S(\mathbf{z}, d),$$

and the threshold level d is the solution of $G_{\mathbf{z}}(d) = c_1(1-\alpha)/\alpha^2$.

Algorithm 2 SVD with elastic-net penalty

```
Require: $Q, \alpha, \beta, c_1, c_2$
    initialize $\mathbf{v}$
    repeat
        $d \leftarrow$ the solution $x$ of $G_{Q\mathbf{v}}(x) = c_1(1-\alpha)/\alpha^2$
        $\mathbf{u} \leftarrow \frac{2d(1-\alpha)}{\alpha} S(\mathbf{Qv}, d)$
        $d \leftarrow$ the solution $x$ of $G_{Q^t\mathbf{u}}(x) = c_2(1-\beta)/\beta^2$
        $\mathbf{v} \leftarrow \frac{2d(1-\beta)}{\beta} S(\mathbf{Q^t u}, d)$
    until $\mathbf{u}, \mathbf{v}$ are converged
    return $\mathbf{u}, \mathbf{v}$
```

The proof of the theorem is provided in the Appendix A.4 and its first part resembles the proof of Lemma 2.2 of Witten et al. (2009). This theorem leads to Algorithm 2. The computation in Algorithm 2 involves solving for the soft threshold level d in equation $G_{\mathbf{z}}(d) = c$, where c is some constant in the range of the function $G_{\mathbf{z}}(\cdot)$. The solution d is described in Lemma 9.

Lemma 9 *The solution of the equation $G_{\mathbf{z}}(d) = c$ for given $c > 0$ is*

$$d = \left(\frac{\sum_{i=1}^{\hat{k}} |z|_{(i)}^2}{4c + \hat{k}} \right)^{\frac{1}{2}}, \tag{12}$$

where $\hat{k}$ is a positive integer in $\{1, 2, \ldots, n\}$ that satisfies $G_{\mathbf{z}}(|z|_{(\hat{k})}) \leq c, G_{\mathbf{z}}(|z|_{(\hat{k}+1)}) > c$.

The proof of Lemma 9 is detailed in Appendix A.5, along with a simple algorithm for finding $\hat{k}$ in the Lemma. Our contribution is that we seek the threshold level d in the linear time that is proportional to the number of non-zero entries of the solutions, which makes the computation feasible for large matrix in comparison to the binary search method proposed in Witten et al. (2009). Even though Q is nonnegative matrix in our problem, the linear search method can be applied to any real valued matrix. Interestingly, (8) can also be solved using the linear search method instead of the binary search method. We verified that the linear search method is faster than the binary search method by 3 to 20 times when the number of nodes in the network is between 10^3 and 10^7.

In summary, we propose two linearly scalable algorithms, the L_0 regularized SVD and the Elastic-net regularized SVD, for extracting one community from a directed graph. In the next section, we will apply these community extraction algorithms repeatedly to a network and try to harvest all tight communities sequentially.

4. Community Harvesting Through Regularized SVDs

Previous results suggest that the regularized SVD algorithms are capable of recovering a community like directional component under the presence of noise edges, when the penalization parameters, η in (4) or (α, β) in (9), are properly chosen. In addition to specifying the algorithm parameters, one also needs to provide a starting vector $\mathbf{v}$ or $\mathbf{u}$ to initialize the algorithm. In this section, we first discuss the effect of these parameter specifications and how to choose them in practice. Then we propose a community harvesting scheme that repeatedly use the regularized SVD algorithm to extract all communities that are approximate directional components.

4.1 Parameter Selection and Initialization for Regularized SVDs

We now concentrate on studying the effect of the penalization parameters on the algorithm outputs and proposing practical rules on how to specify them. We use the Elastic-net regularized SVD in this section, but the proposed parameter-selection procedure generalizes to the L_0 regularized SVD easily. For Elastic-net regularized SVD, we first point out that the parameters c_1 and c_2 in (10) can be set to one as default values, since they only affect the magnitude of the solution vectors. Theorem 8 suggests that, for any change in $c_1 > 0$, the value of α can be modified to produce the same value of $c_1(1-\alpha)/\alpha^2$ since the range of $(1-\alpha)/\alpha^2$ is $(0, \infty]$. One may concern the positive multiplicative change in $\mathbf{u}$ caused by the change of α. However, the multiplicative change in $\mathbf{u}$ only leads to the same multiplicative change in $\mathbf{v}$ in the following optimization. All multiplicative changes over the optimizations do not change the sets of nodes, $S = \{v : \mathbf{u}(v) \neq 0\}$ and $T = \{v : \mathbf{v}(v) \neq 0\}$.

One property of the regularized Elastic-net SVD algorithm is that one may obtain solutions that change smoothly over small changes of penalization parameters (α, β). The trick is to use the solution $\mathbf{v}^*$ (or $\mathbf{u}^*$) at the current sparsity for the initial vector of the optimization problem of the next contiguous sparsity level. The small change in the sparsity levels allows the algorithm converges within few iterations without dramatic changes in $\mathbf{u}^*, \mathbf{v}^*$. This strategy of setting initial vectors of Algorithm 2 allows us to construct the entire *solution path* on a range of sparsity levels efficiently.

We present an example that shows snap-shots of the solutions corresponding to a few different penalization parameters on a solution path. A network of size 60 with 3 communities, A, B, and C with 20 nodes in each, is simulated from the stochastic block model (Holland et al., 1983). For the three groups, the probability of existing a directed edge between any ordered pair of nodes, (v_s, v_t), is set to 0.3 if $v_s \in A, v_t \in B$ or $v_s \in B, v_t \in A$ or $v_s \in C, v_t \in C$ and the probability is set to 0.05 otherwise. A realization of the network is shown in the form of adjacency matrix in Figure 4. We observe three strong blocks of dense connections when the order of nodes follows the true grouping (A, B and C from left to right).

A solution path from the Elastic-net regularized SVD is generated with an initialization vector $\mathbf{v} = \mathbf{1}_{\{v_1\}}$ and parameters $\alpha = \beta$ decreasing from 0.8 to 0.1 by a step size of 0.05. The panels in Figure 4 show the extracted communities (in red dots) at four different sparsity levels $\alpha = \beta = 0.6, 0.4, 0.15, 0.1$ on the path. As we expected, it is obvious to observe the nested structures among the detected communities on the solution path. At a balanced point, we observe that the algorithm captures the most of links in $(S, T) = (B, A)$ at sparsity level 0.15 while not including too many external edges. Therefore, we achieve good community detection if we are able to select a good penalization level.

We propose to use a directed Normalized-cut (d-Ncut) criterion to select proper penalization parameters on the solution path. For two sets of nodes N_1, N_2, the cut of edges starting from N_1 and ended at N_2 is $\mathrm{Cut}(N_1, N_2) = \sum_{i \in N_1, j \in N_2} W_{ij}$. Given a detected pair

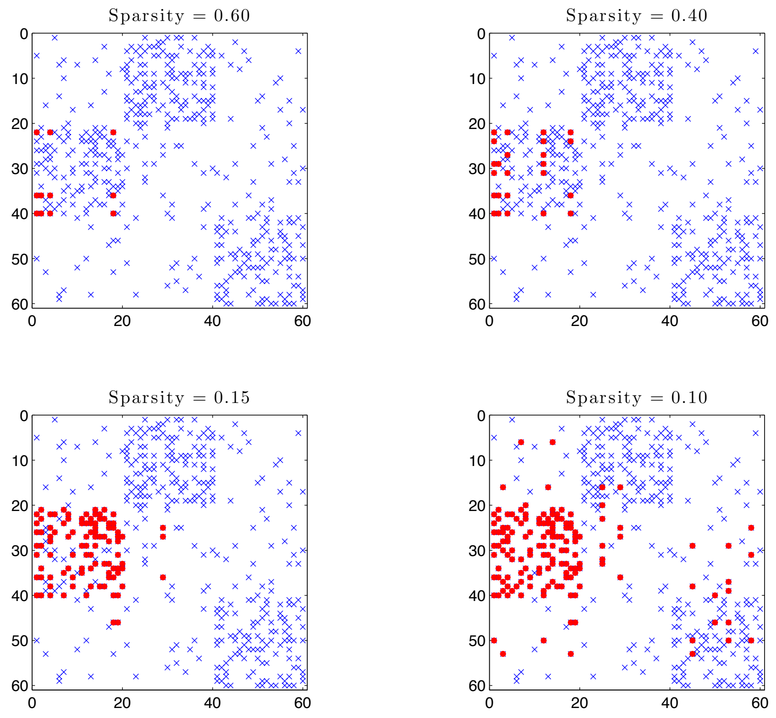


Figure 4: A simulated directed graph from stochastic block model. Within probability 0.3, between probability 0.05, 20 nodes for each block. By decreasing sparsity level, more significantly related links join to (S, T).

of source set and terminal set (S, T), the d-Ncut criterion is defined as

$$\begin{aligned}\text{d-Ncut}(S,T) &= \frac{1}{2}\left\{\frac{\text{Cut}(S,\bar{T})}{\text{Vol}(S)} + \frac{\text{Cut}(\bar{S},T)}{\text{Vol}(T)} + \frac{\text{Cut}(\bar{S},T)}{\text{Vol}(\bar{S})} + \frac{\text{Cut}(S,\bar{T})}{\text{Vol}(\bar{T})}\right\} + \\ &\qquad \text{Cut}(S,T)\left(\frac{1}{\sqrt{\text{Vol}(S)}} - \frac{1}{\sqrt{\text{Vol}(T)}}\right)^2 + \text{Cut}(\bar{S},\bar{T})\left(\frac{1}{\sqrt{\text{Vol}(\bar{S})}} - \frac{1}{\sqrt{\text{Vol}(\bar{T})}}\right)^2,\end{aligned}$$

where $\bar{S}$ and $\bar{T}$ are the compliment nodes sets of S and T respectively. The d-Ncut criterion measures how much (S, T) and $(\bar{S}, \bar{T})$ are close to directional components. The detailed derivation of d-Ncut is presented in Appendix B. The d-Ncut value greater than 1 indicates no community structure in (S, T) while a smaller positive d-Ncut value indicates a stronger community structure, for example, directional components have zero d-Ncut value. The d-Ncut values of the four different results shown in Figure 4 are 0.653, 0.586, 0.398 and 0.425, which correspond to the penalization parameter at 0.6, 0.4, 0.15, and 0.1. The minimum d-Ncut value on the whole solution path is actually 0.398, which leads us to pick the results at sparsity levels $\alpha = \beta = 0.15$.

Another interesting observation made in this example is the dependence between the extracted community and the initialization vector. Since both the Elastic-net regularized

SVD algorithm and the L_0 regularized SVD algorithm are based on local updating, their outputs are sensitive to the initial vector $\mathbf{v}_0$. In the example, the algorithm would have recovered another collection of links in $(S, T) = (A, B)$ if it started from $\mathbf{v}_0 = \mathbf{1}_{\{v_{30}\}}$. Depending on the purpose of the analysis, one may start with a constant vector of $\mathbf{v}_0$ and increase the sparsity levels to recover a rather big d-comp (Top-down), or start with an indicator vector of $\mathbf{v}_0$ and decrease the sparsity levels to recover a d-comp close to the initial node (Bottom-up).

Coupling the d-Ncut criterion with the solution path generating method based on the regularized SVDs, we arrive at a practical algorithm that extracts one community from the network. We name the identified community (S, T) from this algorithm as a *Approximated Directional Component* (*ADC*), to distinguish it from directional components. The steps of the algorithm are described in Algorithm 3. We note that one may replace the Elastic-net regularized SVD with the L_0 regularized version easily.

Algorithm 3 Community Extraction though Elastic-net Regularized SVD

```
Require: $Q$, initialization vector $\mathbf{v}_0$, grid of sparsity levels $\{(\alpha_i, \beta_i)\}_{i=1,\dots,I}$
    initialize $\mathbf{v} \leftarrow \mathbf{v}_0$
    for $i = 1 \rightarrow I$ do
        Obtain $\mathbf{u}^*, \mathbf{v}^*$ by running Algorithm 2 with parameters $(Q, \alpha_i, \beta_i, 1, 1)$ and initialization $\mathbf{v}$.
        $S = \{v : \mathbf{u}^*(v) \neq 0\}$ and $T = \{v : \mathbf{v}^*(v) \neq 0\}$
        d-Ncut$_i \leftarrow$ d-Ncut$(S, T)$
        $(S^i, T^i) \leftarrow (S, T)$, $\mathbf{v} \leftarrow \mathbf{v}^*$
    end for
    return $S = S^j$ and $T = T^j$, where $j = \arg\min_i$ d-Ncut$_i$
```

The algorithm requires an initialization vector $\mathbf{v}_0$ and a grid for the sparsity level parameters from users. The initialization vector $\mathbf{v}_0$ can be set as $\mathbf{1}_{\{v_i\}}$ with a randomly picked v_i with nonzero degree or simply picking the node with the largest in-degree. The choice of grid of sparsity levels is critical to discover sensible *ADC*s but it is not trivial to construct. We provide general guide lines for designing a grid.

- Make $\|(\alpha_{i+1}, \beta_{i+1}) - (\alpha_i, \beta_i)\|$ small enough to obtain gradual change of $(\mathbf{u}^*, \mathbf{v}^*)$.

- The default ratio of α and β is one, which provides similar ratio of source parts and terminal parts to the ratio of the whole network. One may provide different ratio if there is prior knowledge of unbalanced communities.

- The choice of grid should reflect the balance between the computational burden and the quality of communities.

In practice, one can check the d-Ncut values of several solution paths with a fine grid to obtain rough idea of the reasonable range and fineness of the grid for the dataset at hand. We will revisit this issue in more detail in Section 6.

4.2 Community Harvesting Algorithm

In order to identify all communities in a directed network, we propose to apply Algorithm 3 repeatedly through a community harvesting scheme. The idea of community extraction has been discussed in Zhao et al. (2011), in which a modularity based method are proposed. Unfortunately, this class of methods based on modularity is NP hard to solve. Starting with the graph Laplacian matrix Q of the full network, we first apply Algorithm 3 to identify a $ADC(S,T)$. Then all entries in Q that corresponds to (S,T) are set to zero and we reapply the same algorithm to the reduced Q matrix with a different initialization to identify the next ADC. Procedure continues until Q has no significant ADCs. Typically, indication of no more significant ADC appears as a continued sequence of small ADCs with very few nodes ($3 \sim 4$). We call this procedure *harvesting* since it extracts one component from the network at a time. Algorithm 4 summarizes the harvesting algorithm.

Algorithm 4 Community Harvesting Algorithm

```
Require: $Q$, i = 1
    repeat
        Obtain $S,T$ using Algorithm 3 with $Q$ and $\mathbf{1}_{\{v_i\}}$ ($v_i$ is a randomly chosen positive
    degree node of $Q$)
        $Q(\mathbf{1}_S, \mathbf{1}_T) \leftarrow 0$
        $S_i \leftarrow S$, $T_i \leftarrow T$
        $i \leftarrow i + 1$
    until $Q$ has no significant $ADC$s
    return $\{(S_j, T_j)\}_{j=1,\ldots,i}$
```

We point out that all harvested ADCs results in a partition of all edges. The edges of directional components are likely to belong meaningfully large ADCs while external edges are likely to compose trivial ADCs that have only few edges. Thus, we recommend to ignore too small ADCs in the context of communities desired. As for the nodes, the harvesting algorithm does not necessarily result in a partition of source nodes (terminal nodes), since the source parts (terminal parts) of different ADCs may have overlaps. However, if a graph has strong community structure, source parts and terminal parts of meaningfully large ADCs only have negligible overlaps.

We also want to stress that the harvesting algorithm takes different approach from the ones in sparse SVD algorithms for obtaining multiple sparse singular vectors. Witten et al. (2009) and Lee et al. (2010) use the residual matrix, $Q - s\mathbf{u}\mathbf{v}^t$ where s is pseudo singular value, to obtain multiple sparse singular vectors. This approach does not fit to our purpose because only the principal singular vector of a sub-matrix is required for a directional component. In addition, harvesting algorithm increases the sparsity of Q whereas the other approach makes the residual matrix more dense.

5. Simulation Studies

In this section, we evaluate the performance of the two harvesting algorithms, L_0-harvesting and EN-harvesting under various settings of community structure. In addition to the harvesting algorithms, the DI-SIM algorithm is included for the sake of comparison. We find that, in addition to the proportion of external edges, the average degree and the size of communities are also important factors determining community detection accuracies.

To generate networks with different types of community structures, we follow a benchmark model proposed by Lancichinetti et al. (2008), referred as the LFR model. The LFR model is based on a restricted version of the stochastic block model where each node has a probability of being connected to nodes in the same community and another probability of being connected to nodes in other communities. This model is originally developed for undirected networks but it has been extended to directed networks by Lancichinetti and Fortunato (2009b). The LFR model introduces heterogeneous distributions of degree and community sizes. The out-degrees of all nodes are almost constant while the in-degrees follow a power law distribution. This model is more realistic than GN benchmark of Girvan and Newman (2002) in applications like citation networks and online social networks.

In our study, we generate networks from the LFR model with $n = 1000$ nodes, whose in-degrees follow a power law (with decay rate $\tau_1 = -2$) with maximum at $k_{max} = 50$. The sizes of the communities in each network are assumed to follow a power law with a decay rate $\tau_2 = -1$ and the sizes of source part and terminal part are the same. We vary three sets of parameters of LFR model to control different aspects of the simulated networks:

- Range of community sizes is set at two levels through a pair of parameters $(SZ_{\omega=1}(C)_{\text{min}}, SZ_{\omega=1}(C)_{\text{max}})$:
  $(40, 200)$ for big communities and $(20, 100)$ for small communities;
- Average degrees (in-degree and out-degree) k for all nodes are set at three levels:
  $\{5, 10, 20\}$ for sparse, median and dense networks;
- Proportion of external edges μ for all nodes is set at three levels:
  $\{0.05, 0.2, 0.4\}$.

Before providing the details of simulation results, we show an example of the simulated network and results of the three community detection methods in Figure 5. This network with big communities, $(SZ_{\omega=1}(C)_{\text{min}}, SZ_{\omega=1}(C)_{\text{max}}) = (40, 200)$, is generated with parameters $k = 20$ and $\mu = 0.1$. Rows of matrix correspond to source nodes and columns correspond to terminal nodes while each dot in the plot represents an edge. The top left panel is the adjacency matrix of the simulated network. The rest of panels present community structures found by the three algorithms in comparison. By the design of the DI-SIM algorithm, it provides two unrelated partitions for rows and columns. In contrast, the harvesting algorithms recover directional components by collecting edges of each component and they indeed showed almost perfect recovery in this example.

5.1 Simulation Results

Back to the full simulation, we measure the accuracy of community detection results by a mutual information based criterion that was proposed by Lancichinetti et al. (2009) and used for the comparison of various community detection algorithms done by Lancichinetti and Fortunato (2009a). One advantage of this criterion is its ability to handle overlapping communities, see details in the appendix of Lancichinetti et al. (2009). As like other mutual

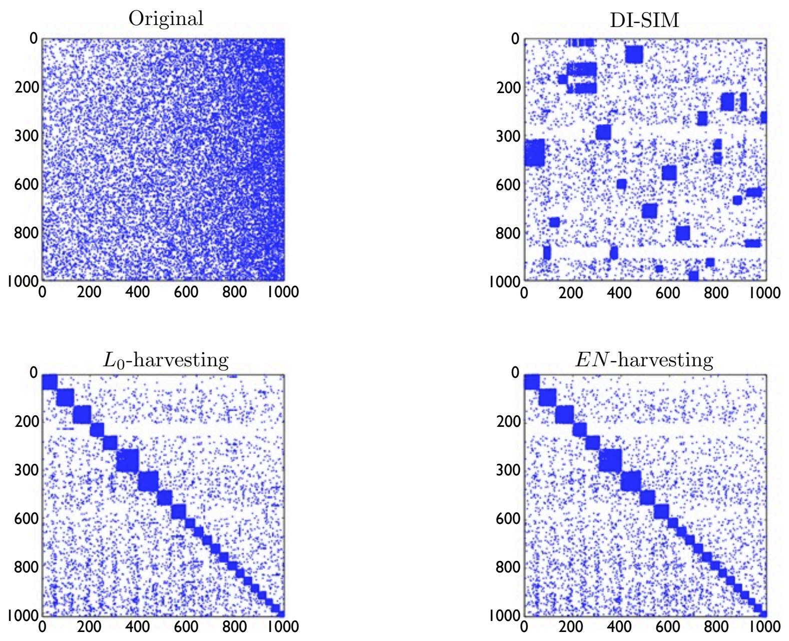


Figure 5: Example random matrix generated by LFR benchmark and the results of DI-SIM algorithm and harvesting algorithms (top right: DI-SIM, bottom left: SVD-L0, bottom right SVD-EN).

information based criterion, the accuracy measure has maximum value one for perfect match and has minimum value zero for community assignment that is independent of the truth.

The DI-SIM algorithm results in two different partitions of nodes, thus we calculate their accuracies separately with respect to the true partitions and average them. In order to compare harvesting algorithms with the DI-SIM algorithm, we calculate the accuracies of harvesting algorithms based on source parts and terminal parts instead of computing based on edge. The source parts and terminal parts of the harvested ADCs are used to measure the accuracies separately and again the accuracies are averaged.

When applying the DI-SIM algorithm, we assume the true number of communities N_C is known. We compute the first N_C singular vectors of Q and apply the k-means algorithm with N_C clusters on the left and right singular vectors separately. One hundred random initialization for k-means algorithm is applied and the result minimizing the within-cluster sums of point-to-cluster-centroid distances is reported as the final result.

For harvesting algorithms, the bottom-up strategy is used accompanied with v_0 being the node of largest in-degree at each harvesting. The sparsity levels for source part and terminal part are set to the same value, $\lambda = 1$ in (4) and $\alpha = \beta$ in (9), and the grids of them are determined without using the knowledge of the size of communities. The grid of sparsity levels for L_0 penalty, η, contains 100 equally spaced points in a range$[10^{-2}, 10^{-5}]$ and the grid of sparsity levels for EN penalty, α, includes 100 equally spaced points in a range $[0.1, 0.98]$. The end-points of the grid are selected to cover *ADC*s sized from two nodes to about the half of nodes in random networks.

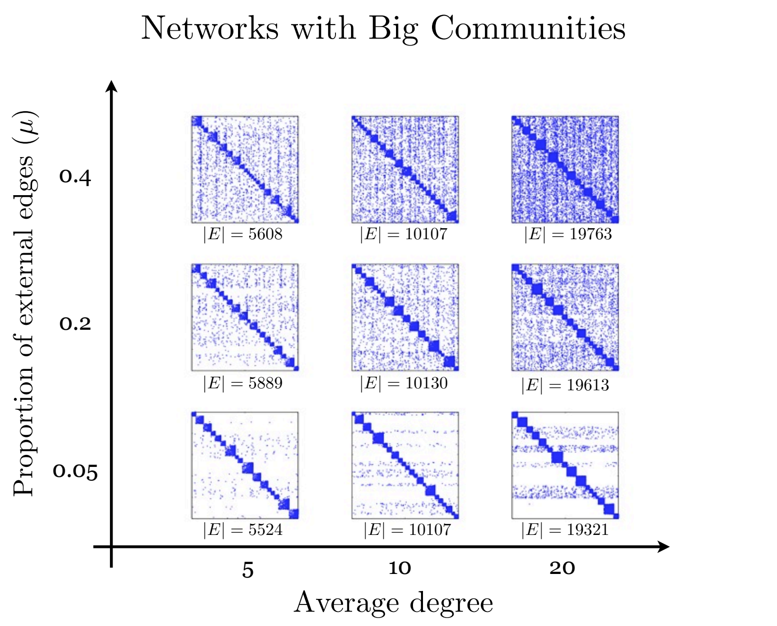
In this simulation, we generate **ten** random networks under each of the eighteen ($2\times3\times3$) parameter combinations and the average of mutual information based accuracy of each algorithm is reported. The results for networks with large communities and those with small communities are reported in Figure 6 and Figure 7 respectively. In these figures, each of the nine panels on the left side visualizes one of the ten generated networks, while the bar chart on the right side shows the corresponding accuracies from three different algorithms. Recall that the range of the accuracy measure is $[0, 1]$ and the larger the value, the better the accuracy.

The results for big communities in Figure 6 show that the harvesting algorithms report almost perfect community detection when nodes has average degree of 10 or 20 and μ is 0.2 or 0.05. The networks with such average degree and μ correspond to the strong community structure that ensures D-connectivity of the nodes in true directional components and relatively small fraction of external edges. The *EN*-harvesting shows better performance than the L_0-harvesting in the region of strong community structure. Moreover, the *EN*-harvesting gives almost the perfect result in the setting of $\mu = 0.4$ and average degree 20. As we mentioned in Figure 5 , the DI-SIM algorithm fails to give a perfect result even for the high average degrees. However, the DI-SIM algorithm gives better results than harvesting algorithms in the region of relatively weak community structures.

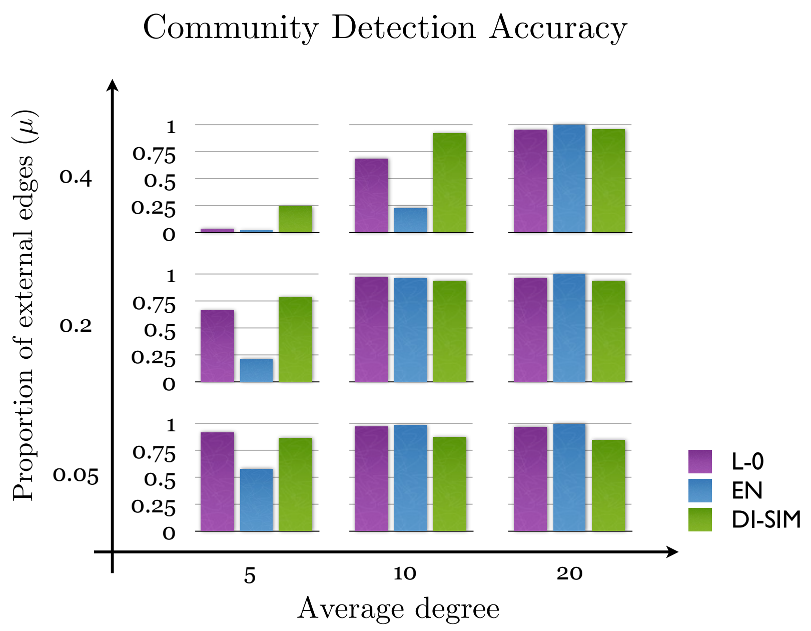
The accuracies for detecting small communities change slightly from the ones of big communities (Figure 7). The accuracies of the other two methods have decreased for the region of strong community structures. The k-means algorithm in DI-SIM algorithm seems to be less accurate for the larger number of clusters in the setting of small communities. However, the accuracy of the *EN*-harvesting algorithm with elastic-net penalty is similar to the results of the setting of big communities.

Closer investigation revealed that sources of the loss in accuracies are quite different for the harvesting algorithms and the DI-SIM algorithm. The loss of accuracies of the DI-SIM algorithm mainly came from some clusters dividing true communities. Unfortunately, it is not straight forward to come up with further improvement of such clusters. In contrast, the loss of accuracies of harvesting algorithms mostly come from several *ADC*s merging true communities. In such a case, it is possible to be further improved by rather simple post-processing, for example, applying a harvesting algorithm one more time on the *ADC* with finer grid of sparsity levels.

In our experiment, we also find that the performance of harvesting algorithms is as good as that of *Infomap*, which shows the best performance in the report of Lancichinetti and Fortunato (2009a). However, the performance of Infomap grounds on the assumption that the true communities have the same source part and terminal part $S = T$ and the performance can be dramatically dropped without the assumption. In contrast, harvesting algorithms do not require the assumption on the true communities since the source part and the terminal part of a directional component may be totally different.

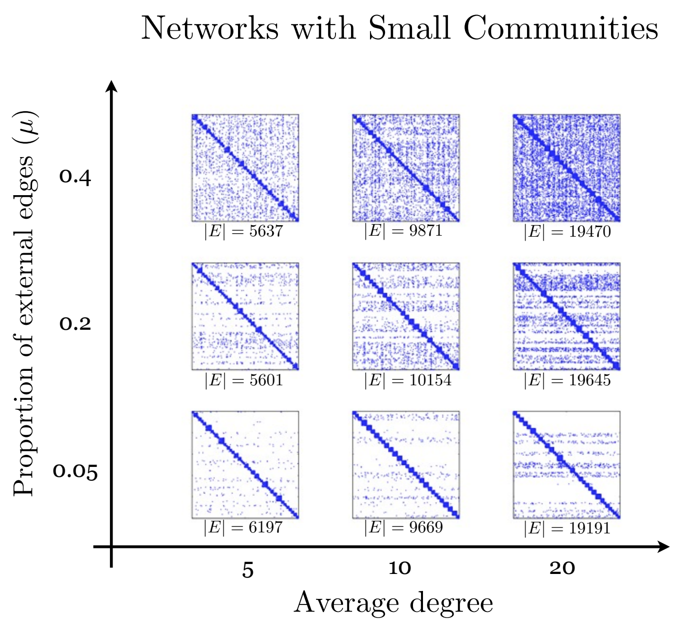


(a) Adjacency matrices of networks with big communities. Rows and columns are arranged by the true communities.

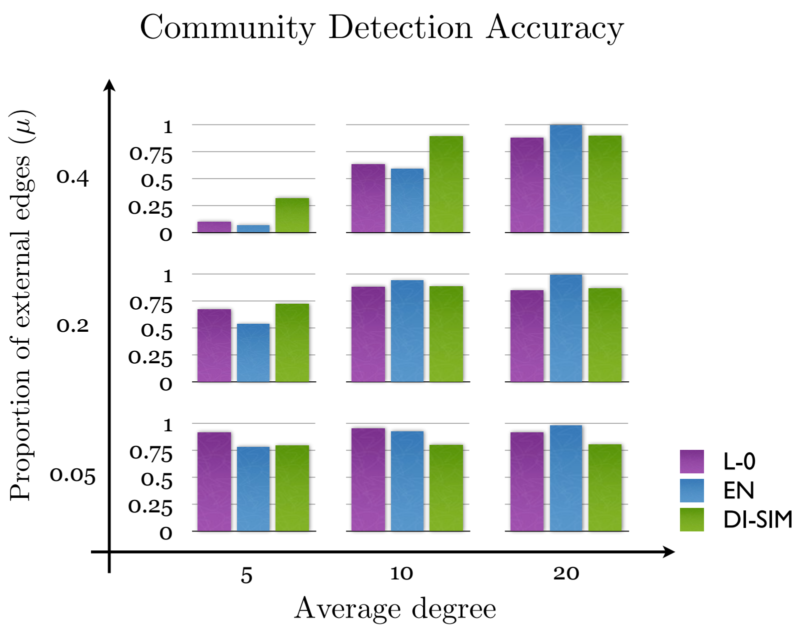


(b) Community detection accuracy of three tested algorithms.

Figure 6: Accuracies of three algorithms, DI-SIM, L_0-harvesting and EN-harvesting, in nine different settings of the community structure. The x-axis indicates average degree and the y-axis indicates the proportion of external edges. The left panel shows an example network at each setting. The accuracies are displayed as bar charts in the right panel. The size of communities ranges in $40 \sim 200$.



(a) Adjacency matrices of networks with small communities. Rows and columns are arranged by the true communities.



(b) Community detection accuracy of the three algorithms.

Figure 7: Accuracies of three algorithms, DI-SIM, L_0-harvesting and EN-harvesting, in nine different settings of the community structure. The x-axis indicates average degree and the y-axis indicates the proportion of external edges. The left panel shows an example network at each setting. The accuracies are displayed as bar charts in the right panel. The size of communities ranges in $20 \sim 100$.

5.2 Computation Time

One driving motivation of this paper is the scalability of community detection algorithms on large or massive networks. In addition to the study of accuracies, we investigate the computation requirements of different algorithms in this simulation study. The algorithms are implemented in MATLAB on a linux machine (2× Six Core Xeon X5650 / 2.66GHz / 48GB).

We gradually increase the number of nodes in networks, $10^3, 5 \times 10^3, 10^4, 10^5$, to see the scalability of the three algorithms in comparison. Networks are generated from the LFR benchmark with the same settings of big communities (size ranged from 40 to 200) with $\mu = 0.2$ and $k = 20$. Upon the fixed range of the size of communities, the number of communities in the network linearly increase as the number of nodes increases.

The harvesting algorithms were performed in the same way as described in Section 5.1 except that the harvesting algorithms run until they find ADCs as many as the number of true communities. The computation time for the DI-SIM algorithm is split into two parts, one for singular value decomposition and the other for the k-means algorithm with 100 random initialization.

Among the three algorithms, the L_0-harvesting algorithm is the fastest for large networks (Table 1). The EN-harvesting algorithm takes roughly two or three times of more computation, which is consistent with the observation that the EN-harvesting does more extensive search of candidate ADCs. In response to more computation time, the accuracy of EN-harvesting method was significantly higher than that of L_0-harvesting for all cases; 0.999 for EN-harvesting and $0.88 \sim 0.92$ for L_0-harvesting. Compared to harvesting algorithms, the DI-SIM algorithm does not scale well to large networks with many communities.

$\|N\|$	# of communities	L_0	EN	DI-SIM
10^3	19	16.9	43.8	$0.08 + 4.7$
5×10^3	103	72	318	$3 + 809.5$
10^4	201	327	807	$14 + 46 \times 100 \approx 4.6 \times 10^4$
10^5	1986	41,400	86,160	$10{,}427 + 38{,}880 \times 100 \approx 4.0 \times 10^6$

Table 1: Computation times (in seconds). Four sample networks are generated for four different numbers of nodes with the common settings $\mu = 0.2$ and average degree 20.

6. Applications

In this section, we apply the proposed harvesting algorithms to two highly asymmetric directed networks, a paper citation network and a social network. Paper citation networks are highly asymmetric directed networks because of their temporal structure; a paper can cite only existing papers. The social network used in this application is highly asymmetric due to a small fraction of popular users with a high fraction of total in-degrees. We show that harvesting algorithms can capture the communities reflecting two different roles of nodes even in such highly asymmetric directed networks.

Before reporting details of the experiments, we first address the issue of deciding the range of sparsity levels for the harvesting algorithms in practice. Since we have limited knowledge on the size of communities, a wide range of sparsity levels is typically necessary. However, we observed that given the wide range of sparsity levels we may end up with a small number of big communities. In order to capture small but meaningful communities, we propose to stop Algorithm 3 early if the d-Ncut value reaches a local minimum in the grid of sparsity levels. A simple implementation used in this experiment is to stop searching for candidate ADCs if the d-Ncut value of the current candidate-ADC bounces up to higher than s_p times of the minimum d-Ncut value of the previously detected candidate-ADCs. In addition, we pre-specify a bound s_l on the desired d-Ncut value so we only stop early at communities that report a d-Ncut value lower than s_l.

6.1 Cora Citation Network

We first apply both harvesting algorithms to the Cora citation network, a directed network formed by citations among Computer Science (CS) research papers [3]. In this experiment, we use a subset of the papers that have been manually assigned into categories that represent 10 major fields in computer science, which is further divided into 70 sub-fields. This leads to a network of 30,228 nodes and 110,654 edges after removing self-edges. In this citation network, only 5.4% of edges are symmetric. The average degree is 3.66, which is relatively low. We also found that 2345 nodes had error labels and they were put into 11th category.

The algorithms are applied with a bottom-up approach such that the algorithms start at the terminal nodes with the largest in-degree among un-harvested nodes at each harvesting run. The sparsity levels parameter η in the L_0-harvesting takes values decreasingly in a grid $\{\exp(-k) : k = 10 + i(12/200), i = 1, \ldots, 200\}$. Similarly, the sparsity levels parameter α in the EN-harvesting takes values decreasingly in a grid $\{\frac{1}{1+\exp(k)} : k = -1 + i(11/200), i = 1, \ldots, 200\}$. The nonlinear decreasing setup helps obtain gradual changes of the size of the candidate-ADCs at low sparsity levels. Early stopping parameters are set to $s_p = 1.5$ and $s_l = 0.3$.

For both harvesting algorithms, we first provide a summary of the first twenty ADCs discovered. We name the ADC obtained in the L_0-harvesting ADC^{L_0} and the ones obtained by the EN-harvesting ADC^{EN}. We report the sizes of source part and terminal part, the number of edges and d-Ncut value for each ADC in Table 2. Out of total 110,654 edges, the first twenty ADC^{L_0}s cover 107,493 edges and the first twenty ADC^{EN}s cover 95,529 edges. We observe that larger communities are likely to be captured in the first several ADCs because the initial value $\mathbf{v}_0$ is more likely to be a member of large communities. Overall, we find ADC^{L_0}s are better than ADC^{EN}s based on the comparison of d-Ncut values. This result is consistent with the result of simulations in Section 5 that L_0-harvesting performs better in networks of low average-degrees.

6.1.1 Comparison to DI-SIM and Infomap

To evaluate the performance of harvesting algorithms, two existing community detection algorithms are also applied for comparisons. First, the DI-SIM algorithm (Rohe and Yu, 2012) is applied, assuming the number of communities equals to the number of major-fields in CS, which is ten. For the k-means step of the DI-SIM algorithm, ten random

3. `http://people.cs.umass.edu/~mccallum/data.html`

Order	$\|S\|$	$\|T\|$	$\|E\|$	d-Ncut
1	1911	1410	6706	0.2261
2	4422	2927	20895	0.1836
3	1213	809	4526	0.1619
4	12863	8872	65850	0.1265
5	756	550	2590	0.2076
6	72	50	187	0.1764
7	98	62	448	0.0787
8	446	282	1506	0.3451
9	55	37	164	0.1039
10	99	65	344	0.3182
11	475	331	1366	0.2757
12	119	78	379	0.2865
13	181	123	559	0.1792
14	22	15	55	0.2023
15	86	53	244	0.1541
16	38	26	125	0.1955
17	24	19	56	0.1934
18	52	35	179	0.1417
19	281	221	1016	0.3544
20	29	28	97	0.3584

(a) First 20 ADC^{L_0} of Cora network.

Order	$\|S\|$	$\|T\|$	$\|E\|$	d-Ncut
1	6903	4395	29847	0.2384
2	3	3	3	0.2929
3	5	6	15	0.0905
4	1707	1047	6696	0.3066
5	9736	6106	48374	0.2700
6	733	511	3464	0.3578
7	616	361	2714	0.1795
8	481	353	1955	0.3467
9	167	115	820	0.2078
10	8	9	16	0.3246
11	176	124	484	0.3064
12	32	20	115	0.3742
13	53	42	249	0.2037
14	1	1	1	0.2929
15	60	46	180	0.1671
16	2	1	2	0.1835
17	33	26	172	0.1985
18	51	44	147	0.3154
19	69	47	234	0.3075
20	18	8	32	0.0985

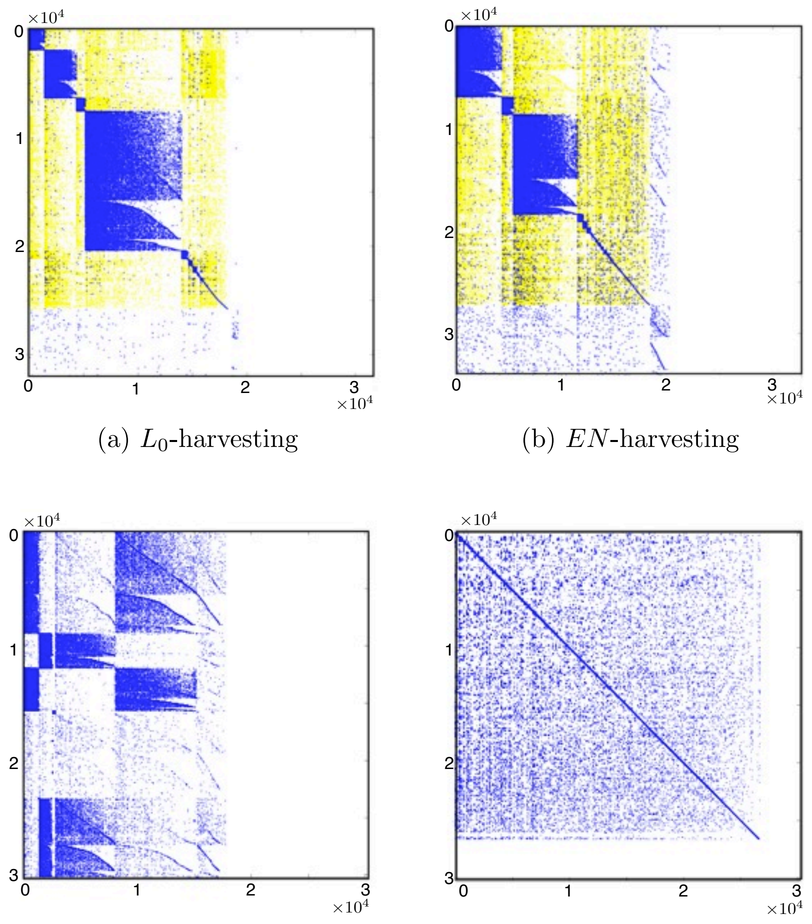
(b) First 20 ADC^{EN} of Cora network.

Table 2: Summary of the first 20 *ADC*s of Cora citation network.

initialization is used. Second, we applied the *Infomap* algorithm of Rosvall et al. (2009), which showed very good performance in LFR benchmark as reported by Lancichinetti and Fortunato (2009a). Next, we first provide a visual comparison of communities detected by these four algorithms in Figure 8.

The visualization of the results of harvesting algorithms through adjacency matrix is not straightforward since nodes may appear more than once due to the possibility of multiple memberships. To see the community structure, the rows and columns are arranged by the source parts and the terminal parts of the twenty approximated directional components and the remaining nodes are appended at the end of rows and columns. Edges are shown as blue dots in the plot. Internal edges of *ADC* appear as blue blocks in the diagonal and all internal edges appear only once in the visualization. Meanwhile, blue dots outside of the blocks are the edges that are not harvested in the first twenty harvesting. As the harvesting goes on, all edges outside of the blocks will eventually append to the diagonal blocks and appear as a thin line at the end. We also use yellow dots to indicate the reappearing internal edges of *ADC*s that appear between blocks because of the multiple memberships of source nodes and terminal nodes.

The lower panels in Figure 8 show results of the methods in comparison. The result from the DI-SIM algorithm is summarized by the adjacency matrix of the Cora citation



(a) L_0-harvesting (b) EN-harvesting

(c) DI-SIM (d) Infomap

Figure 8: Top panels: The results of harvesting algorithms of the Cora citation network. The rows and columns are arranged by the source parts and the terminal parts of the first twenty ADCs and remaining nodes are appended at the end of rows and columns. Bottom panels: Adjacency matrix of the Cora citation network with rows and columns reordered by the result of the DI-SIM algorithm and Infomap.

network with rows and columns reordered by the partitions (Figure 8c). The row of matrix is reordered by the partition of source nodes and the column of matrix is reordered by the partition of terminal nodes. The adjacency matrix rearranged by the communities of *Infomap* is shown in Figure 8d, in which the order of rows and columns are the same. Comparing all four panels, we conclude that the obvious block structure in the plots of L_0-harvesting better represents the community structure in the Cora citation network.

The communities detected by harvesting algorithms have more heterogeneous sizes compared to the communities of Infomap (Figure 8). For example, $\{SZ_1(ADC_i^{L_0})\}_{i=1,\ldots,20}$ range from 50 to 20,000 while the Infomap provide 5312 small communities whose number of nodes is less than 50 and 114 communities whose number of nodes is between 51 and 270. This difference comes from the fact that the harvesting algorithm accounts the asymmetric nature of the citation network while Infomap looks for symmetric communities on the highly asymmetric directed network.

6.1.2 Comparison through Manual Categories

The manually assigned categories of papers in the Cora citation network provided us with extra information to validate the quality of communities detected from different algorithms. Before making comparisons we present a summary of the eleven categories in Table 3. The sizes of category span a large range, from 10,784 papers in Artificial Intelligence to 582 papers in Information Retrieval. Given the categories, we calculate the d-Ncut value of each category to form a baseline. We observe that the values are moderate and ranged from 0.29 to 0.49, which are slightly higher than the d-Ncut values of ADC^{L_0}s.

We investigate the consistency between detected communities from each algorithm and the manually assigned categories. The communities of L_0-harvesting algorithm are reported in detail in Table 4, while the results of other algorithms can be found in Appendix C. The communities are reported by their order of being harvested.

The first five harvested communities are fairy large and reveal interactions among fields of CS. Papers in $ADC_1^{L_0}$ are mainly from two fields, artificial intelligence (AI) and human computer interaction (HCI) and further investigation shows that majority of these papers in AI are in the *vision and pattern recognition* sub-field. $ADC_2^{L_0}$ and $ADC_5^{L_0}$ are dominated by papers from AI and these two communities take up nearly half of the papers of the AI category. $ADC_3^{L_0}$ also embeds an interaction between a sub-field of AI (*theorem proving*) and two sub-fields of data structures algorithms and theory (*formal languages* sub-field and *logic* sub-field). $ADC_4^{L_0}$ is the biggest community that is mainly led by four categories; operating systems, programming, databases and networking.

The rest of those communities are smaller in sizes and each contains less kind of categories. In other words, the small communities have high precision and low recall with respect to the manual categories. Many of small communities are related to AI category and they represent different sub-fields of AI. For example, $ADC_{11}^{L_0}$ corresponds to *speech* sub-field and *natural language processing* sub-field. $ADC_{12}^{L_0}$ mainly covers *knowledge representation* sub-field. There are also meaningful small communities from fields other than

Number	Name of field of CS	Number of papers	d-Ncut
1	Artificial Intelligence	10784	0.2295
2	Data Structures Algorithms and Theory	3104	0.4161
3	Databases	1261	0.3596
4	Encryption and Compression	1181	0.4235
5	Hardware and Architecture	1207	0.4917
6	Human Computer Interaction	1651	0.4727
7	Information Retrieval	582	0.4003
8	Networking	1561	0.3926
9	Operating Systems	2580	0.4305
10	Programming	3972	0.3729
11	Uncategorized	2345	0.7778

Table 3: List of ten fields of Computer Science.

	AI	DSAT	DB	EC	HA	HCI	IR	Net	OS	Prog	Uncategorized
1	770	368	12	152	55	593	9	9	51	88	294
2	4476	93	51	33	43	30	97	10	31	71	305
3	538	274	5	19	66	22	0	34	184	138	163
4	1383	1503	1090	782	600	650	261	1295	2203	3210	1477
5	822	7	1	9	3	47	18	2	7	1	51
6	86	0	0	0	0	0	1	0	0	1	2
7	3	2	0	0	81	0	0	15	0	0	8
8	459	1	1	0	2	1	9	0	3	31	50
9	69	0	0	0	0	0	0	0	0	0	0
10	93	10	0	0	0	0	1	0	0	0	10
11	494	5	0	0	2	17	57	0	0	6	41
12	145	1	0	0	0	2	0	1	0	3	1
13	7	6	0	0	166	1	0	0	4	3	33
14	28	0	0	0	0	0	0	1	1	0	0
15	95	6	0	0	0	0	0	0	0	0	1
16	41	1	0	0	0	0	0	0	3	1	2
17	35	0	0	0	0	0	0	0	0	0	0
18	28	17	1	1	2	0	0	0	0	0	13
19	139	1	0	1	0	3	214	1	0	5	22
20	38	0	0	4	0	0	0	0	0	1	6

Table 4: Number of papers in the first twenty approximated directional components of L_0-harvesting for each category.

AI, for instance, $ADC_{13}^{L_0}$ stands for *logic design* and *VLSI* sub-field of hardware and architecture. Finally, we found that $ADC_{19}^{L_0}$ shows a connection between *natural language processing* sub-field of AI and information retrieval.

The analysis on the composition of manual categories shows that the communities detected by the harvesting algorithm meets our expectations regarding the manual categories. In addition, the discovery of the several large communities and many small communities are consistent with the core-whisker model briefed by Leskovec et al. (2008). We also suspect a possible hierarchical community structure within the large communities but we leave investigations along this direction in our future work.

6.2 Scalability of Harvesting Algorithms in a Large Social Network

The recent emergence of social networks and their wide accessibility greatly enhanced our ability to obtain and deliver information in a much shorter time span. However, the massive size, more than millions of nodes in a modern network, demand scalable algorithms for analysis. Many community detection algorithms that search for optimal partition of nodes does not scale well to these large social networks. On the other hand, the harvesting algorithm detects a community at a time according to the quality of the detected community. In this experiment, we test our harvesting algorithms on one such huge network over which the partitioning approaches may fail, especially when the network is highly asymmetric and the sizes of communities are relatively small.

We use a social network dataset[4] of Tencent Weibo, a micro-blogging website of China. Users in this network may subscribe to news feeds from other users based on their interests. The subscriptions are represented as directed edges between users. This network contains 1,944,589 non-zero degree nodes and 50,655,143 directed edges, which leads to the average out-degree 25. The social network is highly asymmetric and it has only 0.2% of symmetric links. We test the harvesting algorithms implemented in MATLAB on a linux machine (2$\times$ Six Core Xeon X5650 / 2.66GHz / 48GB). The sparsity level parameter in the harvesting algorithms are designed to capture communities with the size in the range of 10 to 2,000,000. The grid of sparsity parameter η in L_0-harvesting is set as $\{\exp(-k) : k = 18 + i(5/50), i = 1, \ldots, 50\}$ and the grid for $\alpha = \beta$ in EN-harvesting is set as $\{\frac{1}{1+\exp(k)}; k = 3 + i(8/50), i = 1, \ldots, 50\}$. The early stopping method is applied with the parameters $s_p = 1.1$ and $s_l = 0.8$.

The computation time to harvest 1000 ADC^{L_0} was about 51 hours while that of harvesting 1000 ADC^{EN} was around 46 hours. On average, each harvesting took about 3 minutes for both algorithms. In addition, approximately 4 $\sim$ 5GB of memory were required for both algorithms. For comparison, a representation of partitioning algorithms, Infomap, is applied with R package *igraph*. Unfortunately, the algorithm has stopped with an error after 56 hours of running.

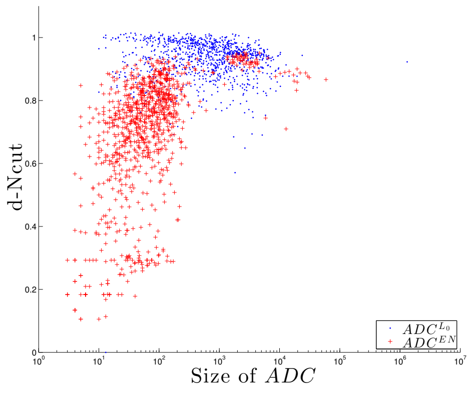
To see the quality of harvested communities, we calculate the d-Ncut values along with the size of ADCs (Figure 9a). Different from the case of Cora citation network, the EN-harvesting reports better d-Ncut values than the L_0-harvesting algorithm does. Especially, the EN-harvesting is able to detect small communities with low d-Ncut values because the constraint (9) is directly applied on the surrogate size of a community, while the optimization (4) in the L_0-harvesting directly penalizes the size. We also verified that good communities are relatively small ($\sim$ 200) in such huge social networks, as reported in Leskovec et al. (2008).

In addition, we find it interesting to compare the size of the source part and that of the terminal part. Specifically, we investigate how much of members are common in both parts. We define *commonality* of a ADC as the ratio of the number of common nodes against the total number of nodes in the union of the two parts. It appears that small communities are more likely to have higher commonality while big communities tends to have low commonality (Figure 9b). The high commonality in small communities implies that the members of small communities play the role of source and terminal at the same time, which is common in small private communities. In contrast, further inspection showed that most of members in big communities play the role of source while only few popular members play the role of terminal. Besides illustration of the scalability of the harvesting algorithms, this empirical observation highlights that our algorithms are capable of incorporating the direction of links and detecting small communities.

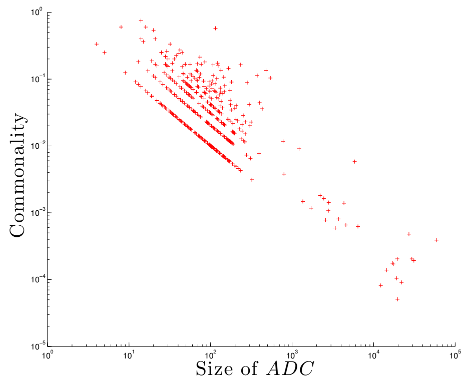
7. Conclusions and Discussion

In this paper, we found that integrating the direction of links plays an important part in the characterization of communities in directed networks. We first introduced a new notion

4. http://www.kddcup2012.org/c/kddcup2012-track1/data



(a)



(b)

Figure 9: Summarize the result of harvesting algorithms on a social network (a) Scatter plot of d-Ncut vs size of ADCs. (b) Scatter plot of commonality vs size of ADC^{EN}s.

of community, *directional components*, that is capable of discerning the two different roles of each node in a community. Such communities are directly connected with the spectrum of the graph Laplacian.

We proposed two regularized SVD based *harvesting algorithms* that sequentially identifie communities that have majority of links within a community while few links are placed between the communities. The regularized SVD method is linearly scalable to the number of nodes and the number of edges in the network, so it is computationally efficient. The L_0-harvesting algorithm showed good performance even in networks having low average-degree. Meanwhile the EN-harvesting algorithm provided great results in detecting relatively small and dense communities. Both simulation studies and applications on real networks show the effectiveness of the proposed harvesting algorithms.

We believe the harvesting algorithm enables genuine analysis on community structures in highly asymmetric directed networks of real applications. Especially, the simple computations that only rely on matrix multiplication and thresholding of vectors lead to further improvement of the algorithm through parallel and distributed computing.

Acknowledgements

This research is partially supported by NSF grant DMS-1007060. The authors would like to thank Dr. Srinivasan Parthasarathy and Dr. Hanson Wang for helpful discussion.

Appendix A. Proofs for Section 3

We want to remark that the definition of directional components does not require the weight of links to be zero or one. Following proofs assume non-negative valued weights.

A.1 Proof of Theorem 3

Proof

The proof of this theorem relies on the properties of Laplacian matrix of A, where the matrix A is

$$A = \begin{bmatrix} 0 & W \\ W^t & 0 \end{bmatrix},$$

$W \in \mathbb{R}^{|\mathcal{S}|} \times \mathbb{R}^{|\mathcal{T}|}$, where $\mathcal{S}$ is the set of source nodes and $\mathcal{T}$ is the set of terminal nodes. The matrix A can be considered as the adjacency matrix of the bipartite graph expression of a directed graph (Zhou et al., 2005; Guimerà et al., 2007). A bipartite graph is denoted by $B = (\mathcal{S}, \mathcal{T}, \mathcal{L})$. $\mathcal{L}$ is the set of all the edges that can be expressed as an ordered pair $(s,t), s \in \mathcal{S}, t \in \mathcal{T}$.

We denote $B(G)$ the bipartite conversion of a directed graph G. We show that a directional component of G is equivalent to a connected component of $B(G)$. First, let us show that a directional component (DC) is a connected component (C) by examining the connectivity and maximality conditions:

- Connectivity: Any pair of nodes ($s \to t$) are connected in $B(G)$.
- Maximality: Assume that there exists a node that is connected to C but not a member of DC. Then there should be a directed edge starting from the node or ended at the node in G. In either case the node is a member of DC. It contradicts to the maximality of DC. Thus there is no such node.

Similarly, we show that a connected component C in $B(G)$ is a directional component DC in G. Any pair of nodes $(s,t), s \in \mathcal{S} \cap B(G), t \in \mathcal{T} \cap B(G)$ is D-connected in G by the connectivity in $B(G)$. Maximality for a directional component is again obtained by using the maximality of C.

A bipartite graph is an undirected graph thus we can apply the proposition 4 of Von Luxburg (2007) that shows us the equivalence between the number of connected components of an undirected graph and the multiplicity of the zero eigenvalue of graph Laplacian matrix of the undirected graph. Let L_{sym} be a normalized graph Laplacian of A, which is defined by

$$L_{sym} = I - Q_A,$$

where,

$$\begin{aligned} Q_A &= D_A^{-\frac{1}{2}} A D_A^{-\frac{1}{2}} \\ &= \begin{bmatrix} 0 & Q \\ Q^t & 0 \end{bmatrix} \end{aligned} \tag{13}$$

and D_A is the diagonal matrix of the row sums of A and it is equal to

$$D_A = \left[\begin{array}{cc} D_r & 0 \\ 0 & D_c \end{array} \right].$$

The proposition 4 of Von Luxburg (2007) says that the multiplicity m of the eigenvalue zero of L_{sym} is equal to the number of connected component in the undirected graph corresponding to A and the eigenspace of zero is spanned by the vectors $\{D_A^{\frac{1}{2}} \mathbf{1}_{C_k}, k = 1, \ldots, m\}$, where $\mathbf{1}_{C_k}$ is the indicator vector for kth connected component.

By the definition of L_{sym}, if λ is an eigenvalue of L_{sym} then $1 - \lambda$ is an eigenvalue of Q_A. It follows that the eigenvalue zero of L_{sym} corresponds to the eigenvalue one of Q_A. In fact, one is the principal eigenvalue of Q_A because the eigenvalue zero is the smallest eigenvalue of L_{sym}, which is a non-negative definite matrix.

By the standard result about the eigenvalues of Q_A and the singular values of Q (see Horn and Johnson, 1994, chap. 3), the principle singular value of Q is the principle eigenvalue of Q_A which is one. A vector $D_A^{\frac{1}{2}} \mathbf{1}_{C_k}$ can be broken into two vectors $D_r^{\frac{1}{2}} \mathbf{1}_{S_k} \in \mathbb{R}^{|\mathcal{S}|}, D_c^{\frac{1}{2}} \mathbf{1}_{T_k} \in \mathbb{R}^{|\mathcal{T}|}$, where $D_r^{\frac{1}{2}} \mathbf{1}_{S_k}$ is the first $|\mathcal{S}|$ entries of $D_A^{\frac{1}{2}} \mathbf{1}_{C_k}$ and $D_c^{\frac{1}{2}} \mathbf{1}_{T_k}$ is the last $|\mathcal{T}|$ entries of $D_A^{\frac{1}{2}} \mathbf{1}_{C_k}$. By (13), the two vectors satisfy

$$\begin{cases} D_r^{\frac{1}{2}} \mathbf{1}_{S_k} & = Q D_c^{\frac{1}{2}} \mathbf{1}_{T_k} \\ D_c^{\frac{1}{2}} \mathbf{1}_{T_k} & = Q^t D_r^{\frac{1}{2}} \mathbf{1}_{S_k}, \end{cases}$$

as one can find in Dhillon (2001). $\{D_r^{\frac{1}{2}} \mathbf{1}_{S_k}, k = 1, \ldots, m\}$ is a set of orthogonal vectors since S_k's are exclusive. The same argument holds for $\{D_c^{\frac{1}{2}} \mathbf{1}_{T_k}, k = 1, \ldots, m\}$. Thus, the pairs of vectors $\{(D_r^{\frac{1}{2}} \mathbf{1}_{S_k}, D_c^{\frac{1}{2}} \mathbf{1}_{T_k}), k = 1, \ldots, m\}$ span the singular space of the singular value one of Q.

■

A.2 Lemmas for Proposition 4

Even though the directional components are initially developed for a decomposition of a directed graph, the same decomposition can be done to a non-negative matrix B if B is treated as a weight matrix of a directed graph whose nodes are mapped from all rows and columns. We call a sub-matrix of B a directional-component block if the sub-matrix is corresponding to a directional component of the directed graph generated from the weight matrix B. Following Lemmas are required for the proof of Proposition 4.

Lemma 10 *Let $B \in \mathbb{R}^m \times \mathbb{R}^n$ be a matrix of a single directional-component block. Then, $B^t B$ is irreducible.*

Proof Notice that B is a nonnegative matrix and its row sums and column sums are all positive by the condition. The single directional-component block can be considered as a weight matrix of a directed graph, say G ,with a single directional component. By Lemma 8.4.1 in Horn and Johnson (1990), it suffices to show that $(I + B^tB)^{n-1} > 0$ for the irreducibility. B^tB can be treated as the weight matrix of an undirected graph, $H = (\mathcal{T}, W_s)$, where $\mathcal{T}$ is the set of terminal nodes of G and W_s is the weight matrix obtained by $W_s(i,j) = (B^tB)_{(i,j)}$. H is a connected component by the D-connectivity of nodes in G. Then, $(I + B^tB)$ is the weight matrix of a connected component since it is the weight matrix of a undirected graph, say J, that is obtained by adding self edges to all the nodes of H. To obtain $(I + B^tB)^{n-1} > 0$, we need to show any nodes in J can reach any nodes in J at $n-1$ steps. Since H is a connected component, the diameter of H is at most $n-1$ and thus any nodes can be reached within $n-1$ steps from any nodes in J. Following the self edges as many as needed to make exact $n-1$ steps, the desired result is obtained. ■

Lemma 11 *For any submatrix of Q, say Q_s, the largest singular value of Q_s is less than or equal to one ($\sigma_1(Q_s) \leq 1$), and the equality holds only when Q_s includes directional-component blocks.*

Proof

Without loss of generality we may assume Q has only one directional component since directional-component blocks can be considered separately for the spectral property. If Q_s includes the directional component block of Q, then $\sigma_1(Q_s) = 1$ by Theorem 3. Thus we showed that $\sigma_1(Q_s) = 1$ if Q_s includes a directional component block.

Notice the fact that any sub-matrix can be obtained by deleting columns and rows of Q sequentially. Q_s does not include the directional component block of Q only if the deleted columns or rows are not zero vectors. Even though Q_s may have more than one directional-component block, we only consider the directional-component block with the largest singular value since $\sigma_1(Q_s)$ is determined by it.

We begin with Q_s that is obtained by deleting a non-zero column of Q. Theorem 3.1.2 of Horn and Johnson (1994) tells that $\sigma_1(Q_s)$ is a solution of the optimization problem,

$$\sigma_1(Q_s) = max_{\|x\|_2=1} \|Q_s x\|_2.$$

Notice that $\sigma_1(Q_s)^2 = \lambda_1(Q_s^t Q_s)$, where $\lambda_1(B)$ is the largest eigenvalue of a symmetric matrix B.

$Q_s^t Q_s$ is irreducible by Lemma 10. Applying theorem 8.4.4 of Horn and Johnson (1990) to $Q_s^t Q_s$ tells us that there exist a vector $x > 0$, $\|x\|_2 = 1$ such that $\lambda_1(Q_s^t Q_s) = \|Q_s x\|_2^2$. It deduces $\sigma_1(Q_s) = \|Q_s x\|_2$.

Now, we show that $\sigma_1(Q_s) < \sigma_1(Q)$. Consider W_s, a sub-matrix of W, which is obtained by taking the same index of Q_s in Q. W_s can be normalized to $\tilde{Q}_s$, in the same way that the full weight matrix W is normalized to Q. We know that $\sigma_1(\tilde{Q}_s) = 1$ and

$$\tilde{Q}_s = Q_s + E$$

where E is a matrix that has non-negative entries and at least one positive entry. Basically, this is because deleting a column of Q decreases at least one entry of the row sum vector.

The existence of a positive entry in E implies that

$$\begin{aligned}\sigma_1(Q_s) &= max_{\|x\|_2=1}\|Q_s x\|_2 \\ &< max_{\|x\|_2=1}\|(Q_s+E)x\|_2 \\ &= \sigma_1(\tilde{Q}_s) = 1\end{aligned}$$

The inequality comes from $x > 0$ and $\|Q_s x\|_2 < \|(Q_s+E)x\|_2$.

The case of deleting a row of Q can be done in the same way by taking the transpose of Q since singular values are invariant under taking a transpose.

Now we know that a deletion of a column or a row of Q result in a reduction of the largest singular value. Deleting more than one columns or rows can be done sequentially, therefore, $\sigma_1(Q_s) < \sigma_1(Q)$ if Q_s does not include at least a directional-component block of Q. ■

A.3 Proof of Proposition 4

Proof

Given a solution $\mathbf{u}, \mathbf{v}$ and $C(S,T)$, notice that $\|\mathbf{u}\|_0 = |S|$ and $\|\mathbf{v}\|_0 = |T|$. We obtain a matrix $Q(C(S,T))$ by setting the rows and columns of Q that are not in S, T to zero vectors. Then, (5) can be written as

$$\max_{S,T} \sigma_1(Q(C(S,T))) - \eta SZ_\omega(C(S,T))$$

Lemma 11 tells us that $\sigma_1(Q(DC_1))$ is equal to 1 and that is one of the largest among $\{\sigma_1(Q(C)) | SZ_\omega(C) \geq SZ_\omega(DC_1)\}$. Thus all C such that $SZ_\omega(C) > SZ_\omega(DC_1)$ can be excluded from the consideration. Thus, to make DC_1 the solution, we need to find $\eta > 0$ satisfying

$$1 - \eta SZ_\omega(DC_1) > \sigma_1(Q(C)) - \eta SZ_\omega(C),$$

for all C such that $SZ_\omega(C) < SZ_\omega(DC_1)$. After an arrangement of above inequality, η should satisfy

$$\eta < \frac{1 - \sigma_1(Q(C))}{SZ_\omega(DC_1) - SZ_\omega(C)}. \tag{14}$$

$SZ_\omega(DC_1) - SZ_\omega(C) > 0$ by the condition of C and $1 - \sigma_1(Q(C)) > 0$ by Lemma 11, thus setting the right hand side of (14) as ϵ finishes the proof. ■

A.4 Proof of Theorem 8

Proof Express the objective function and the constraints by using a Lagrangian multiplier,

$$\min_{\mathbf{u},\lambda} - \mathbf{u}^t\mathbf{z} + \lambda((1-\alpha)\|\mathbf{u}\|_2^2 + \alpha\|\mathbf{u}\|_1). \tag{15}$$

Then, differentiate the objective function in (15) by $\mathbf{u}$ and set it to zero,

$$-\mathbf{z} + \lambda(2(1-\alpha)\mathbf{u} + \alpha\Gamma) = 0, \tag{16}$$

where $\Gamma_i = \text{sign}(u_i)$ if $u_i \neq 0$, otherwise, $\Gamma_i \in [-1, 1]$. The Karush-Kuhn-Tucker (KKT) conditions require $\lambda((1-\alpha)\|\mathbf{u}\|_2^2 + \alpha\|\mathbf{u}\|_1 - c_1) = 0$. If $\lambda > 0$, the solution is

$$\hat{\mathbf{u}} = \frac{S(\mathbf{z}, \lambda\alpha)}{2\lambda(1-\alpha)}. \tag{17}$$

λ can be zero, if the solution is not on the boundary of the contraint. But it does not happen unless $\mathbf{z}$ is a zero vector. Thus, $\lambda > 0$ is chosen so that $\hat{\mathbf{u}}$ satisfies the KKT condition.

$$\begin{aligned} &(1-\alpha)\left\|\frac{S(\mathbf{z}, \lambda\alpha)}{2\lambda(1-\alpha)}\right\|_2^2 + \alpha\left\|\frac{S(\mathbf{z}, \lambda\alpha)}{2\lambda(1-\alpha)}\right\|_1 = c_1 \\ \Rightarrow &\frac{1}{(2\lambda)^2(1-\alpha)}\sum_{i=1}^{k-1}(|z|_{(i)} - \lambda\alpha)^2 + \frac{\alpha}{2\lambda(1-\alpha)}\sum_{i=1}^{k-1}(|z|_{(i)} - \lambda\alpha) = c_1 \end{aligned} \tag{18}$$

where k satisfies $|z|_{(k)} \leq \lambda\alpha < |z|_{(k-1)}$. Denote the threshold level, $d = \lambda\alpha$, then (18) becomes

$$\left(\frac{1}{4d^2}\sum_{i=1}^{k-1}(|z|_{(i)} - d)^2 + \frac{1}{2d}\sum_{i=1}^{k-1}(|z|_{(i)} - d)\right) = c_1\frac{1-\alpha}{\alpha^2}, \tag{19}$$

where k satisfies $|z|_{(k)} \leq d < |z|_{(k-1)}$. Using Lemma 9, one can determine the threshold level d of (19) by setting $\mathbf{z}$ and $c = c_1\frac{1-\alpha}{\alpha^2}$. In the actual algorithm, the value of λ is not required for getting the solution. For the record, the corresponding λ is presented,

$$\lambda = \frac{1}{\alpha}\left(\frac{\sum_{i=1}^{\hat{k}}|z|_{(i)}^2}{4(c_1\frac{1-\alpha}{\alpha^2}) + \hat{k}}\right)^{\frac{1}{2}}. \tag{20}$$

■

A.5 Proof of Lemma 9

Proof As the first step, let us show that $G_{\mathbf{z}}(\cdot)$ is monotone decreasing function. We need to show that if $d_1 > d_2$, then $G_{\mathbf{z}}(d_1) < G_{\mathbf{z}}(d_2)$. Denote $k(d)$ for the k corresponding to the threshold level d. The first term of (11) is monotone decreasing of d because

$$\frac{1}{4d_2^2}\sum_{i=1}^{k(d_2)-1}(|z|_{(i)} - d_2)^2 > \frac{1}{4d_1^2}\sum_{i=1}^{k(d_1)-1}(|z|_{(i)} - d_2)^2 \tag{21}$$

$$> \frac{1}{4d_1^2}\sum_{i=1}^{k(d_1)-1}(|z|_{(i)} - d_1)^2. \tag{22}$$

The first inequality comes from $k(d_2) \leq k(d_1)$ and $d_1 > d_2$. The second inequality comes from $d_1 > d_2$. The second term of (11) can be done in the similar way and the desired result is obtained.

As the second step, we find the approximated solution of d from the set of $\{|z|_{(i)}\}_{i=1\ldots n}$. By plugging in $|z|_{(i)}$ to d in the increasing order of i, we can find $\hat{k}$ such that $G_{\mathbf{z}}(|z|_{(\hat{k})}) \leq c$, $G_{\mathbf{z}}(|z|_{(\hat{k}+1)}) > c$ by the monotonicity of $G_{\mathbf{z}}(\cdot)$ and being c in the range of $G_{\mathbf{z}}(\cdot)$. This computation can be done efficiently by computing two cumulative sums, $\{\sum_i^k |z|^2_{(i)}\}_{k=1\ldots n}$ and $\{\sum_i^k |z|_{(i)}\}_{k=1\ldots n}$, in the increasing order of i until $\hat{k}$ is obtained. A simple algorithm for finding k that satisfies condition (12) in this Lemma is provided in Algorithm 5.

Algorithm 5 How to find $\hat{k}$

```
Require: $(z_1 \geq z_2 \geq, \ldots, \geq z_n), c > 0$
  initialize $S_1 \leftarrow 0, S_2 \leftarrow 0, \hat{k} \leftarrow 2$
  for $k = 2 : n$ do
    $S_1 \leftarrow S_1 + z_{k-1}$
    $S_2 \leftarrow S_2 + z^2_{k-1}$
    $G_k = \frac{1}{4z_k^2}(S_2 - 2z_k S_1 + (k-1)z_k^2) + \frac{1}{2z_k}(S_1 - (k-1)z_k)$
    if $G_k > c$ then
      $\hat{k} \leftarrow k - 1$
      return $\hat{k}$
    end if
  end for
  if $G_k \leq c$ then
    $\hat{k} \leftarrow n$
    return $\hat{k}$
  end if
```

In the second step, we already know that $|z|_{(\hat{k}+1)} < d \leq |z|_{(\hat{k})}$ which means $k = \hat{k}$ fixed now. Therefore solving a quadratic equation of d,

$$\frac{1}{4d^2}\sum_{i=1}^{\hat{k}}(|z|_{(i)} - d)^2 + \frac{1}{2d}\sum_{i=1}^{\hat{k}}(|z|_{(i)} - d) = c \tag{23}$$

determines the solution d. By the quadratic formula, the solution is

$$d = \left(\frac{\sum_{i=1}^{\hat{k}} |z|^2_{(i)}}{4c + \hat{k}}\right)^{\frac{1}{2}}, \tag{24}$$

knowing that $d > 0$. ■

Appendix B. Derivation of d-Ncut

We derive the directed-Normalized-cut (d-Ncut) criterion by mimicking the relationship between Normalized-cut and the graph Laplacian matrix of a undirected graph (Von Luxburg, 2007) .

Let us introduce a notation, $\mathbf{f}^t\mathcal{M}\mathbf{g} := \frac{1}{2}\sum_{i,j} W_{ij}(f_i - g_j)^2$, where W is an weight matrix. Given a community $C(S,T)$, S is the source part of a directional component,and T is its terminal part. We also denote $\bar{S}, \bar{T}$ the complement sets of S, T respectively.

d-Ncut criterion is obtained by assigning the normalized membership vectors to $\mathbf{f}, \mathbf{g}$ in $\mathbf{f}^t\mathcal{M}\mathbf{g}$. Let $\mathbf{f}, \mathbf{g} \in \mathbb{R}^n$ be piecewise constant vectors,

$$f_i = \begin{cases} \frac{1}{\sqrt{\mathrm{Vol}(S)}}, & i \in S \\ 0, & i \in \bar{S} \end{cases}$$
$$g_j = \begin{cases} \frac{1}{\sqrt{\mathrm{Vol}(T)}}, & j \in T \\ 0, & j \in \bar{T} \end{cases} \tag{25}$$

where $\mathrm{Vol}(S)$ is the sum of out-degrees of the nodes in S and $\mathrm{Vol}(T)$ is the sum of in-degrees of the nodes in T. Then $\mathbf{f}^t\mathcal{M}\mathbf{g}$ can be rearranged into

$$\begin{aligned} \mathbf{f}^t\mathcal{M}\mathbf{g} &= \sum_{i,j} W_{ij}(f_i - g_j)^2 \\ &= \sum_{i\in S, j\in\bar{T}} W_{ij}\left(\frac{1}{\sqrt{\mathrm{Vol}(S)}}\right)^2 + \sum_{i\in\bar{S}, j\in T} W_{ij}\left(\frac{1}{\sqrt{\mathrm{Vol}(T)}}\right)^2 \\ &\quad + \sum_{i\in S, j\in T} W_{ij}\left(\frac{1}{\sqrt{\mathrm{Vol}(S)}} - \frac{1}{\sqrt{\mathrm{Vol}(T)}}\right)^2 \\ &= \frac{\mathrm{Cut}(S,\bar{T})}{\mathrm{Vol}(S)} + \frac{\mathrm{Cut}(\bar{S},T)}{\mathrm{Vol}(T)} + \mathrm{Cut}(S,T)\left(\frac{1}{\sqrt{\mathrm{Vol}(S)}} - \frac{1}{\sqrt{\mathrm{Vol}(T)}}\right)^2. \end{aligned} \tag{26}$$

Here $\mathbf{f}^t\mathcal{M}\mathbf{g}$ measures the ratio of weights in external links and the balance of weights in S, T, which can be considered as a measure of the community structure of $C(S,T)$.

The d-Ncut criterion simultaneously considers a community and its complement, $C(S,T)$ and $C(\bar{S},\bar{T})$, and it is defined by

$$\text{d-Ncut}(S,T) = \mathbf{f}^t\mathcal{M}\mathbf{g} + \bar{\mathbf{f}}^t\mathcal{M}\bar{\mathbf{g}},$$

where $\bar{\mathbf{f}}, \bar{\mathbf{g}}$ are the membership vectors of $\bar{S}, \bar{T}$. In other words, small d-Ncut value of a community implies that the community not only has a strong community structure in itself but also separates other good community.

Furthermore, we want to discuss that regularized SVDs, (4) and (9), have connection to the minimization of d-Ncut criterion. The argument is similar to connection between Normalized-cut and spectral clustering in undirected graphs (Von Luxburg, 2007). Since the minimization of normalized cut is well-known NP-hard problem, a relaxation in the optimization problem is needed. The first term of d-Ncut criterion can be expressed as,

$$\begin{aligned}
\mathbf{f}^t \mathcal{M} \mathbf{g} &= \frac{1}{2} \sum_{i,j} W_{ij} (f_i - g_j)^2 \\
&= \frac{1}{2} \sum_{i,j} W_{ij} (f_i^2 + g_j^2 - 2 f_i g_j) \\
&= \frac{1}{2} \left(\sum_i d_{r,i} f_i^2 + \sum_j d_{c,j} g_j^2 - 2 \sum_{i,j} W_{ij} f_i g_j \right) \\
&= \frac{1}{2} \left(\mathbf{f}^t D_r \mathbf{f} + \mathbf{g}^t D_c \mathbf{g} - 2 \mathbf{f}^t W \mathbf{g} \right).
\end{aligned}$$

Notice that (25) implies $\mathbf{f}^t D_r \mathbf{f} = \mathbf{g}^t D_c \mathbf{g} = 1$. Thus, minimizing $\mathbf{f}^t \mathcal{M} \mathbf{g}$ is equivalent to

$$\min_{\mathbf{f},\mathbf{g}} -\mathbf{f}^t W \mathbf{g}, \quad \text{subject to } \mathbf{f}^t D_r \mathbf{f} = \mathbf{g}^t D_c \mathbf{g} = 1, \tag{27}$$

if the condition of piecewise constant vectors on $\mathbf{f}, \mathbf{g}$ is dropped from (25). Further simplification of (27) brings

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^t \mathbf{Q} \mathbf{v}, \quad \text{subject to } \mathbf{u}^t \mathbf{u} = \mathbf{v}^t \mathbf{v} = 1. \tag{28}$$

where $\mathbf{u} = D_r^{\frac{1}{2}} \mathbf{f}$, $\mathbf{v} = D_c^{\frac{1}{2}} \mathbf{g}$.

Even though the relaxation converts the NP-hard problem to linear algebra problem that is easier to solve, it is unknown that how good the approximated solution is. Typically, the approximated solution provides global division of a graph. Our initial motivation was to give sparsity-inducing penalization on $\mathbf{u}, \mathbf{v}$, which resembles the sparse membership vectors, in order to recover small communities directly.

Appendix C. Tables of Section 6.1.2

Here we provide the results of community detection algorithms (EN-harvesting, DI-SIM, Infomap) on Cora citation network. The result of L_0-harvesting algorithm is displayed in Table 4. These tables show the number of papers in manually assigned categories for each community.

	AI	DSAT	DB	EC	HA	HCI	IR	Net	OS	Prog	Uncategorized
1	6042	320	147	115	144	129	235	45	136	255	548
2	0	0	0	0	0	0	0	0	6	0	0
3	0	0	0	0	0	0	0	0	8	0	0
4	446	385	20	47	152	31	4	3	106	655	225
5	1800	1256	805	305	389	1139	190	637	1734	2039	1205
6	6	5	6	130	5	60	0	509	133	13	95
7	41	315	3	256	8	0	3	1	24	2	50
8	18	25	11	250	7	27	8	51	182	28	47
9	92	2	18	1	1	0	0	0	0	73	20
10	0	0	0	0	0	0	0	2	12	0	2
11	87	67	8	29	8	1	2	3	3	3	19
12	7	3	0	1	0	0	0	25	1	0	4
13	14	18	0	0	0	1	0	25	6	1	2
14	2	0	0	0	0	0	0	0	0	0	0
15	78	0	0	0	0	0	2	0	0	0	1
16	0	0	1	0	2	0	0	0	0	0	0
17	0	0	33	0	2	0	1	0	0	0	2
18	57	1	2	0	0	0	0	0	0	3	10
19	55	3	6	0	0	0	0	0	0	2	23
20	0	0	0	0	0	0	0	13	0	0	5

Table 5: Number of papers in the first twenty approximated directional components of EN-harvesting for each category.

	AI	DSAT	DB	EC	HA	HCI	IR	Net	OS	Prog	Uncategorized
1	687	1084	723	575	571	416	71	750	1176	1933	890
2	28	4	0	0	0	0	0	0	1	0	1
3	4	0	0	0	0	0	0	0	0	0	5
4	2650	14	18	21	6	47	95	1	5	14	144
5	120	165	78	85	177	173	13	489	1023	1075	332
6	100	42	5	14	13	17	5	10	18	23	20
7	2509	1374	269	316	373	506	126	288	305	779	519
8	13	8	0	18	0	3	0	0	0	0	0
9	4658	406	167	149	64	485	272	20	50	148	430
10	15	7	1	3	3	4	0	3	2	0	4

Table 6: Number of papers in the source partition DI-SIM for each category.

	AI	DSAT	DB	EC	HA	HCI	IR	Net	OS	Prog	Uncategorized
1	118	1	0	1	0	1	0	0	0	0	10
2	49	87	1	24	1	0	0	0	1	1	6
3	0	0	0	6	2	2	0	1	36	13	2
4	7	0	5	1	27	5	1	2	56	16	18
5	1	0	7	8	30	2	0	1	137	11	15
6	0	0	0	1	0	3	0	8	1	0	0
7	222	0	0	0	0	0	0	3	0	0	9
8	1	1	2	0	0	1	0	2	38	35	15
9	0	0	0	0	0	0	0	0	1	66	2
10	133	0	0	0	1	0	2	0	0	1	9
11	6	80	0	0	0	0	0	1	16	4	6
12	0	0	0	17	0	43	0	183	8	0	19
13	1	16	0	0	7	0	0	0	22	131	8
14	166	0	0	0	0	0	0	0	0	0	11
15	1	17	0	0	2	0	0	0	31	38	9
16	14	0	0	0	0	0	0	0	0	0	0
17	4	1	2	1	0	5	0	0	1	116	3
18	0	0	0	0	0	6	0	74	17	3	8
19	1	1	0	0	1	0	0	0	1	103	15
20	98	3	0	0	0	0	0	0	0	0	6

Table 7: Number of papers in the randomly selected 20 communities detected by Infomap for each category.

References

R. Andersen and K.J. Lang. Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, pages 223–232. ACM, 2006.

R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

A. Arenas, J. Duch, A. Fernández, and S. Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9(6):176, 2007.

A. Capocci, V.D.P. Servedio, G. Caldarelli, and F. Colaiori. Detecting communities in large networks. *Physica A: Statistical Mechanics and its Applications*, 352(2):669–676, 2005.

M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analy Data Mining*, 2011.

L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

A. d'Aspremont, F. Bach, and L.E. Ghaoui. Optimal solutions for sparse principal component analysis. *The Journal of Machine Learning Research*, 9:1269–1294, 2008.

I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

I.S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11): 1944–1957, 2007.

S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

R. Guimerà, M. Sales-Pardo, and L.A.N. Amaral. Module identification in bipartite and directed networks. *Physical Review E*, 76(3):036102, 2007.

P.W. Holland, K.B. Laskey, and S. Leinhardt. Stochastic blockmodels: first steps. *Social networks*, 5(2):109–137, 1983.

R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, 1990. ISBN 9780521386326.

R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis.* Topics in Matrix Analysis. Cambridge University Press, 1994. ISBN 9780521467131.

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5):056117, 2009a.

A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80 (1):016118, 2009b.

A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(046110), 2008.

A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

M. Lee, H. Shen, J.Z. Huang, and JS Marron. Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095, 2010.

E.A. Leicht and M.E.J. Newman. Community structure in directed networks. *Physical Review Letters*, 100(11):118703, 2008.

J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceeding of the 17th international conference on World Wide Web*, pages 695–704. ACM, 2008.

MATLAB. *version 8.0.0.783 (R2012b), glnxa64*. The MathWorks Inc., Natick, Massachusetts, 2012.

M.E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

M.E.J. Newman and E.A. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564–9569, 2007.

S. Parthasarathy, Y. Ruan, and V. Satuluri. Community discovery in social networks: Applications, methods and emerging trends. *Social Network Data Analytics*, pages 79–113, 2011.

K. Rohe and B. Yu. Co-clustering for directed graphs; the stochastic co-blockmodel and a spectral algorithm. *arXiv preprint arXiv:1204.2296*, 2012.

M. Rosvall, D. Axelsson, and C.T. Bergstrom. The map equation. *The European Physical Journal-Special Topics*, 178(1):13–23, 2009.

V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 737–746. ACM, 2009.

V. Satuluri and S. Parthasarathy. Symmetrizations for clustering directed graphs. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 343–354. ACM, 2011.

H. Shen and J.Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of multivariate analysis*, 99(6):1015–1034, 2008.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

D. M Witten, R Tibshirani, and T Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, Jul 2009.

D. Yang, Z. Ma, and A. Buja. A sparse svd method for high-dimensional data. *arXiv preprint arXiv:1112.2433*, 2011.

Y. Zhao, E. Levina, and J. Zhu. Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 108(18):7321–7326, 2011.

D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. *Advances in neural information processing systems 17.*, 2005.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.