Locally-biased Spectral Approximation for Community Detection

Pan Shi\textsuperscript{a}, Kun He\textsuperscript{a}\textsuperscript{*}, David Bindel\textsuperscript{b}, John E. Hopcroft\textsuperscript{b}

\textsuperscript{a}School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
\textsuperscript{b}Department of Computer Science, Cornell University, Ithaca, NY 14853, USA

Abstract

We propose a Locally-Biased Spectral Approximation (LBSA) approach for identifying all latent members of a local community from very few seed members. To reduce the computation complexity, we first apply a fast random walk, personalized PageRank and heat kernel diffusion to sample a comparatively small subgraph covering almost all potential community members around the seeds. Then starting from a normalized indicator vector of the seeds and by a few steps of either Lanczos iteration or power iteration on the sampled subgraph, a local eigenvector is gained for approximating the eigenvector of the transition matrix with the largest eigenvalue. Elements of this local eigenvector is a relaxed indicator for the affiliation probability of the corresponding nodes to the target community. We conduct extensive experiments on real-world datasets in various domains as well as synthetic datasets. Results show that the proposed method outperforms state-of-the-art local community detection algorithms. To the best of our knowledge, this is the first work to adapt the Lanczos method for local community detection, which is natural and potentially effective. Also, we did the first attempt of using heat kernel as a sampling method instead of detecting communities directly, which is proved empirically to be very efficient and effective.

Keywords:
Community detection, Local spectral approximation, Power iteration, Lanczos method

1. Introduction

Community detection is an important task in network analysis, which aims to find a set of nodes in a network that are internally cohesive but comparatively separated from the remainder of the network. In social networks, community detection is a classical and challenging problem which is very useful for analyzing the topology structure and extracting information from the network, and numerous algorithms and techniques have been proposed \cite{1, 2, 3, 4, 5, 6}.

Over the past decades, most of the researchers have focused on uncovering the global community structure, including modularity maximization model \cite{7, 8, 9, 10}, spectral method \cite{11, 12}, stochastic model \cite{13, 14} and so on. With the rapid growth of the network scale, global community detection becomes very costly or even impossible for very large networks. The big data drives researchers to shift their attention from the global structure to the local structure \cite{15, 16, 17, 18, 19}. How to adapt the effective methods initially designed for the global community detection in order to uncover the local community structure is a natural and important approach for the accurate membership identification from a few exemplary members. Several probability diffusion methods, PageRank \cite{15}, heat kernel \cite{16} and spectral subspace approximation \cite{20, 21, 22, 23, 24} are three main techniques for local community detection.

The power iteration \cite{25} and Lanczos iteration \cite{26} are two classic methods proposed for calculating the eigenvalues, aka the spectra of a matrix. As a simple eigenvalue calculation method, power method has been widely used for spectral clustering \cite{27, 28}. On the other hand, though there exists some work using the Lanczos method for the spectral bisection \cite{29}, unlike other spectra calculation methods, the Lanczos method is seldom used for community detection and to the best of our knowledge, it has never been used for the local community detection. In this paper, we propose a novel approach called the Locally-Biased Spectral Approximation (LBSA) for local community detection. Specifically, we execute a few steps of power iteration or Lanczos iteration to attain a local eigenvector that approximates the eigenvector of the transition matrix with the largest eigenvalue. Elements of this local eigenvector is a relaxed indicator for the affiliation probability of the corresponding nodes to the target community. As compared with other spectral approximation methods, the power method and Lanczos iterative method are efficient for computing the top eigenpairs of large sparse matrices and they are space efficient, which are very helpful for large and usually sparse social networks.

This paper is a significant extension of our ECML-PKDD conference paper \cite{30}. The conference paper uses heat kernel diffusion as a sampling method, and does Lanczos iteration for local community detection. We did plenty extensions on methods, analysis and experiments so as to have an integrate method and extensive analysis. Our new contributions include: (1) Besides heat kernel, we also apply a fast random walk and personalized PageRank diffusion to sample a localized subgraph to largely reduce the subsequent calculation. (2) Compared to Lanczos method, we also adopt power method for...
locally-biased spectral approximation, and provide some sound convergence analysis. (3) Based on more baseline algorithms and datasets, we conduct extensive experiments on real-world networks as well as synthetic datasets to demonstrate the superiority of the proposed method.

2. Related work

2.1. Local community detection

Techniques for local community detection can be classified into three categories, namely the PageRank, heat kernel and local spectral methods. Other techniques like finding minimum cut [31, 32, 33] can also be used for local community detection.

PageRank. The PageRank method is widely used for local community detection. Spielman and Teng [34] use degree-normalized, personalized PageRank (DN PageRank) with respect to the initial seeds and do truncation on small probability values. DN PageRank is adopted by several competitive PageRank based clustering algorithms [35, 36], including the popular PageRank Nibble method [37]. Kloumann and Kleinberg [15] propose a deterministic method that uses coordinate relaxation on an implicit linear system to estimate the heat kernel diffusion, and a heat value of each node represents the likelihood of affiliation.

Heat kernel. The heat kernel method involves the Taylor series expansion of the exponential of the transition matrix. Chung [38, 39] provides a theoretical analysis and a local graph partitioning algorithm based on heat kernel diffusion. Chung and Simpson [40] propose a randomized Monte Carlo method to estimate the diffusion speed, and Kloster and Gleich [16] propose a deterministic method that uses coordinate relaxation on an implicit linear system to estimate the heat kernel diffusion, and the heat value of each node represents the likelihood of affiliation.

Local spectral. A third branch is to adapt the classic spectral method to locate the target community. Mahoney et al. [41] introduce a locally-biased analogue of the second eigenvector, the Fiedler vector associated with the algebraic connectivity, to extract local properties of data graphs, and apply the method for a semi-supervised image segmentation and a local community extraction by finding a sparse-cut around the seeds in small social networks. He et al. [20, 22, 23] and Li et al. [21, 24] extract the local community by seeking a sparse vector from the local spectral subspaces using $\ell_1$ norm optimization.

2.2. Spectral calculation

2.2.1. Power method

One of the oldest techniques for solving eigenvalue problems is the so-called power method [25]. The power method is an iterative technique used to determine the dominant eigenvalue of a matrix, that is, the eigenvalue with the largest magnitude. By modifying the method slightly, it can also be used to determine other eigenvalues [42]. One useful feature of the power method is that it produces not only an eigenvalue, but also an associated eigenvector. In fact, the power method is often applied to find an eigenvector for an eigenvalue that is determined by some other means.

As a simple and classic eigenvalue calculation method, the power method has been applied in many field. For example in the application of graph clustering and feature selection, Lin and Cohen [27] propose a simple graph clustering algorithm which finds a very low-dimensional embedding of a dataset using truncated power iteration; Thang et al. [43] illustrate a novel clustering method based on the deflation power iteration technique to compute multiple orthogonal pseudo-eigenvectors; Huang et al. [44] propose diverse power iteration embedding method for feature selection.

2.2.2. Lanczos method

Many real world problems can be modeled as sparse graphs and be represented as matrices, and the eigenvalue calculation of the matrices is usually a crucial step for the problem solving. All the eigenpairs can be calculated by eigenvalue decomposition [42], SVD [45], or QR factorization [46]. However, these methods are intractable for large matrices due to the high complexity and memory consumption. As the Lanczos method can significantly reduce the time and space complexity, it is usually applied to large sparse matrices [47].

As a classic eigenvalue calculation method, the original Lanczos method [26] cannot hold the orthogonality of the calculated Krylov subspace and it is not widely used in practice. Paige [47] computes the eigenpairs for very large sparse matrices by an improved Lanczos method, as only a few iterations are typically required to get a good approximation on the extremal eigenvalues. Thereafter, the Lanczos method becomes very attractive for large sparse matrix approximation. For example in the application of graph partitioning and image reconstruction, Barnes [29] illustrates that Lanczos method is an efficient implementation of the spectral bisection method; Wu et al. [48] propose an incremental bilinear Lanczos algorithm for high dimensionality reduction and image reconstruction; Bentbib et al. [49] illustrate that efficient image restoration can be achieved by Tikhonov regularization based on the global Lanczos method.

To the best of our knowledge, there exist no Lanczos based algorithms for local community detection in the literature.

3. Preliminaries

3.1. Problem formulation

In a network, we want to identify a target community supervised by a few observed members. We call the problem the local community detection. It can be formalized as follows. We are given a connected, unweighted and undirected graph $G = (V,E)$ with $n$ nodes and $m$ edges. Let $A \in \{0, 1\}^{n \times n}$ be the symmetric associated adjacency matrix, and $D$ the diagonal matrix of node degrees. Let $S$ be the seed set of a few exemplary members in the target ground-truth community, denoted by a set of nodes $T (S \subset T, |T| \ll |V|)$. Let $s \in \{0, 1\}^n$ be a binary indicator vector representing the exemplary members in $S$. We are asked to identify the remaining latent members in the ground-truth community $T$ as accurate as possible.

For accuracy evaluation, we adopt $F_1$ score and Jaccard index to quantify the similarity between the detected local community $C$ and the target ground-truth community $T$. The $F_1$
score for each pair of \((C, T)\) is defined by:
\[
F_1(C, T) = \frac{2 \cdot P(C, T) \cdot R(C, T)}{P(C, T) + R(C, T)}.
\]
where the precision \(P\) and recall \(R\) are defined as:
\[
P(C, T) = \frac{|C \cap T|}{|C|}, R(C, T) = \frac{|C \cap T|}{|T|}.
\]
The Jaccard index for each pair of \((C, T)\) is defined by:
\[
J(C, T) = \frac{|C \cap T|}{|C \cup T|}.
\]

3.2. Notations and descriptions

Table 1 summarizes a list of different notations we will use throughout the paper. In general, we use italic lowercase letters, e.g., \(n\), to denote scalars; lowercase boldface characters, e.g., \(y\), to denote vectors; uppercase boldface characters, e.g., \(A\), to denote matrices; and italic capital letters, e.g., \(C\), to denote sets.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G)</td>
<td>Graph</td>
</tr>
<tr>
<td>(A)</td>
<td>Adjacency matrix of graph (G)</td>
</tr>
<tr>
<td>(D)</td>
<td>Diagonal degree matrix of graph (G)</td>
</tr>
<tr>
<td>(S)</td>
<td>Seed set</td>
</tr>
<tr>
<td>(T)</td>
<td>Ground-truth community</td>
</tr>
<tr>
<td>(C)</td>
<td>Detected community</td>
</tr>
<tr>
<td>(G_s)</td>
<td>Subgraph extracted from the neighborhood surrounding the seed set (S)</td>
</tr>
<tr>
<td>(A_s)</td>
<td>Adjacency matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(D_s)</td>
<td>Diagonal degree matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(L_{sym})</td>
<td>Normalized Laplacian matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(N_{rw})</td>
<td>Transition matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(\Phi(\hat{V}))</td>
<td>Conductance of a node set (\hat{V})</td>
</tr>
<tr>
<td>(\lambda_1)</td>
<td>The first largest eigenvalue of matrix (N_{rw})</td>
</tr>
<tr>
<td>(\lambda_2)</td>
<td>The smallest eigenvalue of matrix (L_{sym})</td>
</tr>
<tr>
<td>(y)</td>
<td>Indicator vector, where larger value indicates higher probability being in the same community</td>
</tr>
</tbody>
</table>

4. Locally-biased spectral approximation methods

In this section, there are three key steps in the proposed algorithms: local sampling, locally-biased spectral approximation and community boundary truncation.

4.1. Local sampling

In this subsection, we adopt fast random walk, personalized PageRank and heat kernel diffusion to do three types of the sampling, and set parameters such that the resulting subgraph is large enough to cover almost all members in the target community.

4.1.1. Local random walk sampling

We do \(k_0\) steps of standard random walks, and define the \(k_0\)-walk diffusion vector as
\[
p_{k_0} = (AD^{-1})^{k_0}p_0,
\]
where \(p_0 = s/|S|\) is the initial probability distribution on the source seeds.

For a large graph, it is impractical to compute the matrix \(AD^{-1}\) directly, so we implement a fast method to update the probability vector \(p_{k_0}\). Details of the sampling are as shown in Algorithm 1. For notations in Algorithm 1, at iteration \(t\), \(p_s(s_i)\) denotes the probability of the specified node \(s_i\), and \(p_s\) without parameter denotes the probability vector for nodes 1 to \(n_s\).

Algorithm 1 The random walk sampling

**Input:** graph \(G = (V, E)\), seed set \(S \subset V\), steps of standard random walks \(k_0\), upper bound of the subgraph size \(n_1\)

**Output:** Sampled subgraph \(G_s = (V_s, E_s)\)

1: initialize \(t = 0\), \(p_0 = s/|S|\)
2: \(V_0 \leftarrow S\)
3: while \((t < k_0)\) do
4: \(p_{t+1} = p_t\)
5: for each \(s_i \in V_s\) do
6: \(U_i \leftarrow\) neighbors of \(s_i\)
7: \(p_{t+1}(s_i) = p_t(s_i) + \frac{p_t(s_i)}{|U_i|}\)
8: for each \(v_i \in U_i\) do
9: \(p_{t+1}(v_i) = p_{t+1}(v_i) + p_t(s_i)/|U_i|\)
10: end for
11: \(V_s = V_s \cup U_i\)
12: end for
13: \(t = t + 1\)
14: end while
15: if \(|V_s| > n_1\) then
16: \(V_s \leftarrow n_1\) nodes inside \(V_s\) with higher probability in \(p_s\)
17: end if
18: \(G_s = (V_s, E_s)\) is the induced subgraph from \(V_s\)

4.1.2. Local personalized PageRank sampling

For a fixed \(\alpha \in (0, 1)\), the personalized PageRank vector can be defined as
\[
p = (1 - \alpha) \sum_{k=0}^{\infty} \alpha^k (AD^{-1})^k p_0.
\]
where \(p_0 = s/|S|\) is the initial probability distribution on the source seeds.

In practice, we usually seek a vector \(q\) to approximate \(p\):
\[
||D^{-1}p - D^{-1}q||_\infty < \epsilon_1.
\]

Andersen et al. [37] propose a \textit{pprpush} algorithm to guarantee (3), and use the probability values in \(q\) to identify memberships in the local community. We just adapt the advantage of \textit{pprpush}'s fast speed to do the personalized PageRank sampling, shown in Algorithm 2.
indicate the sum of the first l terms.

In practice, we usually seek a vector x to approximate h:

$$\|D^{-1}h - D^{-1}x\|_\infty < \epsilon_2.$$ (6)

Premultiplying $e^t$ on both sides, we have

$$\|D^{-1}e^t h - D^{-1}e^t x\|_\infty < e^t \epsilon_2.$$ (7)

If for an integer l,

$$\|D^{-1}e^t h - D^{-1}e^t h_l\|_\infty < e^t \epsilon_2 / 2,$$ (8)

and $z = e^t x \approx e^t h_l$ satisfies

$$\|D^{-1}e^t h_l - D^{-1}z\|_\infty < e^t \epsilon_2 / 2,$$ (9)

then by the triangle inequality, (7) holds, and then (6) holds.

Algorithm 3 The heat kernel sampling

Input: Graph $G = (V,E)$, seed set $S \subset V$, heat kernel diffusion parameters $\tau$ and $\epsilon_2$, upper bound of the subgraph size $n_3$

Output: Sampled subgraph $G_s = (V_s,E_s)$

1: Start from $S$, calculate the heat value vector $x$ to approximate the heat kernel diffusion vector $h$
2: Sort elements in $x$ in decreasing order to get a vector $\tilde{x}$
3: $V_s \leftarrow$ nodes corresponding to all the nonzero elements in $\tilde{x}$
4: if $|V_s| > n_3$ then
5: $V_s \leftarrow n_3$ nodes inside $V_s$ with higher heat value in $\tilde{x}$
6: end if
7: $G_s = (V_s,E_s)$ is the induced subgraph from $V_s$

Kloster and Gleich [16] propose a hk-relax algorithm to guarantee (8) by letting $t$ be no greater than $2t \log(\frac{1}{\epsilon_2})$ and computing a vector $z$ that satisfies (9), then use the heat values in $x$ to identify memberships in the local community. We just adapt the advantage of hk-relax’s fast speed to do the heat kernel sampling, shown in Algorithm 3.

Denote the sampled subgraph as $G_s = (V_s,E_s)$ with $n_s$ nodes and $m_s$ edges in the following discussion. We then extract the local community from this comparatively small subgraph instead of the original large network. This pre-processing procedure runs in milliseconds in large networks with millions of nodes, and significantly reduces the computation cost for the follow-up community detection.

4.2. Locally-biased spectral approximation

In this subsection, we first provide the necessary theoretical base that finding a low-conductance community corresponds to finding the eigenvector of the transition matrix with the largest eigenvalue. Then we briefly introduce a classical power iteration and a variant of the Lanczos iteration to approximate this eigenvector, and get a “local” eigenvector indicating the implicit topology structure of the network around the seeds, and provide some convergence analysis on the power iteration and Lanczos iteration.

4.2.1. Theoretical base

Let $L = D_s - A_s$ be the unnormalized Laplacian matrix of $G_s$, where $A_s$ and $D_s$ denote the adjacency matrix and the diagonal degree matrix of $G_s$. We define two normalized graph Laplacian matrices:

$$L_{rw} = D_s^{-1}L = I - N_{rw},$$

$$L_{sym} = D_s^{-\frac{1}{2}}LD_s^{-\frac{1}{2}} = I - N_{sym},$$

where $I$ is the identity matrix, $N_{rw} = D_s^{-1}A_s$ is the transition matrix, and $N_{sym} = D_s^{-\frac{1}{2}}A_sD_s^{-\frac{1}{2}}$ is the normalized adjacency matrix.

For a community $C$, the conductance [50] of $C$ is defined as

$$\Phi(C) = \frac{\text{cut}(C, \bar{C})}{\min(\text{vol}(C), \text{vol}(\bar{C}))},$$

where $\bar{C}$ consists of all nodes outside $C$, cut$(C, \bar{C})$ denotes the number of edges between, and vol(·) calculates the “edge volume”, i.e. for the subset of nodes, we count their total node degrees in graph $G_s$. Low conductance gives priority to a community with dense internal links and sparse external links.

Let $y \in \{0,1\}^{n \times 1}$ be a binary indicator vector representing a small community $C$ in the sampled graph $G_s$. Here for “small community”, we mean vol$(C) \leq \frac{1}{4} \text{vol}(V_s)$. As $y^T D_s y$ equals the total node degrees of $C$, and $y^T D_s y$ equals two times the number of internal edges of $C$, the conductance $\Phi(C)$ could be written as a generalized Rayleigh quotient:

$$\Phi(C) = \frac{y^T Ly}{y^T D_s y} = \frac{(D_s^\frac{1}{2} y)^T L_{sym} (D_s^\frac{1}{2} y)}{(D_s^\frac{1}{2} y)^T (D_s^\frac{1}{2} y)}.$$ (13)
Theorem 1. (Cheeger Inequality) Let $\lambda_2$ be the second smallest eigenvalue of $L_{\text{sym}}$ for a graph $G$, then $\phi(G) \geq \frac{\lambda_2}{2}$. where $\phi(G) = \min_{C \subset V}, \Phi(V)$.

The proof refers to [51], and we attach the details in Appendix A. According to this theorem and the definition of $\Phi(C)$, we have $\frac{\lambda_2}{2} \leq \Phi(C) \leq 1$.

According to the Rayleigh–Ritz theorem [52], if we want to minimize the conductance $\Phi(C)$ by relaxing the indicator vector $y$ to take arbitrary real values, then the scaled relaxed indicator vector $D^\frac{1}{2}_s y$ should be the eigenvector of $L_{\text{sym}}$ with the smallest eigenvalue 0, which is $D^\frac{1}{2}_s e_s$ with $e_s$ the vector of length $n_s$ with ones as the entries.

We know that:

$$L_{\text{rw}} v = \lambda v \iff L_{\text{sym}} (D^\frac{1}{2}_s v) = \lambda (D^\frac{1}{2}_s v), \quad (14)$$

where $v$ is a nonzero vector. The relaxed indicator vector $y$ should be the eigenvector of $L_{\text{rw}}$ with the smallest eigenvalue. As $L_{\text{rw}} = I - N_{\text{rw}}$, the eigenvalue decomposition of the Laplacian matrix is also closely related to the expansion of rapid mixing of random walks. As

$$L_{\text{rw}} v = \lambda v \iff N_{\text{rw}} v = (1 - \lambda) v, \quad (15)$$

it follows that $L_{\text{rw}}$ and $N_{\text{rw}}$ share the same set of eigenvectors and the corresponding eigenvalue of $N_{\text{rw}}$ is $1 - \lambda$ where $\lambda$ is the eigenvalue of $L_{\text{rw}}$. Equivalently, the relaxed indicator vector $y$ should be the eigenvector of $N_{\text{rw}}$ with the largest eigenvalue.

The largest eigenvalue of $N_{\text{rw}}$ is 1 and the corresponding eigenvector is $e_s$ [12], so the relaxed indicator vector $y = e_s$, corresponding to the whole graph with zero conductance by Eq. (13). This relaxed indicator vector $y$ contains global information while the real solution of the indicator vector $y$ reveals local property for a small community whose total degree is no greater than half of the total degree of the whole graph. As the power iteration and the Lanczos method are efficient for computing the top eigen-pairs of large sparse matrices and they are space efficient, we propose a power iteration and a variant of Lanczos method on $N_{\text{rw}}$ to get a “local” eigenvector indicating the latent local structure around the seeds.

4.2. Power iteration spectral approximation

Power iteration spectral approximation. Instead of using the eigenvalue decomposition to get the global eigenvector, we consider power iteration to approximate the eigenvector of $N_{\text{rw}}$ with the largest eigenvalue. A few steps of the power iteration lead to the local approximation of this eigenvector.

By $k$ steps of power iteration, we could get an iteration vector

$$x_k = (N_{\text{rw}})^k x_0, \quad (16)$$

where $x_0$ is the normalized indicator vector for the seed set. When $k \to \infty$, Theorem 2 states that the normalized iteration vector $x_k$ converges to the eigenvector of $N_{\text{rw}}$ with the largest eigenvalue and it contains global information of $G$. Our interest, though, is not in the limiting case when $k$ is large, but for a very small number of iteration steps to reveal the local property around the seeds.

The Power Iteration Spectral Approximation (PISA) procedure on the sampled graph is summarized in Algorithm 4. The slowest step is Step 6 for calculating $\lambda^{(k)}$. It requires $O(k/n^2_s)$ time to implement the power iteration, where $k_i$ is the steps of power iteration and $n_s$ is the number of nodes in graph $G$.

**Algorithm 4** Power Iteration Spectral Approximation

**Input:** $G_s = (V_s, E_s)$, maximum iteration steps $k_i$, initial vector $x_0$

**Output:** $y = y_k$

1: Initialize $k = 0$
2: while $(k < k_i)$ do
3: \quad $k = k + 1$
4: \quad $x_k = N_{\text{rw}} x_{k-1}$
5: \quad $y_k = x_k/\|x_k\|_2$
6: \quad $\lambda^{(k)} = y_k^T N_{\text{rw}} y_k$
7: end while

**Convergence analysis.** In the following discussion, we provide an analysis on the convergence of the power iteration, i.e. when $k \to \infty$, the approximation vector $y_k$ converges to the eigenvector of $N_{\text{rw}}$ with the largest eigenvalue.

**Lemma 1.** Let $G_s = (V_s, E_s)$ be a connected and non-bipartite graph with $n_s$ nodes and $m_s$ edges, $N_{\text{rw}}$ the transition matrix of $G_s$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{n_s}$, and the corresponding normalized eigenvectors are $v_1, v_2, \ldots, v_{n_s}$. Then $v_1, v_2, \ldots, v_{n_s}$ are linearly independent, and

$$1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{n_s} > -1, \quad v_1 = \frac{e_s}{\|e_s\|_2},$$

where $e_s$ is a vector of length $n_s$ with ones as the entries.

**Proof.** By Eq. (15), we know that $L_{\text{rw}}$ and $N_{\text{rw}}$ share the same eigenvectors $v_1, v_2, \ldots, v_{n_s}$ and the corresponding eigenvalues of $L_{\text{rw}}$ are $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n_s}$ where $\lambda_i = 1 - (1 - 1)^2 (i \leq n_s)$.

According to Proposition 3 of [12], $L_{\text{sym}}$ and $L_{\text{rw}}$ share the $n_s$ non-negative eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n_s}$, and the corresponding eigenvectors of $L_{\text{sym}}$ are $D^\frac{1}{2}_s v_1, D^\frac{1}{2}_s v_2, \ldots, D^\frac{1}{2}_s v_{n_s}$.

From Theorem 8.1.1 of [42], there exists an orthogonal matrix $V = [v_1, v_2, \ldots, v_{n_s}]$ such that

$$V^T L_{\text{sym}} V = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{n_s}).$$

It shows that $v_1, v_2, \ldots, v_{n_s}$ are linearly independent eigenvectors of $L_{\text{sym}}$, so $D^\frac{1}{2}_s v_1, D^\frac{1}{2}_s v_2, \ldots, D^\frac{1}{2}_s v_{n_s}$ are linearly independent. As $G_s$ is a connected graph, $D^\frac{1}{2}_s$ is invertible, and it is obvious that $v_1, v_2, \ldots, v_{n_s}$ are linearly independent.

Additionally, as $L_{\text{rw}} x \|_{\| \cdot \|_2} = 0$, we have $1 - \lambda_1 = \lambda_1 = 0$ and $v_1 = \frac{e_s}{\|e_s\|_2}$, so $\lambda_1 = 1$.

As $G_s$ is a connected and non-bipartite graph, by Lemma 1.7 of [51], we have $1 - \lambda_2 = \lambda_2 > 0$ and $1 - \lambda_{n_s} = \lambda_{n_s} < 2$. Therefore, $\lambda_2 < 1$ and $\lambda_{n_s} > -1$. }

5
**Theorem 2.** Let $G_k = (V_k, E_k)$ be a connected and non-bipartite graph with $n_k$ nodes and $m_k$ edges, when $k \to \infty$, the vector $y_k$ converges to $v_1$ and $k^{(i)}$ converges to $\lambda_1$.

**Proof.** By Lemma 1, $v_1, v_2, \ldots, v_n$ are linearly independent normalized eigenvectors of $N_{rw}$, so there exist $a_1, a_2, \ldots, a_n$ such that $x_0 = \sum_{i=1}^{n} a_i v_i$.

As $v_1 = \frac{x_0}{\|x_0\|}$ and $x_0$ is the normalized indicator vector for the seed set, $v_1$ and $x_0$ are not orthogonal. It shows that $a_1$ is the nonzero projection portion of $x_0$ on the eigenvector $v_1$. Then by Eq. (16) and $(N_{rw})v_i = \lambda_i v_i$ $(1 \leq i \leq n)$, we have

$$x_k = (N_{rw})^k x_0 = \sum_{i=1}^{n} a_i k^i v_i.$$  

Since $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{n_1} > -1$, for all $i = 2, 3, \ldots, n_1$, we have

$$\lim_{k \to \infty} k^i = 0,$$

and

$$\lim_{k \to \infty} a_i k^i v_i = a_1 v_1.$$  

As $a_1 \neq 0$, we have

$$\lim_{k \to \infty} y_k = \lim_{k \to \infty} \frac{x_k}{\|x_k\|} = \frac{a_1 v_1}{a_1} = v_1,$$

and

$$\lim_{k \to \infty} k^{(i)} = \lim_{k \to \infty} y_k^T N_{rw} y_k = v_1^T N_{rw} v_1 = \lambda_1 v_1^T v_1 = \lambda_1.$$  

As $v_1, v_2, \ldots, v_{n_1}$ are linearly independent normalized eigenvectors of $N_{rw}$, there exist orthonormal vectors $u_1, u_2, \ldots, u_{n_1}$ such that $N_{rw} u_i = \lambda_i u_i$, $\|u_i\|_2 = 1$, and $u_i^T u_j = 0$ $(1 \leq i \neq j \leq n_1)$. Then we could find $b_1, b_2, \ldots, b_{n_1}$ such that $x_0 = \sum_{i=1}^{n_1} b_i u_i$.

By Eq. (16), we have

$$x_k = (N_{rw})^k x_0 = \sum_{i=1}^{n_1} b_i k^i u_i.$$  

Let $y = y_k$ and $\lambda = \lambda^{(i)}$, the approximated residual value can be calculated as

$$\|r_k\|_2 = \|N_{rw} y - \lambda y\|_2 = \|N_{rw} y_k - \lambda^{(i)} y_k\|_2.$$  

By Algorithm 4, Eq. (18) can be modified as

$$\|r_k\|_2 = \|x_{k+1}\|_2 - \|x_k\|_2 - \|y_k^T x_{k+1} + (x_k^T x_k) \|_2 = \frac{1}{\|x_k\|_2} \|x_k^T x_k\|.$$  

Let $x = x_k^T x_{k+1} - x_k^T x_{k+1} x_k$, by Eq. (17),

$$\|x\|_2 = \sqrt{x^T x} = \sqrt{x_k^T x_k - x_k^T x_{k+1} x_k} = \sqrt{x_k^T x_k - x_k^T x_{k+1} x_k},$$

By Eq. (20), we have

$$\|r_k\|_2 = \sqrt{\left(\sum_{i=1}^{n_1} b_i^2 \lambda_i^{2k} \right) \left(\sum_{i=1}^{n_1} b_i^2 \lambda_i^{2k+2} \right) - \left(\sum_{i=1}^{n_1} b_i^2 \lambda_i^{2k+1} \right)^2}.$$  

By Lemma 1, $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{n_1} > -1$, we have

$$\lambda_1^{2k+2} = \lambda_1^{2k} = 1$$

$$\lambda_i^{2k+2} < \lambda_i^{2k} < 1 \quad (1 \leq i \leq n_1).$$  

When $k \to \infty$, Theorem 2 states that the residual value $\|r_k\|_2 = 0$. And we just need to use a small power iteration steps to find the “local” eigenvector.

### 4.2.3. Lanczos iteration spectral approxiamtion

**Lanczos process.** Based on a theoretical guarantee [53], there exists an orthogonal matrix $Q$ and a tridiagonal matrix $T$ such that

$$Q^T (D_s^{-\frac{1}{2}} A_s D_s^{-\frac{1}{2}}) Q = T.$$  

Designate the columns of $Q$ by

$$Q = [q_1 \cdots q_{n_1}].$$

Let $\tilde{Q} = D_s^{-\frac{1}{2}} Q_s$, so

$$\tilde{Q} = \begin{bmatrix} D_s^{-\frac{1}{2}} q_1 & \cdots & D_s^{-\frac{1}{2}} q_{n_1} \end{bmatrix} \triangleq \begin{bmatrix} q_1 & \cdots & q_{n_1} \end{bmatrix}.$$  

Eq. (23) can be rewritten as

$$\tilde{Q}^T A_s \tilde{Q} = T.$$  

As $Q$ is an orthogonal matrix,

$$\tilde{Q}^T \tilde{Q} = D_s^{-1} Q Q^T D_s^{-1} = D_s^{-1}.$$  

Premultiplying $\tilde{Q}$ on both sides of Eq. (25), we have

$$D_s^{-\frac{1}{2}} A_s \tilde{Q} = \tilde{Q} T.$$  

Equating the columns in Eq. (27), we conclude that for $k \in [1, \ldots, n_1]$,

$$D_s^{-\frac{1}{2}} A_s q_k = \beta_{k-1} \tilde{q}_{k-1} + \alpha_k \tilde{q}_k + \beta_k \tilde{q}_{k+1}.$$  

\[\]
by setting $\beta_0 \tilde{q}_0 = 0$, and $\beta_0 \tilde{q}_{k-1} = 0$.

By the orthogonality of $Q$, we have $\tilde{Q}^T D_k \tilde{Q} = I$. Premultiplying $\tilde{q}_i^T D_k$ on both sides of Eq. (28), the $D_k$-inner product orthornormality of the $\tilde{q}_i$-vectors implies

$$a_k = \tilde{q}_i^T A_i \tilde{q}_i. \quad (29)$$

Let the vector $\tilde{r}_k$ be

$$\tilde{r}_k = D_k^{-1} A_i \tilde{q}_k - \alpha_k \tilde{q}_k - \beta_k \tilde{q}_{k-1}. \quad (30)$$

If $\tilde{r}_k$ is nonzero, then by Eq. (28) we have

$$\tilde{q}_{k+1} = \tilde{r}_k / \beta_k. \quad (31)$$

With the “canonical” choice such that $\tilde{Q}^T D_k \tilde{Q} = I$,

$$\beta_k = || D_k^{-\frac{1}{2}} \tilde{r}_k ||_2. \quad (32)$$

For any unit vector $q_1$, let $\beta_0 = 0$, $\tilde{q}_0 = 0$, and $\tilde{r}_0 = D_k^{-\frac{1}{2}} q_1$.

Start from $k = 1$, we could iteratively calculate the entries of $a_k$, $\beta_k$ in $T$ until $k = n$. Meanwhile, $\tilde{Q}$ is also obtained during the iteration.

**Spectral calculation via Lanczos process.** Let $v_1$ be the eigenvector of $N_{rw}$ with the largest eigenvalue $\lambda_1$, we know

$$N_{rw} v_1 = D_k^{-\frac{1}{2}} A_i v_1 = \lambda_1 v_1. \quad (33)$$

Premultiplying $\tilde{Q}^T D_k$ on both sides, we get

$$\tilde{Q}^T A_i \tilde{Q} v_1 = \lambda_1 (\tilde{Q}^T D_k) v_1. \quad (34)$$

According to Eq. (26), $\tilde{Q}^T D_k = I$. Then by Eq. (25), the left hand side of Eq. (33) equals

$$\tilde{Q}^T A_i \tilde{Q}^2 T D_k v_1. \quad (35)$$

Let $u = \tilde{Q}^T D_k v_1$, we get

$$T u = \lambda_1 u. \quad (36)$$

On the other hand, premultiplying $\tilde{Q}$ and postmultiplying $u$ on both sides of Eq. (25), we have

$$N_{rw} \tilde{Q} u = \lambda_1 \tilde{Q} u,$$

so $\lambda_1$ is also the largest eigenvalue of $T$. As

$$v_1 = \tilde{Q} \tilde{Q}^T D_k v_1 = \tilde{Q} u,$$

we can calculate $v_1$ by calculating the eigenvector $u$ of $T$ with the largest eigenvalue $\lambda_1$.

**Lanczos iteration spectral approximation.** Instead of using the eigenvalue decomposition to get the “global spectra”, He et al. [20] use short random walks starting from the seed set to get a local proxy for the eigenvectors of $N_{rw}$, which they call the “local spectra”. Here we consider a novel way based on the Lanczos method [47] to approximate the eigenvector of $N_{rw}$ with the largest eigenvalue. A few steps of the Lanczos iteration lead to the local approximation of this eigenvector.

Let $q_1$ be the normalized indicator vector for the seed set, set $\beta_0 = 1$, $\tilde{q}_0 = 0$, and $\tilde{r}_0 = D_k^{-\frac{1}{2}} q_1$. By $k$ steps of Lanczos iteration, we could get the first $k$ by $k$ submatrix of $T$, denoted by $T_k$. Correspondingly, let the first $k$ columns of $\tilde{Q}$ be a matrix $\tilde{Q}_k$. Let the eigenvectors of $T_k$ with larger eigenvalues be a matrix $U_k$. According to Eq. (35), the columns of $V_k = \tilde{Q}_k U_k$ approximate the eigenvectors of $N_{rw}$ with larger eigenvalues. The first column of $V_k$ approximates the eigenvector of $N_{rw}$ with the largest eigenvalue, which is the indicator vector $y$ we want to find.

The Lanczos Iteration Spectral Approximation (LISA) procedure on the sampled graph is summarized in Algorithm 5.

---

The slowest step is Step 4 for calculating $\tilde{q}_k$, $\alpha_k$, $\tilde{r}_k$, and $\beta_k$. It requires $O(k2n T^2)$ time to implement the Lanczos iteration, where $k$ is the steps of Lanczos iteration and $n_t$ is the number of nodes in graph $G_t$. Also, note that the Lanczos iteration requires only a few vectors of intermediate storage.

**Algorithm 5 Lanczos Iteration Spectral Approximation**

**Input:** $G_s = (V_s, E_s)$, maximum iteration steps $k_2$, initial vector $q_1$

**Output:** $y = \tilde{Q}_s u$

1: Initialize $k = 0, \beta_0 = 1, q_0 = 0, \tilde{r}_0 = D_s^{-\frac{1}{2}} q_1$

2: while $(k < k_2)$ do

3: $k = k + 1$

4: Calculate $\tilde{q}_k, \alpha_k, \tilde{r}_k, \beta_k$ by Eq. (31), (29), (30), (32)

5: end while

6: Let $T_k$ be the first $k \times k$ entries of $T$ in Eq. (24)

7: Get the eigenvector $u$ of $T_k$ with the largest eigenvalue $\lambda$.

---

**Convergence analysis.** Here we provide an analysis on the convergence of the Lanczos process, i.e. the approximation gap between the local eigenvector which indicates the local structure around the seeds and the global eigenvector of the graph.

By Eq. (28) and Eq. (30), we conclude that for $1 \leq k < n_t$,

$$N_{rw} \tilde{Q}_s u = D_k^{-\frac{1}{2}} A_i \tilde{Q}_s = \tilde{Q}_s T_k + \tilde{r}_k e_i^T. \quad (36)$$

where $e_i$ is the $i$th unit vector with unity in the $i$th element and zero otherwise.

Let $u$ be the eigenvector of $T_k$ with the largest eigenvalue $\lambda$, premultiplying $u$ on both sides of Eq. (36), we have

$$N_{rw} \tilde{Q}_s u = \tilde{Q}_s T_k u + \tilde{r}_k e_i^T u. \quad (37)$$

Let $y = \tilde{Q}_s u$, the approximated residual value can be calculated as

$$||r_k||_2 = || N_{rw} y - Ay ||_2 = || N_{rw} \tilde{Q}_s u - \tilde{Q}_s T_k u ||_2. \quad (38)$$

By Eq. (37), Eq. (38) can be modified as

$$||r_k||_2 = || N_{rw} \tilde{Q}_s u - \tilde{Q}_s T_k u ||_2 = || \tilde{r}_k e_i^T u ||_2. \quad (39)$$

Furthermore, by Eq. (31),

$$||r_k||_2 = || \beta_k \tilde{q}_{k+1} e_i^T u ||_2 = || \beta_k D_s^{-\frac{1}{2}} q_{k+1} e_i^T u ||_2$$

$$= || \beta_k |u_k| \cdot || D_s^{-\frac{1}{2}} q_{k+1} ||_2. \quad (40)$$

where $u_k$ is the $k$th (last) term of eigenvector $u$.

As $q_{k+1}$ is a unit vector, according to the Rayleigh-Ritz theorem [52],

$$|| D_s^{-\frac{1}{2}} q_{k+1} ||_2 \leq \max_{x \neq 0} || D_s^{-\frac{1}{2}} x ||_2 = d_{min}^{-\frac{1}{2}}, \quad (41)$$

where $d_{min}$ denotes the minimum degree of the nodes in graph $G_s$, $d_{min}^{-\frac{1}{2}}$ is also the largest eigenvalue of the diagonal matrix $D_s^{-\frac{1}{2}}$.

By Eq. (40) and Eq. (41), we have

$$||r_k||_2 \leq ||d_{min}^{-\frac{1}{2}}||_2. \quad (42)$$

Generally, the higher the value of $k$ is, the smaller the residual value $||r_k||_2$ is. Especially, when $k = n_t$, $N_{rw} \tilde{Q}_s = \tilde{Q}_s T_k$, we have $||r_k||_2 = 0$. And we just need to use a small Lanczos iteration steps to find the “local” eigenvector.
4.3. Community boundary truncation

The value of the kth element of y indicates how likely node k belongs to the target community. We use a heuristic similar to [36] to determine the community boundary.

We sort the nodes based on the element values of y in the decreasing order, and find a set \( S_k \) with the first \( k \) nodes having a comparatively low conductance. Specifically, we start from an index \( k_0 \) where set \( S_{k_0} \) contains all the seeds. Then we generate a sweep curve \( \Phi(S_k) \) by increasing index \( k \). Let \( k^* \) be the value of \( k \) where \( \Phi(S_k) \) achieves a first local minimum. The set \( S_{k^*} \) is regarded as the detected community.

We determine a local minima as follows. As shown in Fig. 1, if at some point \( k^* \) when we are increasing \( k \), \( \Phi_k = \Phi(S_k) \) stops decreasing, then this \( k^* \) is a candidate point for the local minimum. If \( \Phi_k \) keeps increasing after \( k^* \) and eventually becomes higher than \( \beta \Phi_{k^*} \), then we take \( k^* \) as a valid local minimum. We experimented with several values of \( \beta \) on a small trial of data and found that \( \beta = 1.03 \) gives good performance across all the datasets.

![Fig. 1. Finding the local minimum on conductance.](image)

Algorithm 6 The overall LBSA algorithm

**Input:** \( G = (V, E) \), seed set \( S \subseteq V \)

**Output:** Community \( C = S_{k^*} \)

1. Get sampled subgraph \( G_s = (V_s, E_s) \) by Algorithm 1, Algorithm 2, or Algorithm 3
2. Calculate vector \( y \) by Algorithm 4 or Algorithm 5
3. Sort nodes by the decreasing value of elements in \( y \)
4. Find \( k_0 \) where \( S_{k_0} \) contains all the seeds
5. For \( k = k_0 : n_s \), compute the conductance \( \Phi(S_k) \): \( \Phi_k = \Phi(S_k = \{v_i | i \leq k \text{ in the sorted list}\}) \)
6. Find \( k^* \) with the first local minimum \( \Phi(S_{k^*}) \)

The overall Locally-Biased Spectral Approximation (LBSA) algorithm is shown in Algorithm 6. The overall LBSA algorithm includes six methods:

- rwPISA: adopt Algorithm 1 and Algorithm 4;
- rwLISA: adopt Algorithm 1 and Algorithm 5;
- prPISA: adopt Algorithm 2 and Algorithm 4;
- prLISA: adopt Algorithm 2 and Algorithm 5;
- hkPISA: adopt Algorithm 3 and Algorithm 4;
- hkLISA: adopt Algorithm 3 and Algorithm 5.

Without sampling on small network, LBSA algorithm only contains two methods: PISA (only adopt Algorithm 4) and LISA (only adopt Algorithm 5).

5. Experiments

In this section, we compare LBSA with several state-of-the-art local community detection algorithms, and evaluate the performance by popular \( F_1 \) score and Jaccard index.

5.1. Data description

Seven synthetic datasets (Table 3) and six real-world datasets (Table 4) are considered for a comprehensive evaluation.

5.1.1. LFR benchmark graphs

Lancichinetti et al. [54, 55] proposed a method for generating LFR benchmark graphs \(^1\) with a built-in binary community structure, which simulates properties of real-world networks on heterogeneity of node degree and community size distributions. The LFR benchmark graphs are widely used for evaluating community detection algorithms, and Xie et al. [2] performed a thorough performance comparison of different community detection algorithms on the LFR benchmark datasets.

We adopt the same set of parameter settings used in [2] and generate seven LFR benchmark graphs. Table 2 summarizes the parameter settings, among which the mixing parameter \( \mu \) controls the average fraction of neighboring nodes that do not belong to any community for each node. Each node belongs to either one community or \( om \) overlapping communities, and the number of nodes in overlapping communities is specified by \( on \). Table 3 summarizes the statistics of the seven LFR networks and their ground-truth communities. For each of the networks, we calculate the number of nodes and edges. For the ground-truth communities, we calculate the average and standard deviation of the community size, and the average conductance.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n = 5000 )</td>
<td>Number of nodes in the graph</td>
</tr>
<tr>
<td>( \mu = 0.3 )</td>
<td>Mixing parameter</td>
</tr>
<tr>
<td>( d = 10 )</td>
<td>Average degree of the nodes</td>
</tr>
<tr>
<td>( d_{\text{max}} = 50 )</td>
<td>Maximum degree of the nodes</td>
</tr>
<tr>
<td>( [20, 100] )</td>
<td>Range of the community size</td>
</tr>
<tr>
<td>( \tau_1 = 2 )</td>
<td>Node degree distribution exponent</td>
</tr>
<tr>
<td>( \tau_2 = 1 )</td>
<td>Community size distribution exponent</td>
</tr>
<tr>
<td>( om \in {2, 3, \ldots, 8} )</td>
<td>Overlapping membership</td>
</tr>
<tr>
<td>( on = 500 )</td>
<td>Number of overlapping nodes</td>
</tr>
</tbody>
</table>

\(^1\)https://sites.google.com/site/santofortunato/inthepress2
5.1.2. Real-world networks

We choose six real-world network datasets with labeled ground-truth from the SNAP collection\(^2\), namely Amazon, DBLP, LiveJ, YouTube, Orkut and Friendster in the domains of product, collaboration and social contact [36]. These datasets are widely used as ground-truth for evaluating community detection algorithms as they have thousands of annotated ground-truth communities. We just use the top 5000 ground-truth communities supplied by the SNAP collection for each dataset. Table 4 summarizes the statistics of the six real-world networks and their ground-truth communities. For each dataset, we calculate the number of nodes and edges. For the ground-truth communities, we calculate the average and standard deviation of the community size, and the average conductance, where low conductance gives priority to communities with dense internal links and sparse external links.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Name</th>
<th># Nodes</th>
<th># Edges</th>
<th>Avg. ± Std. Size</th>
<th>Avg. Cond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration</td>
<td>DBLP</td>
<td>317,080</td>
<td>1,049,866</td>
<td>22 ± 201</td>
<td>0.414</td>
</tr>
<tr>
<td>Social</td>
<td>LiveJ</td>
<td>3,977,962</td>
<td>34,681,189</td>
<td>28 ± 58</td>
<td>0.388</td>
</tr>
<tr>
<td>Social</td>
<td>YouTube</td>
<td>1,134,890</td>
<td>2,987,624</td>
<td>21 ± 73</td>
<td>0.839</td>
</tr>
<tr>
<td>Social</td>
<td>Orkut</td>
<td>3,072,441</td>
<td>117,185,083</td>
<td>216 ± 321</td>
<td>0.731</td>
</tr>
<tr>
<td>Social</td>
<td>Friendster</td>
<td>65,608,366</td>
<td>1,806,067,135</td>
<td>47 ± 70</td>
<td>0.850</td>
</tr>
</tbody>
</table>

5.2. Experimental setup

We implement the proposed LBSA method in Matlab\(^3\) through a C mex interface and conduct experiments on a computer with 2 Intel Xeon processors at 2.30GHz and 256GB memory. For the six SNAP datasets, we randomly locate 500 ground-truth communities on each dataset, and randomly pick three exemplary seeds from each target community. For the seven LFR datasets, we deal with every ground-truth community and randomly pick three exemplary seeds from each ground-truth community. We just pre-process all real-world datasets by sampling, and apply the LBSA methods for each network. To make a fair comparison, we run all baseline algorithms using the same set of random seeds.

For the parameters, we fix \((k_0, n_1) = (3, 5000)\) for Algorithm 1, \((\alpha, \epsilon_1, n_2) = (0.9, 10^{-6}, 5000)\) for Algorithm 2, and \((t, \epsilon_2, n_3) = (3, 10^{-6}, 5000)\) for Algorithm 3 such that the resulting subgraph is large enough to cover almost all the members in the target community. We set \(k_1 = 6\) for Algorithm 4 and \(k_2 = 4\) for Algorithm 5 to have a good trade-off on real-world datasets as well as the synthetic data.

5.2.1. Comparison baselines

We select four representative local community detection algorithms as the baselines. All algorithms accept as inputs an adjacency matrix \(A\) and a seed set \(S\), and run on their default parameter settings. They apply different techniques to compute diffusion ranks starting from the seed set, then perform a sweep cut on the resulting ranks.

- pppush (PR) [37]: the popular PageRank Nibble method.
- hk-relax (HK) [16]: the current best-performing heat kernel diffusion method.
- LOSP [20]: the current best-performing local spectral subspace based method.
- LEMON [24]: another local spectral subspace based method.

5.3. Experimental results

5.3.1. Statistics on sampling

Table 5, Table 6 and Table 7 show the statistics for the random walk, personalized PageRank and heat kernel sampling on the real-world datasets, respectively. As shown in Table 5, Table 6 and Table 7, the sampled subgraphs are relatively small with 3155, 3424 and 3503 nodes on average, only sampled about 0.18%, 0.45% and 0.29% of the nodes from the original datasets.

---

\(^2\)https://snap.stanford.edu/data
\(^3\)https://github.com/PanShi2016/LBSA

---

**Table 3**

Statistics for LFR networks and their ground-truth communities.

<table>
<thead>
<tr>
<th>Networks</th>
<th>Ground-truth communities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td># Nodes</td>
</tr>
<tr>
<td>LFR ((om = 2))</td>
<td>5,000</td>
</tr>
<tr>
<td>LFR ((om = 3))</td>
<td>5,000</td>
</tr>
<tr>
<td>LFR ((om = 4))</td>
<td>5,000</td>
</tr>
<tr>
<td>LFR ((om = 5))</td>
<td>5,000</td>
</tr>
<tr>
<td>LFR ((om = 6))</td>
<td>5,000</td>
</tr>
<tr>
<td>LFR ((om = 7))</td>
<td>5,000</td>
</tr>
<tr>
<td>LFR ((om = 8))</td>
<td>5,000</td>
</tr>
</tbody>
</table>

**Table 4**

Statistics for real-world networks and their ground-truth communities.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Network</th>
<th># Nodes</th>
<th># Edges</th>
<th>Ground-truth communities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td></td>
<td></td>
<td>Avg. ± Std. Size</td>
</tr>
<tr>
<td>Product</td>
<td>Amazon</td>
<td>334,863</td>
<td>925,872</td>
<td>13 ± 18</td>
</tr>
<tr>
<td>Collaboration</td>
<td>DBLP</td>
<td>317,080</td>
<td>1,049,866</td>
<td>22 ± 201</td>
</tr>
<tr>
<td>Social</td>
<td>LiveJ</td>
<td>3,977,962</td>
<td>34,681,189</td>
<td>28 ± 58</td>
</tr>
<tr>
<td>Social</td>
<td>YouTube</td>
<td>1,134,890</td>
<td>2,987,624</td>
<td>21 ± 73</td>
</tr>
<tr>
<td>Social</td>
<td>Orkut</td>
<td>3,072,441</td>
<td>117,185,083</td>
<td>216 ± 321</td>
</tr>
<tr>
<td>Social</td>
<td>Friendster</td>
<td>65,608,366</td>
<td>1,806,067,135</td>
<td>47 ± 70</td>
</tr>
</tbody>
</table>
graph. Nevertheless, there are very high coverage ratio (ratio of ground-truth nodes covered by the subgraph) of 97.0%, 96.9% and 97.7%, and the sampling procedure are very fast in 1.301, 0.051 and 0.309 seconds. Additionally, Table 5 shows that the sampled subgraph is very small with only 41 nodes on Amazon but with very high coverage ratio of 99.1%. On the other hand, the random walk sampling procedure on the Friendster is longer than that of the other two, as shown in Table 6 and Table 7. In summary, the three methods all work well for sampling; heat kernel is slightly better among the three methods with higher average coverage ratio and short sampling time.

### 5.3.2. Convergence analysis

As LBSA algorithm involves the power iteration and Lanczos iteration, we experimentally investigate the convergence property of Algorithm 4 and Algorithm 5 on LFR benchmark graphs as well as real-world networks. For the output of each iteration, we calculate the residual value $\|r_i\|_2$ for Algorithm 4 and $\|r_i\|_2$ for Algorithm 5.

Fig. 2 shows the convergence analysis on LFR networks with power iteration and Lanczos iteration. One can see that the Lanczos iteration converges more quickly than the power iteration, as shown in Fig. 2. Specifically, Fig. 2 (a) indicates that the residual value $\|r_i\|_2$ converges to zero for more than 20 iterations. However, Fig. 2 (b) illustrates that the residual value $\|r_i\|_2$ converges to zero only for more than 10 iterations.

Fig. 3 shows the convergence analysis on real-world networks with power iteration and Lanczos iteration. Overall, as shown in Fig. 3, the Lanczos iteration runs more stably and converges more quickly than the power iteration. Specifically, Fig. 3 (b), Fig. 3 (d) and Fig. 3 (f) show that the residual values $\|r_i\|_2$ converge to zero only for more than 20 iterations. However, Fig. 3 (a), Fig. 3 (c) and Fig. 3 (e) indicate that the residual values $\|r_i\|_2$ converge to zero for more than 40 iterations. Additionally, the power iteration runs unstably on YouTube, Orkut and Friendster, as shown in Fig. 3 (a), Fig. 3 (c) and Fig. 3 (e).

As the Lanczos iteration converges more quickly than the power iteration on real-world networks as well as LFR datasets, we set the iteration steps $k_1 = 6$ for Algorithm 4 and $k_2 = 4$ for Algorithm 5 to gain a “local” eigenvector indicating the implicit topology structure of the local region around the seeds.

### 5.3.3. Accuracy comparison on LFR datasets

Fig. 4 illustrates the average detection accuracy of our methods and the baselines on the LFR datasets. As shown in Fig. 4 (a) and Fig. 4 (b), PISA and LISA significantly outperform all baseline methods evaluated by $F_1$ score and Jaccard index on all the seven synthetic networks, and PISA achieves the best accuracy but HK and PR are the last two place on all LFR datasets. Additionally, one can see that all algorithms tend to achieve a lower accuracy with a larger overlapping membership om. As on = 500 overlapping nodes are assigned to om $\in \{2, 3, ... 8\}$ communities, a larger om makes the detection more difficult, this may explain why all algorithms tend to achieve a lower accuracy with a larger om.

Fig. 5 and Table 8 show more comparisons on LFR datasets. Fig. 5 illustrates the average conductance of the detected communities on LFR datasets. One can see that HK and PR tend to achieve the lower conductance, and PISA and LISA get the most consistent results as compared with conductance of ground-truth communities shown in Table 3. However, as compared with ground-truth communities shown in Table 3, Table 8 shows that PISA uncovers more suitable size of the detected communities compared to LISA, and HK and PR attain much large size of the detected communities. These comparison analysis may explain why PISA and LISA significantly outperform all baseline methods, and PISA achieves the best accuracy but HK and PR perform worse.

For the running time comparison, as the graph loading time is the same for all algorithms, we omit the loading time for our algorithms and all baselines to show their differences on the actual processing. Table 8 shows that all algorithms are very fast and run in seconds. On average, HK is the fastest in 0.025 seconds, and the other five run in 0.1 to 5.1 seconds. PISA and LISA costs several more seconds as compared with other algorithms, as they involve finding a community with the local minimal conductance. Also, different methods are implemented in

---

### Table 5

Statistics of the average values for the random walk sampling.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Coverage</th>
<th>$n_s$</th>
<th>$n_s/n$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.991</td>
<td>41</td>
<td>$1.2 \times 10^{-4}$</td>
<td>0.002</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.987</td>
<td>1347</td>
<td>$4.2 \times 10^{-3}$</td>
<td>0.002</td>
</tr>
<tr>
<td>LiveJ</td>
<td>0.999</td>
<td>2780</td>
<td>$7.0 \times 10^{-4}$</td>
<td>0.045</td>
</tr>
<tr>
<td>YouTube</td>
<td>0.935</td>
<td>4801</td>
<td>$4.2 \times 10^{-3}$</td>
<td>0.137</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.914</td>
<td>4985</td>
<td>$1.6 \times 10^{-3}$</td>
<td>0.542</td>
</tr>
<tr>
<td>Friendster</td>
<td>0.994</td>
<td>4977</td>
<td>$7.6 \times 10^{-5}$</td>
<td>0.076</td>
</tr>
<tr>
<td>Average</td>
<td>0.970</td>
<td>3155</td>
<td>$1.8 \times 10^{-4}$</td>
<td>1.301</td>
</tr>
</tbody>
</table>

### Table 6

Statistics of the average values for the personalized PageRank sampling.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Coverage</th>
<th>$n_s$</th>
<th>$n_s/n$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>1.000</td>
<td>2054</td>
<td>$6.1 \times 10^{-4}$</td>
<td>0.014</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.992</td>
<td>4637</td>
<td>$1.5 \times 10^{-2}$</td>
<td>0.051</td>
</tr>
<tr>
<td>LiveJ</td>
<td>0.999</td>
<td>2661</td>
<td>$6.7 \times 10^{-4}$</td>
<td>0.036</td>
</tr>
<tr>
<td>YouTube</td>
<td>0.966</td>
<td>4973</td>
<td>$4.4 \times 10^{-3}$</td>
<td>0.081</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.882</td>
<td>3493</td>
<td>$1.1 \times 10^{-3}$</td>
<td>0.060</td>
</tr>
<tr>
<td>Friendster</td>
<td>0.977</td>
<td>2727</td>
<td>$4.2 \times 10^{-5}$</td>
<td>0.066</td>
</tr>
<tr>
<td>Average</td>
<td>0.969</td>
<td>3424</td>
<td>$4.5 \times 10^{-4}$</td>
<td>0.051</td>
</tr>
</tbody>
</table>

### Table 7

Statistics of the average values for the heat kernel sampling.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Coverage</th>
<th>$n_s$</th>
<th>$n_s/n$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.999</td>
<td>449</td>
<td>$1.3 \times 10^{-4}$</td>
<td>0.002</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.991</td>
<td>3034</td>
<td>$9.6 \times 10^{-3}$</td>
<td>0.028</td>
</tr>
<tr>
<td>LiveJ</td>
<td>0.999</td>
<td>2639</td>
<td>$6.6 \times 10^{-4}$</td>
<td>0.076</td>
</tr>
<tr>
<td>YouTube</td>
<td>0.969</td>
<td>4949</td>
<td>$4.4 \times 10^{-3}$</td>
<td>0.447</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.908</td>
<td>4990</td>
<td>$1.6 \times 10^{-3}$</td>
<td>0.511</td>
</tr>
<tr>
<td>Friendster</td>
<td>0.995</td>
<td>4954</td>
<td>$7.6 \times 10^{-5}$</td>
<td>0.789</td>
</tr>
<tr>
<td>Average</td>
<td>0.977</td>
<td>3503</td>
<td>$2.9 \times 10^{-4}$</td>
<td>0.309</td>
</tr>
</tbody>
</table>

---

### 5.3.2. Convergence analysis

As LBSA algorithm involves the power iteration and Lanczos iteration, we experimentally investigate the convergence property of Algorithm 4 and Algorithm 5 on LFR benchmark graphs as well as real-world networks. For the output of each iteration, we calculate the residual value $\|r_i\|_2$ for Algorithm 4 and $\|r_i\|_2$ for Algorithm 5.

Fig. 2 shows the convergence analysis on LFR networks with power iteration and Lanczos iteration. One can see that...
Fig. 2. Convergence analysis on LFR networks with power iteration and Lanczos iteration.

Fig. 3. Convergence analysis on real-world networks with power iteration and Lanczos iteration.
overlapping membership
0.0
0.2
0.4
0.6
0.8
1.0
F1 score
PISA
LISA
LEMON
LOSP
HK
PR

Fig. 4. Accuracy comparison on LFR datasets.

overlapping membership
0.0
0.2
0.4
0.6
0.8
1.0
Jaccard index
PISA
LISA
LEMON
LOSP
HK
PR

Fig. 5. Conductance comparison on LFR datasets.

Table 8
Average size of the detected communities and average running time of the algorithms on LFR datasets.

<table>
<thead>
<tr>
<th></th>
<th>Size</th>
<th></th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PISA</td>
<td>LISA</td>
<td>LEMON</td>
</tr>
<tr>
<td>LFR (om = 2)</td>
<td>42</td>
<td>232</td>
<td>16</td>
</tr>
<tr>
<td>LFR (om = 3)</td>
<td>41</td>
<td>269</td>
<td>14</td>
</tr>
<tr>
<td>LFR (om = 4)</td>
<td>144</td>
<td>699</td>
<td>16</td>
</tr>
<tr>
<td>LFR (om = 5)</td>
<td>110</td>
<td>597</td>
<td>18</td>
</tr>
<tr>
<td>LFR (om = 6)</td>
<td>68</td>
<td>926</td>
<td>17</td>
</tr>
<tr>
<td>LFR (om = 7)</td>
<td>186</td>
<td>1137</td>
<td>17</td>
</tr>
<tr>
<td>LFR (om = 8)</td>
<td>229</td>
<td>1305</td>
<td>16</td>
</tr>
<tr>
<td>Average</td>
<td>117</td>
<td>738</td>
<td>16</td>
</tr>
</tbody>
</table>
Fig. 6. Accuracy comparison on real-world networks.

Fig. 7. Conductance comparison on real-world networks.

Table 9
Average size of the detected communities of the algorithms on real-world networks.

<table>
<thead>
<tr>
<th></th>
<th>rwPISA</th>
<th>rwLISA</th>
<th>prPISA</th>
<th>prLISA</th>
<th>hkPISA</th>
<th>hkLISA</th>
<th>LEMON</th>
<th>LOSP</th>
<th>HK</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>9</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>13</td>
<td>8</td>
<td>48</td>
<td>4485</td>
</tr>
<tr>
<td>DBLP</td>
<td>42</td>
<td>7</td>
<td>19</td>
<td>7</td>
<td>19</td>
<td>7</td>
<td>20</td>
<td>20</td>
<td>87</td>
<td>9077</td>
</tr>
<tr>
<td>LiveJ</td>
<td>258</td>
<td>98</td>
<td>54</td>
<td>31</td>
<td>73</td>
<td>31</td>
<td>26</td>
<td>28</td>
<td>119</td>
<td>512</td>
</tr>
<tr>
<td>YouTube</td>
<td>1875</td>
<td>113</td>
<td>658</td>
<td>290</td>
<td>750</td>
<td>309</td>
<td>18</td>
<td>10</td>
<td>122</td>
<td>13840</td>
</tr>
<tr>
<td>Orkut</td>
<td>2430</td>
<td>744</td>
<td>618</td>
<td>236</td>
<td>1319</td>
<td>444</td>
<td>9</td>
<td>17</td>
<td>341</td>
<td>1648</td>
</tr>
<tr>
<td>Friendster</td>
<td>2225</td>
<td>1174</td>
<td>409</td>
<td>161</td>
<td>1035</td>
<td>553</td>
<td>9</td>
<td>13</td>
<td>107</td>
<td>752</td>
</tr>
<tr>
<td>Average</td>
<td>1140</td>
<td>357</td>
<td>295</td>
<td>122</td>
<td>535</td>
<td>225</td>
<td>16</td>
<td>16</td>
<td>137</td>
<td>5052</td>
</tr>
</tbody>
</table>

Table 10
Average running time measured by seconds of the algorithms on real-world networks.

<table>
<thead>
<tr>
<th></th>
<th>rwPISA</th>
<th>rwLISA</th>
<th>prPISA</th>
<th>prLISA</th>
<th>hkPISA</th>
<th>hkLISA</th>
<th>LEMON</th>
<th>LOSP</th>
<th>HK</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.002</td>
<td>0.002</td>
<td>0.665</td>
<td>0.496</td>
<td>0.051</td>
<td>0.039</td>
<td>0.039</td>
<td>0.117</td>
<td>0.008</td>
<td>0.795</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.547</td>
<td>0.427</td>
<td>2.556</td>
<td>1.848</td>
<td>1.366</td>
<td>0.990</td>
<td>0.271</td>
<td>0.127</td>
<td>0.026</td>
<td>0.581</td>
</tr>
<tr>
<td>LiveJ</td>
<td>3.654</td>
<td>2.430</td>
<td>1.873</td>
<td>1.167</td>
<td>2.099</td>
<td>1.257</td>
<td>0.553</td>
<td>1.092</td>
<td>0.018</td>
<td>0.279</td>
</tr>
<tr>
<td>YouTube</td>
<td>9.445</td>
<td>7.218</td>
<td>4.098</td>
<td>2.549</td>
<td>6.313</td>
<td>3.742</td>
<td>1.458</td>
<td>0.812</td>
<td>0.035</td>
<td>0.241</td>
</tr>
<tr>
<td>Orkut</td>
<td>13.566</td>
<td>8.651</td>
<td>4.132</td>
<td>2.097</td>
<td>9.913</td>
<td>5.138</td>
<td>1.328</td>
<td>1.960</td>
<td>0.025</td>
<td>0.091</td>
</tr>
<tr>
<td>Friendster</td>
<td>14.693</td>
<td>10.308</td>
<td>5.903</td>
<td>4.681</td>
<td>10.194</td>
<td>4.410</td>
<td>1.649</td>
<td>28.980</td>
<td>0.116</td>
<td>0.098</td>
</tr>
<tr>
<td>Average</td>
<td>6.985</td>
<td>4.839</td>
<td>3.205</td>
<td>2.140</td>
<td>4.974</td>
<td>2.596</td>
<td>0.883</td>
<td>5.515</td>
<td>0.038</td>
<td>0.348</td>
</tr>
</tbody>
</table>
different languages, so the running time could give an indication of the overall trend, and it can not be compared directly.

5.3.4. Accuracy comparison on real-world networks

Fig. 6 illustrates the average detection accuracy on real-world networks. As shown in Fig. 6 (a) and Fig. 6 (b), prLISA and hkLISA outperform all baseline methods evaluated by $F_1$ score and Jaccard index on all six real-world networks. rwPISA, prPISA and hkPISA outperform all baseline methods on Amazon and LiveJ, and rwLISA outperforms all baseline methods on Amazon and LiveJ. On the other hand, LEMON shows the lowest accuracy on LiveJ and Orkut, HK shows the worst performance on YouTube and Friendster, and PR is in the last place on Amazon and DBLP. Though rwPISA, rwLISA, prPISA and hkPISA do not outperform all baseline methods on all six real-world networks, they are the most robust methods and very competitive on average. As a whole, prLISA and hkLISA perform the better on the six real-world datasets while HK and PR are in the last two place.

Fig. 7 and Table 9 show more comparisons on real-world datasets. As shown in Fig. 7, LEMON obtains the highest conductance on LiveJ, which is much bigger than the ground-truth as shown in Table 4. Table 9 shows that LEMON achieves much small size of the detected communities on Orkut compared with ground-truth communities, as shown in Table 4. This may explain why LEMON shows the worst performance on LiveJ and Orkut. Fig. 7 also indicates that HK achieves the lowest conductance on YouTube and Friendster, which are very small as compared with ground-truth communities shown in Table 4. This may explain why HK achieves the lowest accuracy on YouTube and Friendster. Table 9 also illustrates that PR attains much large size of the detected communities on Amazon and DBLP as compared with ground-truth communities shown in Table 4. This may explain why PR is in the last place on Amazon and DBLP. On the other hand, as shown in Fig. 7, prLISA and hkLISA attain the modest conductance as compared with ground-truth communities shown in Table 4. This may explain why prLISA and hkLISA outperform all baseline methods evaluated by $F_1$ score and Jaccard index on all six real-world networks.

For the running time comparison, as the graph loading time is the same for all algorithms, we omit the loading time for our algorithms and all baselines to show their differences on the actual processing. Table 10 illustrates that all algorithms are very fast and run in seconds. On average, HK is the fastest in 0.038 seconds, and the other nine algorithms run in 0.3 to 6.9 seconds. Our methods costs several more seconds as compared with HK and PR, as they involve finding a community with the local minimal conductance. Also, different methods are implemented in different languages, so the running time could give an indication of the overall trend, and it can not be compared directly.

6. Conclusions and future works

In this paper, we propose a Locally-Biased Spectral Approximation (LBSA) approach for local community detection. Firstly, we utilize the very fast random walk, personalized PageRank and heat kernel diffusion to get a local sampled subgraph that largely reduces the complexity of the subsequent computation. Based on Rayleigh quotient and conductance, we provide theoretical base for the proposed method. In addition, we adopt power iteration and Lanczos iteration to get a “local” eigenvector. To the best of our knowledge, this is the first time to apply Lanczos method for local community detection. The convergence analysis results show that Lanczos iteration is more robustly and converges more quickly than the power iteration. Extensive experimental results on the real-word networks as well as synthetic LFR datasets also show that the LBSA algorithm could be a new and effective way to detect local communities in large graphs. We wish our work inspire more researches based on the locally-biased spectral approximation method for network analysis and community detection.

Acknowledgements

This work was supported by National Natural Science Foundation of China (61772219, 61472147, 61572221).

Appendix A. The proof of Theorem 1

Before given the proof of Theorem 1, we first prove the following proposition.

Proposition 1. Let $\lambda_2$ be the second smallest eigenvalue of $L_{sym}$ for a graph $G_s$, then

$$\lambda_2 = \min_{x \perp 0} \frac{x^T L_{sym} x}{x^T x},$$

(A.1)

where $D_s$ denote the diagonal degree matrix of $G_s$, and $e_s$ the vector of length $n_s$ with ones as the entries.

Proof. As $L_{sym}$ is a symmetric matrix, let $L_{sym} = \tilde{\Phi} \bar{A} \tilde{\Phi}^T$ be the eigenvalue decomposition, where $\tilde{\Phi} = [\tilde{\phi}_1, \tilde{\phi}_2, ..., \tilde{\phi}_{n_s}]$ is an orthogonal matrix and $\bar{A} = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_{n_s})$, $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{n_s}$. Then

$$\frac{x^T L_{sym} x}{x^T x} = \frac{x^T (\tilde{\Phi} \bar{A} \tilde{\Phi}^T) x}{x^T (\tilde{\Phi} \tilde{\Phi}^T) x} = \frac{(\tilde{\Phi}^T x)^T \bar{A} \tilde{\Phi}^T x}{(\tilde{\Phi}^T x)^T (\tilde{\Phi}^T x)}.$$  

(A.2)

Let $w = \tilde{\Phi}^T x$, we have

$$\frac{x^T L_{sym} x}{x^T x} = \frac{w^T \bar{A} w}{w^T w} = \frac{\sum_{i=1}^{n_s} \lambda_i w_i^2}{\sum_{i=1}^{n_s} w_i^2},$$

(A.3)

where $w_i$ denotes the $i$th element of the vector $w$.

As $L_{sym}(D_s^\frac{1}{2} e_s) = 0$, we have $\tilde{\phi}_1 = \frac{D_s^\frac{1}{2} e_s}{\|D_s^\frac{1}{2} e_s\|}$. Let $x = \perp D_s^\frac{1}{2} e_s$, $w_1 = \tilde{\phi}_1^T x = 0$. Then by Eq. (A.3),

$$\frac{x^T L_{sym} x}{x^T x} = \frac{\sum_{i=2}^{n_s} \lambda_i w_i^2}{\sum_{i=2}^{n_s} w_i^2} \geq \frac{\sum_{i=2}^{n_s} \lambda_2 w_i^2}{\sum_{i=2}^{n_s} w_i^2} = \lambda_2.$$  

(A.4)

Additionally, as $\tilde{\phi}_2^T x = 0$, we have $\tilde{\phi}_2 \perp D_s^\frac{1}{2} e_s$. By $L_{sym} \tilde{\phi}_2 = \lambda_2 \tilde{\phi}_2$ and $\|\tilde{\phi}_2\| = 1$,

$$\lambda_2 \geq \frac{\tilde{\phi}_2^T L_{sym} \tilde{\phi}_2}{\tilde{\phi}_2^T \tilde{\phi}_2} \geq \min_{x \perp 0} \frac{x^T L_{sym} x}{x^T x}.$$  

(A.5)
By Eq. (A.4) and Eq. (A.5), we have
\[ \tilde{\lambda}_2 = \min_{\text{sym}} \frac{x^T \text{sym} x}{x^T x}. \] (A.6)

**Proof of Theorem 1.** By Proposition 1, we have
\[ \lambda_2 = \min_{\text{sym}} \frac{x^T \text{sym} x}{x^T x} \leq \frac{\sum_{i,j} (z_i - z_j)^2}{\min_{\text{sym}} \{\text{vol}(\tilde{V})\^2\}}. \] (A.7)

where \( z = D^{-1} x, d_i \) is the degree of the \( i \)-th node and \( \sum_{i,j} \) denotes the sum over all unordered pairs \( \{i,j\} \) for which \( i \) and \( j \) are adjacent.

We choose an optimum subset \( \tilde{V} \subset V \), which achieves \( \phi(G) \) and separates the graph \( G \) into two parts, \( \tilde{V} \) and \( \tilde{V} \). Let
\[ z_i = \begin{cases} \frac{1}{\text{vol}(\tilde{V})} & \text{if the } i \text{-th node is in } \tilde{V}, \\ \frac{1}{\text{vol}(\tilde{V})} & \text{if the } i \text{-th node is in } \tilde{V}. \end{cases} \] (A.8)

Here \( \text{vol}(\tilde{V}) \) indicate the total degrees of nodes inside \( \tilde{V} \) in graph \( G \).

As \( z \perp D_e x_s \), by Eq. (A.7) and Eq. (A.8), we have the following:
\[ \lambda_2 \leq \text{cut}(\tilde{V}, \tilde{V}) (1/\text{vol}(\tilde{V}) + 1/\text{vol}(\tilde{V})) \leq 2 \text{cut}(\tilde{V}, \tilde{V}) \min\{\text{vol}(\tilde{V}), \text{vol}(\tilde{V})\} = 2 \phi(G), \] (A.9)

where \( \text{cut}(\tilde{V}, \tilde{V}) \) denotes the number of edges between \( \tilde{V} \) and \( \tilde{V} \).

From Eq. (A.9), we have
\[ \phi(G) \geq \frac{\lambda_2}{2}. \] (A.10)

---

References

[40] F. Chung, O. Simpson, Solving linear systems with boundary conditions.


Locally-biased Spectral Approximation for Community Detection

Pan Shi\textsuperscript{a}, Kun He\textsuperscript{a,∗}, David Bindel\textsuperscript{b}, John E. Hopcroft\textsuperscript{b}

\textsuperscript{a}School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
\textsuperscript{b}Department of Computer Science, Cornell University, Ithaca, NY 14853, USA

Abstract

We propose a Locally-Biased Spectral Approximation (LBSA) approach for identifying all latent members of a local community from very few seed members. To reduce the computation complexity, we first apply a fast random walk, personalized PageRank and heat kernel diffusion to sample a comparatively small subgraph covering almost all potential community members around the seeds. Then starting from a normalized indicator vector of the seeds and by a few steps of either Lanczos iteration or power iteration on the sampled subgraph, a local eigenvector is gained for approximating the eigenvector of the transition matrix with the largest eigenvalue. Elements of this local eigenvector is a relaxed indicator for the affiliation probability of the corresponding nodes to the target community. We conduct extensive experiments on real-world datasets in various domains as well as synthetic datasets. Results show that the proposed method outperforms state-of-the-art local community detection algorithms. To the best of our knowledge, this is the first work to adapt the Lanczos method for local community detection, which is natural and potentially effective. Also, we did the first attempt of using heat kernel as a sampling method instead of detecting communities directly, which is proved empirically to be very efficient and effective.

Keywords: Community detection, Local spectral approximation, Power iteration, Lanczos method

1. Introduction

Community detection is an important task in network analysis, which aims to find a set of nodes in a network that are internally cohesive but comparatively separated from the remainder of the network. In social networks, community detection is a classical and challenging problem which is very useful for analyzing the topology structure and extracting information from the network, and numerous algorithms and techniques have been proposed [1, 2, 3, 4, 5, 6].

Over the past decades, most of the researchers have focused on uncovering the global community structure, including modularity maximization model [7, 8, 9, 10], spectral method [11, 12], stochastic model [13, 14] and so on. With the rapid growth of the network scale, global community detection becomes very costly or even impossible for very large networks. The big data drives researchers to shift their attention from the global structure to the local structure [15, 16, 17, 18, 19]. How to adapt the existing effective methods initially designed for the global community detection in order to uncover the local community structure is a natural and important approach for the accurate membership identification from a few exemplary members. Several probability diffusion methods, PageRank [15], heat kernel [16] and spectral subspace approximation [20, 21, 22, 23, 24] are three main techniques for local community detection.

The power iteration [25] and Lanczos iteration [26] are two classic methods proposed for calculating the eigenvalues, aka the spectra of a matrix. As a simple eigenvalue calculation method, power method has been widely used for spectral clustering [27, 28]. On the other hand, though there exists some work using the Lanczos method for the spectral bisection [29], unlike other spectra calculation methods, the Lanczos method is seldom used for community detection and to the best of our knowledge, it has never been used for the local community detection. In this paper, we propose a novel approach called the Locally-Biased Spectral Approximation (LBSA) for local community detection. Specifically, we execute a few steps of power iteration or Lanczos iteration to attain a local eigenvector that approximates the eigenvector of the transition matrix with the largest eigenvalue. Elements of this local eigenvector is a relaxed indicator for the affiliation probability of the corresponding nodes to the target community. As compared with other spectral approximation methods, the power method and Lanczos iterative method are efficient for computing the top eigenpairs of large sparse matrices and they are space efficient, which are very helpful for large and usually sparse social networks.

This paper is a significant extension of our ECML-PKDD conference paper [30]. The conference paper uses heat kernel diffusion as a sampling method, and does Lanczos iteration for local community detection. We did plenty extensions on methods, analysis and experiments so as to have an integrate method and extensive analysis. Our new contributions include: (1) Besides heat kernel, we also apply a fast random walk and personalized PageRank diffusion to sample a localized subgraph to largely reduce the subsequent calculation. (2) Compared to Lanczos method, we also adopt power method for
locally-biased spectral approximation, and provide some sound convergence analysis. (3) Based on more baseline algorithms and datasets, we conduct extensive experiments on real-world networks as well as synthetic datasets to demonstrate the superiority of the proposed method.

2. Related work

2.1. Local community detection

Techniques for local community detection can be classified into three categories, namely the PageRank, heat kernel and local spectral methods. Other techniques like finding minimum cut [31, 32, 33] can also be used for local community detection.

PageRank. The PageRank method is widely used for local community detection. Spielman and Teng [34] use degree-normalized, personalized PageRank (DN PageRank) with respect to the initial seeds and do truncation on small probability values. DN PageRank is adopted by several competitive PageRank based clustering algorithms [35, 36], including the popular PageRank Nibble method [37]. Kloumann and Kleinberg [15] evaluate different variations of PageRank method and find that the standard PageRank yields higher performance than the DN PageRank.

Heat kernel. The heat kernel method involves the Taylor series expansion of the exponential of the transition matrix. Chung [38, 39] provides a theoretical analysis and a local graph partitioning algorithm based on heat kernel diffusion. Chung and Simpson [40] propose a randomized Monte Carlo method to estimate the diffusion speed, and Kloster and Gleich [16] propose a deterministic method that uses coordinate relaxation on an implicit linear system to estimate the heat kernel diffusion, and the heat value of each node represents the likelihood of affiliation.

Local spectral. A third branch is to adapt the classic spectral method to locate the target community. Mahoney et al. [41] introduce a locally-biased analogue of the second eigenvector, the Fiedler vector associated with the algebraic connectivity, to extract local properties of data graphs, and apply the method for a semi-supervised image segmentation and a local community extraction by finding a sparse-cut around the seeds in small social networks. He et al. [20, 22, 23] and Li et al. [21, 24] extract the local community by seeking a sparse vector from the local spectral subspaces using $\ell_1$ norm optimization.

2.2. Spectral calculation

2.2.1. Power method

One of the oldest techniques for solving eigenvalue problems is the so-called power method [25]. The power method is an iterative technique used to determine the dominant eigenvalue of a matrix, that is, the eigenvalue with the largest magnitude. By modifying the method slightly, it can also used to determine other eigenvalues [42]. One useful feature of the power method is that it produces not only an eigenvalue, but also an associated eigenvector. In fact, the power method is often applied to find an eigenvector for an eigenvalue that is determined by some other means.

As a simple and classic eigenvalue calculation method, the power method has been applied in many field. For example in the application of graph clustering and feature selection, Lin and Cohen [27] propose a simple graph clustering algorithm which finds a very low-dimensional embedding of a dataset using truncated power iteration; Thang et al. [43] illustrate a novel clustering method based on the deflation power iteration technique to compute multiple orthogonal pseudo-eigenvectors; Huang et al. [44] propose diverse power iteration embedding method for feature selection.

2.2.2. Lanczos method

Many real world problems can be modeled as sparse graphs and be represented as matrices, and the eigenvalue calculation of the matrices is usually a crucial step for the problem solving. All the eigenpairs can be calculated by eigenvalue decomposition [42], SVD [45], or QR factorization [46]. However, these methods are intractable for large matrices due to the high complexity and memory consumption. As the Lanczos method can significantly reduce the time and space complexity, it is usually applied to large sparse matrices [47].

As a classic eigenvalue calculation method, the original Lanczos method [26] cannot hold the orthogonality of the calculated Krylov subspace and it is not widely used in practice. Paige [47] computes the eigenpairs for very large sparse matrices by an improved Lanczos method, as only a few iterations are typically required to get a good approximation on the extremal eigenvalues. Thereafter, the Lanczos method becomes very attractive for large sparse matrix approximation. For example in the application of graph partitioning and image reconstruction, Barnes [29] illustrates that Lanczos method is an efficient implementation of the spectral bisection method; Wu et al. [48] propose an incremental bilinear Lanczos algorithm for high dimensionality reduction and image reconstruction; Bentbib et al. [49] illustrate that efficient image restoration can be achieved by Tikhonov regularization based on the global Lanczos method.

To the best of our knowledge, there exist no Lanczos based algorithms for local community detection in the literature.

3. Preliminaries

3.1. Problem formulation

In a network, we want to identify a target community supervised by a few observed members. We call the problem the local community detection. It can be formalized as follows. We are given a connected, unweighted and undirected graph $G = (V, E)$ with $n$ nodes and $m$ edges. Let $A \in \{0, 1\}^{n \times n}$ be the symmetric associated adjacency matrix, and $D$ the diagonal matrix of node degrees. Let $S$ be the seed set of a few exemplary members in the target ground-truth community, denoted by a set of nodes $T \subset T$, $|T| \ll |V|$. Let $s \in \{0, 1\}^{|S|}$ be a binary indicator vector representing the exemplary members in $S$. We are asked to identify the remaining latent members in the ground-truth community $T$ as accurate as possible.

For accuracy evaluation, we adopt $F_1$ score and Jaccard index to quantify the similarity between the detected local community $C$ and the target ground-truth community $T$. The $F_1$
score for each pair of \((C, T)\) is defined by:
\[
F_1(C, T) = \frac{2 \cdot P(C, T) \cdot R(C, T)}{P(C, T) + R(C, T)},
\]
where the precision \(P\) and recall \(R\) are defined as:
\[
P(C, T) = \frac{|C \cap T|}{|C|}, \quad R(C, T) = \frac{|C \cap T|}{|T|}.
\]
The Jaccard index for each pair of \((C, T)\) is defined by:
\[
J(C, T) = \frac{|C \cap T|}{|C \cup T|}.
\]

3.2. Notations and descriptions

Table 1 summarizes a list of different notations we will use throughout the paper. In general, we use italic lowercase letters, e.g., \(n\), to denote scalars; lowercase boldface characters, e.g., \(y\), to denote vectors; uppercase boldface characters, e.g., \(A\), to denote matrices; and italic capital letters, e.g., \(C\), to denote sets.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G)</td>
<td>Graph</td>
</tr>
<tr>
<td>(A)</td>
<td>Adjacency matrix of graph (G)</td>
</tr>
<tr>
<td>(D)</td>
<td>Diagonal degree matrix of graph (G)</td>
</tr>
<tr>
<td>(S)</td>
<td>Seed set</td>
</tr>
<tr>
<td>(T)</td>
<td>Ground-truth community</td>
</tr>
<tr>
<td>(C)</td>
<td>Detected community</td>
</tr>
<tr>
<td>(G_s)</td>
<td>Subgraph extracted from the neighborhood surrounding the seed set (S)</td>
</tr>
<tr>
<td>(A_s)</td>
<td>Adjacency matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(D_s)</td>
<td>Diagonal degree matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(L_{sym})</td>
<td>Normalized Laplacian matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(N_{rw})</td>
<td>Transition matrix of subgraph (G_s)</td>
</tr>
<tr>
<td>(\Phi(\bar{V}))</td>
<td>Conductance of a node set (\bar{V})</td>
</tr>
<tr>
<td>(\lambda_i)</td>
<td>The (i)th largest eigenvalue of matrix (N_{rw})</td>
</tr>
<tr>
<td>(\lambda_i)</td>
<td>The (i)th smallest eigenvalue of matrix (L_{sym})</td>
</tr>
<tr>
<td>(y)</td>
<td>Indicator vector, where larger value indicates higher probability being in the same community</td>
</tr>
</tbody>
</table>

4. Locally-biased spectral approximation methods

In this section, there are three key steps in the proposed algorithms: local sampling, locally-biased spectral approximation and community boundary truncation.

4.1. Local sampling

In this subsection, we adopt fast random walk, personalized PageRank and heat kernel diffusion to do three types of the sampling, and set parameters such that the resulting subgraph is large enough to cover almost all members in the target community.

4.1.1. Local random walk sampling

We do \(k_0\) steps of standard random walks, and define the \(k_0\)-walk diffusion vector as
\[
p_{k_0} = (AD^{-1})^{k_0} p_0,
\]
where \(p_0 = s/|S|\) is the initial probability distribution on the source seeds.

For a large graph, it is unpractical to compute the matrix \(AD^{-1}\) directly, so we implement a fast method to update the probability vector \(p_{k_0}\). Details of the sampling are as shown in Algorithm 1. For notations in Algorithm 1, at iteration \(t\), \(p_i(s_j)\) denotes the probability on the specified node \(s_j\), and \(p_i\) without parameter denotes the probability vector for nodes 1 to \(n_i\).

Algorithm 1 The random walk sampling

Input: graph \(G = (V, E)\), seed set \(S \subset V\), steps of standard random walks \(k_0\), upper bound of the subgraph size \(n_1\)

Output: Sampled subgraph \(G_s = (V_s, E_s)\)

1: initialize \(t = 0\), \(p_0 = s/|S|\)
2: \(V_s \leftarrow S\)
3: while \((t < k_0)\) do
4: \(p_{t+1} = p_t\)
5: for each \(s_j \in V_s\) do
6: \(U_j \leftarrow\) neighbors of \(s_j\)
7: \(p_{t+1}(s_i) = p_{t+1}(s_i) - p_t(s_i)\)
8: for each \(v_i \in U_j\) do
9: \(p_{t+1}(v_i) = p_{t+1}(v_i) + p_i(s_j)/|U_j|\)
10: end for
11: \(V_s = V_s \cup U_j\)
12: end for
13: \(t = t + 1\)
14: end while
15: if \(|V_s| > n_1\) then
16: \(V_s \leftarrow n_1\) nodes inside \(V_s\) with higher probability in \(p_t\)
17: end if
18: \(G_s = (V_s, E_s)\) is the induced subgraph from \(V_s\)

4.1.2. Local personalized PageRank sampling

For a fixed \(\alpha \in (0, 1)\), the personalized PageRank vector can be defined as
\[
p = (1 - \alpha) \sum_{k=0}^{\infty} \alpha^k (AD^{-1})^k p_0,
\]
where \(p_0 = s/|S|\) is the initial probability distribution on the source seeds.

In practice, we usually seek a vector \(q\) to approximate \(p\):
\[
\|D^{-1} p - D^{-1} q\|_\infty < \epsilon_1.
\]

Andersen et al. [37] propose a \(pprpush\) algorithm to guarantee (3), and use the probability values in \(q\) to identify memberships in the local community. We just adapt the advantage of \(pprpush\)'s fast speed to do the personalized PageRank sampling, shown in Algorithm 2.
Algorithm 2 The personalized PageRank sampling

Input: Graph $G = (V, E)$, seed set $S \subseteq V$, personalized PageRank diffusion parameters $\alpha$ and $\epsilon_1$, upper bound of the subgraph size $n_2$

Output: Sampled subgraph $G_s = (V_s, E_s)$

1. Start from $S$, calculate the vector $q$ to approximate the personalized PageRank diffusion vector $p$
2. Sort elements in $q$ in decreasing order to get a vector $\bar{q}$
3. $V_s \leftarrow$ nodes corresponding to all the nonzero elements in $\bar{q}$
4. if $|V_s| > n_2$ then
5. $V_s \leftarrow n_2$ nodes inside $V_s$ with higher probability in $\bar{q}$
6. end if
7. $G_s = (V_s, E_s)$ is the induced subgraph from $V_s$

4.1.3. Local heat kernel sampling

The heat kernel diffusion model spreads the heat across a graph regarding the seed set as the persistent heat source. The heat kernel diffusion vector is defined by

$$ h = e^{-t} \sum_{k=0}^{\infty} \frac{t^k}{k!} (AD^{-1})^k p_0, $$

where $p_0 = s/|S|$ is the initial heat distribution on the source seeds. For simplicity of notation, let

$$ h_l = e^{-t} \sum_{k=0}^{l} \frac{t^k}{k!} (AD^{-1})^k p_0 $$

indicate the sum of the first $l$ terms.

In practice, we usually seek a vector $x$ to approximate $h$:

$$ ||D^{-1}h - D^{-1}x||_\infty < \epsilon_2. \quad \text{(6)} $$

Premultiplying $e^T$ on both sides, we have

$$ ||D^{-1}e^T h - D^{-1}e^T x||_\infty < \epsilon_2. \quad \text{(7)} $$

If for an integer $l$,

$$ ||D^{-1}e^T h - D^{-1}e^T h_l||_\infty < \epsilon_2/2, \quad \text{(8)} $$

and $z = e^T x \approx e^T h_l$ satisfies

$$ ||D^{-1}e^T h_l - D^{-1}z||_\infty < \epsilon_2/2, \quad \text{(9)} $$

then by the triangle inequality, (7) holds, and then (6) holds.

Algorithm 3 The heat kernel sampling

Input: Graph $G = (V, E)$, seed set $S \subseteq V$, heat kernel diffusion parameters $t$ and $\epsilon_2$, upper bound of the subgraph size $n_3$

Output: Sampled subgraph $G_s = (V_s, E_s)$

1. Start from $S$, calculate heat value vector $x$ to approximate the heat kernel diffusion vector $h$
2. Sort elements in $x$ in decreasing order to get a vector $\bar{x}$
3. $V_s \leftarrow$ nodes corresponding to all the nonzero elements in $\bar{x}$
4. if $|V_s| > n_3$ then
5. $V_s \leftarrow n_3$ nodes inside $V_s$ with higher heat value in $\bar{x}$
6. end if
7. $G_s = (V_s, E_s)$ is the induced subgraph from $V_s$

Kloster and Gleich [16] propose a $hk$-relax algorithm to guarantee (8) by letting $t$ be no greater than $2t\log(1/\epsilon_2)$ and computing a vector $z$ that satisfies (9), then use the heat values in $x$ to identify memberships in the local community. We just adapt the advantage of $hk$-relax’s fast speed to do the heat kernel sampling, shown in Algorithm 3.

Denote the sampled subgraph as $G_s = (V_s, E_s)$ with $n_n$ nodes and $m_m$ edges in the following discussion. We then extract the local community from this comparatively small subgraph instead of the original large network. This pre-processing procedure runs in milliseconds in large networks with millions of nodes, and significantly reduces the computation cost for the follow-up community detection.

4.2. Locally-biased spectral approximation

In this subsection, we first provide the necessary theoretical base that finding a low-conductance community corresponds to finding the eigenvector of the transition matrix with the largest eigenvalue. Then we briefly introduce a classical power iteration and a variant of the Lanczos iteration to approximate this eigenvector, and get a “local” eigenvector indicating the implicit topology structure of the network around the seeds, and provide some convergence analysis on the power iteration and Lanczos iteration.

4.2.1. Theoretical base

Let $L = D_s - A_s$ be the unnormalized Laplacian matrix of $G_s$ where $A_s$ and $D_s$ denote the adjacency matrix and the diagonal degree matrix of $G_s$. We define two normalized graph Laplacian matrices:

$$ L_{rw} = D_s^{-1}L = I - N_{rw}, \quad \text{(10)} $$

$$ L_{sym} = D_s^{-\frac{1}{2}}LD_s^{-\frac{1}{2}} = I - N_{sym}, \quad \text{(11)} $$

where $I$ is the identity matrix, $N_{rw} = D_s^{-1}A_s$ is the transition matrix, and $N_{sym} = D_s^{-\frac{1}{2}}A_sD_s^{-\frac{1}{2}}$ is the normalized adjacency matrix.

For a community $C$, the conductance [50] of $C$ is defined as

$$ \Phi(C) = \frac{\text{cut}(C, \bar{C})}{\min(\text{vol}(C), \text{vol}(\bar{C}))}, \quad \text{(12)} $$

where $\bar{C}$ consists of all nodes outside $C$, cut($C, \bar{C}$) denotes the number of edges between, and vol($\cdot$) calculates the “edge volume”, i.e. for the subset of nodes, we count their total node degrees in graph $G_s$. Low conductance gives priority to a community with dense internal links and sparse external links.

Let $y \in \{0, 1\}^{n \times 1}$ be a binary indicator vector representing a small community $C$ in the sampled graph $G_s$. Here for “small community”, we mean $\text{vol}(C) \leq \frac{1}{4} \text{vol}(V_s)$. As $y^T D_s y$ equals the total node degrees of $C$, and $y^T A_s y$ equals two times the number of internal edges of $C$, the conductance $\Phi(C)$ could be written as a generalized Rayleigh quotient:

$$ \Phi(C) = \frac{y^T Ly}{y^T D_s y} = \frac{(D_s^2 y)^T L_{sym}(D_s^2 y)}{(D_s^2 y)^T (D_s^2 y)}. \quad \text{(13)} $$
Theorem 1. (Cheeger Inequality) Let $\lambda_2$ be the second smallest eigenvalue of $L_{sym}$ for a graph $G_s$, then $\phi(G_s) \geq \frac{\lambda_2}{2}$. where $\phi(G_s) = \min_{x \neq 0, \langle \Phi, x \rangle = 0} \frac{\langle \Phi, x \rangle^2}{\langle x, L x \rangle}$.

The proof refers to [51], and we attach the details in Appendix A. According to this theorem and the definition of $\Phi(C)$, we have $\frac{\lambda_2}{2} \leq \Phi(C) \leq 1$.

According to the Rayleigh-Ritz theorem [52], if we want to minimize the conductance $\Phi(C)$ by relaxing the indicator vector $y$ to take arbitrary real values, then the scaled relaxed indicator vector $D_s^{\frac{1}{2}} y$ should be the eigenvector of $L_{sym}$ with the smallest eigenvalue 0, which is $D_s^{\frac{1}{2}} e_s$ with $e_s$ the vector of length $n_s$ with ones as the entries.

We know that:

\[ L_{rw} v = \lambda v \iff L_{sym}(D_s^{\frac{1}{2}} v) = \lambda (D_s^{\frac{1}{2}} v), \]

where $v$ is a nonzero vector. The relaxed indicator vector $y$ should be the eigenvector of $L_{rw}$ with the smallest eigenvalue. As $L_{rw} = 1 - N_{rw}$, the eigenvalue decomposition of the Laplacian matrix is also closely related to the expansion of rapid mixing of random walks. As

\[ L_{rw} v = \lambda v \iff N_{rw} v = (1 - \lambda) v, \]

it follows that $L_{rw}$ and $N_{rw}$ share the same set of eigenvectors and the corresponding eigenvalue of $N_{rw}$ is $1 - \lambda$ where $\lambda$ is the eigenvalue of $L_{rw}$. Equivalently, the relaxed indicator vector $y$ should be the eigenvector of $N_{rw}$ with the largest eigenvalue.

The largest eigenvalue of $N_{rw}$ is 1 and the corresponding eigenvector is $e_s$ [12], so the relaxed indicator vector $y = e_s$, corresponding to the whole graph with zero conductance by Eq. (13). This relaxed indicator vector $y$ contains global information while the real solution of the indicator vector $y$ reveals local property for a small community whose total degree is no greater than half of the total degree of the whole graph. As the power iteration and the Lanczos method are efficient for computing the top eigen-pairs of large sparse matrices and they are space efficient, we propose a power iteration and a variant of Lanczos method on $N_{rw}$ to get a “local” eigenvector indicating the latent local structure around the seeds.

4.2.2. Power iteration spectral approximation

Power iteration spectral approximation. Instead of using the eigenvalue decomposition to get the global eigenvector, we consider power iteration to approximate the eigenvector of $N_{rw}$ with the largest eigenvalue. A few steps of the power iteration lead to the local approximation of this eigenvector.

By $k$ steps of power iteration, we could get an iteration vector

\[ x_k = (N_{rw})^k x_0, \]

where $x_0$ is the normalized indicator vector for the seed set. When $k \to \infty$, Theorem 2 states that the normalized iteration vector $x_k$ converges to the eigenvector of $N_{rw}$ with the largest eigenvalue and it contains global information of $G_s$. Our interest, though, is not in the limiting case when $k$ is large, but for a very small number of iteration steps to reveal the local property around the seeds.

The Power Iteration Spectral Approximation (PISA) procedure on the sampled graph is summarized in Algorithm 4. The slowest step is Step 6 for calculating $\lambda^{(k)}$. It requires $O(k n_s^2)$ time to implement the power iteration, where $k_1$ is the steps of power iteration and $n_s$ is the number of nodes in graph $G_s$.

**Algorithm 4 Power Iteration Spectral Approximation**

**Input:** $G_s = (V_s, E_s)$, maximum iteration steps $k_1$, initial vector $x_0$

**Output:** $y = y_k$

1: Initialize $k = 0$
2: while $(k < k_1)$ do
3: $k = k + 1$
4: $x_k = N_{rw} x_{k-1}$
5: $y_k = x_k / \|x_k\|_2$
6: $\lambda^{(k)} = x_k^T N_{rw} y_k$
7: end while

**Convergence analysis.** In the following discussion, we provide an analysis on the convergence of the power iteration, i.e. when $k \to \infty$, the approximation vector $y_k$ converges to the eigenvector of $N_{rw}$ with the largest eigenvalue.

**Lemma 1.** Let $G_s = (V_s, E_s)$ be a connected and non-bipartite graph with $n_s$ nodes and $m_s$ edges, $N_{rw}$ the transition matrix of $G_s$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{n_s}$, and the corresponding normalized eigenvectors are $v_1, v_2, \ldots, v_{n_s}$. Then $v_1, v_2, \ldots, v_{n_s}$ are linearly independent, and

\[ 1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{n_s} > -1, \quad v_1 = \frac{e_s}{\|e_s\|_2}, \]

where $e_s$ is a vector of length $n_s$ with ones as the entries.

**Proof.** By Eq. (15), we know that $L_{rw}$ and $N_{rw}$ share the same eigenvalues $v_1, v_2, \ldots, v_{n_s}$ and the corresponding eigenvalues of $L_{rw}$ are $\lambda_i \leq \lambda_2 \leq \ldots \leq \lambda_{n_s}$ where $\lambda_i = 1 - \lambda_i (1 \leq i \leq n_s)$.

According to Proposition 3 of [12], $L_{sym}$ and $L_{rw}$ share the $n_s$ non-negative eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n_s}$, and the corresponding eigenvectors of $L_{sym}$ are $D_s^{\frac{1}{2}} v_1, D_s^{\frac{1}{2}} v_2, \ldots, D_s^{\frac{1}{2}} v_{n_s}$.

From Theorem 8.1.1 of [42], there exists an orthogonal matrix $\tilde{V} = [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{n_s}]$ such that

\[ \tilde{V}^T L_{sym} \tilde{V} = diag(\lambda_1, \lambda_2, \ldots, \lambda_{n_s}). \]

It shows that $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{n_s}$ are linearly independent eigenvectors of $L_{sym}$, so $D_s^{\frac{1}{2}} \tilde{v}_1, D_s^{\frac{1}{2}} \tilde{v}_2, \ldots, D_s^{\frac{1}{2}} \tilde{v}_{n_s}$ are linearly independent. As $G_s$ is a connected graph, $D_s^{\frac{1}{2}}$ is invertible, and it is obvious that $v_1, v_2, \ldots, v_{n_s}$ are linearly independent.

Additionally, as $L_{rw} x_0 = 0$, we have $1 - \lambda_1 = \lambda_1 = 0$ and $v_1 = \frac{e_s}{\|e_s\|_2}$, so $\lambda_1 = 1$.

As $G_s$ is a connected and non-bipartite graph, by Lemma 1.7 of [51], we have $1 - \lambda_2 = \lambda_2 > 0$ and $1 - \lambda_{n_s} = \lambda_{n_s} < 2$. Therefore, $\lambda_2 < 1$ and $\lambda_{n_s} > -1$. 

\[ \square \]
Theorem 2. Let $G_k = (V_k, E_k)$ be a connected and non-bipartite graph with $n_i$ nodes and $m_i$ edges, when $k \to \infty$, the vector $y_k$ converges to $v_1$ and $\lambda^{(k)}$ converges to $\lambda_1$.

Proof. By Lemma 1, $v_1, v_2, \ldots, v_{n_i}$ are linearly independent normalized eigenvectors of $N_{rw}$, so there exist $a_1, a_2, \ldots, a_{n_i}$ such that $x_0 = \sum_{i=1}^{n_i} a_i v_i$.

As $v_1 = \frac{x_0}{\|x_0\|}$ and $x_0$ is the normalized indicator vector for the seed set, $v_1$ and $x_0$ are not orthogonal. It shows that $a_1$ is the nonzero projection portion of $x_0$ on the eigenvector $v_1$. Then by Eq. (16) and $(N_{rw})_i v_i = \lambda_i v_i (1 \leq i \leq n_i)$, we have

$$x_k = (N_{rw})^k x_0 = \sum_{i=1}^{n_i} a_i \lambda_i^k v_i.$$ 

Since $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{n_i} > -1$, for all $i = 2, 3, \ldots, n_i$, we have

$$\lim_{k \to \infty} \lambda_i^k = 0,$$
and

$$\lim_{k \to \infty} x_k = \lim_{k \to \infty} \frac{x_k}{\|x_k\|} = \frac{a_1 v_1}{a_1} = v_1.$$ 

As $a_1 \neq 0$, we have

$$\lim_{k \to \infty} y_k = \lim_{k \to \infty} \frac{x_k}{\|x_k\|} = \frac{a_1 v_1}{a_1} = v_1,$$
and

$$\lim_{k \to \infty} \lambda^{(k)} = \lim_{k \to \infty} y_k^T N_{rw} y_k = v_1^T N_{rw} v_1 = \lambda_1 v_1^T v_1 = \lambda_1.$$

By Eq. (20), we have

$$\|r\|_2 = \sqrt{\sum_{i=1}^{n_i} b_i^2 \lambda_i^{2k}(\sum_{i=1}^{n_i} b_i^2 \lambda_i^{2k+2}) - (\sum_{i=1}^{n_i} b_i^2 \lambda_i^{2k+1})^2}.$$

By Eq. (20), we have

$$\|r\|_2 = \sqrt{\sum_{i=1}^{n_i} b_i^2 \lambda_i^{2k}(\sum_{i=1}^{n_i} b_i^2 \lambda_i^{2k+2}) - (\sum_{i=1}^{n_i} b_i^2 \lambda_i^{2k+1})^2}.$$

By Lemma 1, $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{n_i} > -1$, we have $\lambda_2^{k+2} = \lambda_2^k = 1$ and $\lambda_3^{k+2} < \lambda_3^k < 1 (1 \leq i \leq n_i)$. Then

$$\|r\|_2 < \sqrt{\sum_{i=1}^{n_i} b_i^2 \lambda_i^{2k}}.$$

When $k \to \infty$, Theorem 2 states that the residual value $\|r\|_2 = 0$. And we just need to use a small power iteration steps to find the "local" eigenvector.

4.2.3. Lanczos iteration spectral approximation

Lanczos process. Based on a theoretical guarantee [53], there exists an orthogonal matrix $Q$ and a tridiagonal matrix $T$ such that

$$Q^T (D_s^{-\frac{1}{2}} A_s D_s^{-\frac{1}{2}}) Q = T.$$

Designate the columns of $Q$ by $Q = [q_1 \mid \cdots \mid q_{n_i}]$.

Let $\tilde{Q} = D_s^{-\frac{1}{2}} Q$, so

$$\tilde{Q} = [D_s^{-\frac{1}{2}} q_1 \mid \cdots \mid D_s^{-\frac{1}{2}} q_{n_i}] \triangleq [\tilde{q}_1 \mid \cdots \mid \tilde{q}_{n_i}].$$

Eq. (23) can be rewritten as

$$\tilde{Q}^T A_s \tilde{Q} = T.$$

As $Q$ is an orthogonal matrix,

$$\tilde{Q}^T \tilde{Q} = D_s^{-\frac{1}{2}} Q^T D_s^{-\frac{1}{2}} = D_s^{-1}.$$ 

Premultiplying $\tilde{Q}$ on both sides of Eq. (25), we have

$$D_s^{-\frac{1}{2}} A_s \tilde{Q} = \tilde{T} Q.$$ 

Equating the columns in Eq. (27), we conclude that for $k \in [1, n_i]$,

$$D_s^{-\frac{1}{2}} A_s \tilde{q}_k = \beta_{k-1} \tilde{q}_{k-1} + \alpha_k \tilde{q}_k + \beta_k \tilde{q}_{k+1}.$$
by setting $\beta_0 \tilde{q}_0 \triangleq 0$, and $\beta_k \tilde{q}_k = 0$.  
By the orthogonality of $\tilde{Q}$, we have $\tilde{Q}^T D_k \tilde{Q} = I$. Premultiplying $\tilde{Q}^T D_k$ on both sides of Eq. (28), the $D_k$-inner product orthonormality of the $\tilde{q}$-vectors implies

$$a_k = \tilde{q}_k^T A_k \tilde{q}_k. \quad (29)$$

Let the vector $\tilde{r}_k$ be

$$\tilde{r}_k = D_k^{-1} A_k \tilde{q}_k - a_k \tilde{q}_k = -\beta_k \tilde{q}_{k-1}. \quad (30)$$

If $\tilde{r}_k$ is nonzero, then by Eq. (28) we have

$$\tilde{q}_{k+1} = \tilde{r}_k / \beta_k. \quad (31)$$

With the “canonical” choice such that $\tilde{Q}^T D_k \tilde{Q} = I,$

$$\beta_k = \| D_k^{\frac{1}{2}} \tilde{r}_k \|_2. \quad (32)$$

For any unit vector $q_1$, let $\beta_0 = 1$, $\tilde{q}_0 = 0$, and $\tilde{r}_0 = D_0^{-\frac{1}{2}} q_1$. Start from $k = 1$, we could iteratively calculate the entries of $\alpha_k, \beta_k$ in $T$ until $k = n$. Meanwhile, $\tilde{Q}$ is also obtained during the iteration.

**Spectral calculation via Lanczos process.** Let $v_1$ be the eigenvector of $N_{rw}$ with the largest eigenvalue $\lambda_1$, we know

$$N_{rw} v_1 = D_1^{-\frac{1}{2}} A_1 v_1 = \lambda_1 v_1.$$ 

Premultiplying $\tilde{Q}^T D_k$ on both sides, we get

$$\tilde{Q}^T A_k \tilde{Q} v_1 = \lambda_1 \tilde{Q}^T D_k v_1, \quad (33)$$

According to Eq. (26), $\tilde{Q}^T D_k = I$. Then by Eq. (25), the left hand side of Eq. (33) equals

$$\tilde{Q}^T A_k \tilde{Q}^T D_k v_1 = T \tilde{Q}^T D_k v_1.$$ 

Let $u = \tilde{Q}^T D_k v_1$, we get

$$Tu = \lambda_1 u. \quad (34)$$

On the other hand, premultiplying $\tilde{Q}$ and postmultiplying $u$ on both sides of Eq. (25), we have

$$N_{rw} \tilde{Q} u = \lambda_1 \tilde{Q} u,$$

so $\lambda_1$ is also the largest eigenvalue of $T$. As

$$v_1 = \tilde{Q} \tilde{Q}^T D_k v_1 = \tilde{Q} u,$$

we can calculate $v_1$ by calculating the eigenvector $u$ of $T$ with the largest eigenvalue $\lambda_1$.

**Lanczos iteration spectral approximation.** Instead of using the eigenvalue decomposition to get the “global spectra”, He et al. [20] use short random walks starting from the seed set to get a local proxy for the eigenvectors of $N_{rw}$, which they call the “local spectra”. Here we consider a novel way based on the Lanczos method [47] to approximate the eigenvector of $N_{rw}$ with the largest eigenvalue. A few steps of the Lanczos iteration lead to the local approximation of this eigenvector.

Let $q_1$ be the normalized indicator vector for the seed set, set $\beta_0 = 1$, $\tilde{q}_0 = 0$, and $\tilde{r}_0 = D_0^{-\frac{1}{2}} q_1$. By $k$ steps of Lanczos iteration, we could get the first $k$ by $k$ submatrix of $T$, denoted by $T_k$. Correspondingly, let the first $k$ columns of $Q$ be a matrix $\tilde{Q}_k$. Let the eigenvectors of $T_k$ with larger eigenvalues be a matrix $U_k$. According to Eq. (35), the columns of $V_k = \tilde{Q}_k U_k$ approximate the eigenvectors of $N_{rw}$ with larger eigenvalues. The first column of $V_k$ approximates the eigenvector of $N_{rw}$ with the largest eigenvalue, which is the indicator vector $y$ we want to find.

The Lanczos Iteration Spectral Approximation (LISA) procedure on the sampled graph is summarized in Algorithm 5.

The slowest step is Step 4 for calculating $\tilde{q}_k, \alpha_k, \tilde{r}_k,$ and $\beta_k$. It requires $O(k^2 n_t^2)$ time to implement the Lanczos iteration, where $k_2$ is the steps of Lanczos iteration and $n_t$ is the number of nodes in graph $G_t$. Also, note that the Lanczos iteration requires only a few vectors of intermediate storage.

**Algorithm 5 Lanczos Iteration Spectral Approximation**

**Input:** $G_s = (V_s, E_s)$, maximum iteration steps $k_2$, initial vector $q_1$

**Output:** $y = \tilde{Q}_k u$

1. Initialize $k = 0, \beta_0 = 1, \tilde{q}_0 = 0, \tilde{r}_0 = D_0^{-\frac{1}{2}} q_1$
2. while $(k < k_2)$ do
3. $k = k + 1$
4. Calculate $\tilde{q}_k, \alpha_k, \tilde{r}_k, \beta_k$ by Eq. (31), (29), (30), (32)
5. end while
6. Let $T_k$ be the first $k \times k$ entries of $T$ in Eq. (24)
7. Get the eigenvector $u$ of $T_k$ with the largest eigenvalue $\lambda$

**Convergence analysis.** Here we provide an analysis on the convergence of the Lanczos process, i.e., the approximation gap between the local eigenvector which indicates the local structure around the seeds and the global eigenvector of the graph.

By Eq. (28) and Eq. (30), we conclude that for $1 \leq k < n$, $N_{rw} \tilde{Q}_k = D_k^{-\frac{1}{2}} A_k \tilde{Q}_k = \tilde{Q}_k T_k + \tilde{r}_k e_k^T$. \quad (36)

where $e_k$ is the $k$th unit vector with unity in the $k$th element and zero otherwise.

Let $u$ be the eigenvector of $T_k$ with the largest eigenvalue $\lambda$, postmultiplying $u$ on both sides of Eq. (36), we have

$$N_{rw} \tilde{Q}_k u = \tilde{Q}_k T_k u + \tilde{r}_k e_k^T u.$$ \quad (37)

Let $y = \tilde{Q}_k u$, the approximated residual value can be calculated as

$$\| r_2 \|_2 = \| N_{rw} y - \lambda y \|_2 = \| N_{rw} \tilde{Q}_k u - \lambda \tilde{Q}_k u \|_2.$$ \quad (38)

By Eq. (37). Eq. (38) can be modified as

$$\| r_2 \|_2 = \| N_{rw} \tilde{Q}_k u - \tilde{Q}_k T_k u \|_2 = \| \tilde{r}_k e_k^T u \|_2.$$ \quad (39)

Furthermore, by Eq. (31).

$$\| r_2 \|_2 = \| \beta_k \tilde{q}_k e_k^T u \|_2 = \| \tilde{q}_k e_k^T \|_2 \| D_k^{-\frac{1}{2}} \tilde{q}_{k+1} \|_2,$$

$$\beta_k = \| u_k \| \cdot \| D_k^{-\frac{1}{2}} \tilde{q}_{k+1} \|_2,$$ \quad (40)

where $u_k$ is the $k$th (last) term of eigenvector $u$.

As $q_{k+1}$ is a unit vector, according to the Rayleigh-Ritz theorem [52].

$$\| D_k^{-\frac{1}{2}} \tilde{q}_{k+1} \|_2 \leq \max_{x \neq 0} \| D_k^{-\frac{1}{2}} x \|_2 = d_{\min}^{-\frac{1}{2}},$$ \quad (41)

where $d_{\min}$ denotes the minimum degree of the nodes in graph $G_t$, $d_{\min}^{-\frac{1}{2}}$ is also the largest eigenvalue of the diagonal matrix $D_k^{-\frac{1}{2}}$.

By Eq. (40) and Eq. (41), we have

$$\| r_2 \|_2 \leq \beta_k d_{\min}^{-\frac{1}{2}} \| u_k \|.$$ \quad (42)

Generally, the higher the value of $k$ is, the smaller the residual value $\| r_2 \|_2$ is. Especially, when $k = n$, $N_{rw} \tilde{Q}_k = \tilde{Q}_k T_k$, we have $\| r_2 \|_2 = 0$. And we just need to use a small Lanczos iteration steps to find the “local” eigenvector.
4.3. Community boundary truncation

The value of the $k$th element of $\mathbf{y}$ indicates how likely node $k$ belongs to the target community. We use a heuristic similar to [36] to determine the community boundary.

We sort the nodes based on the element values of $\mathbf{y}$ in the decreasing order, and find a set $S_k$ with the first $k$ nodes having a comparatively low conductance. Specifically, we start from an index $k_0$ where set $S_{k_0}$ contains all the seeds. We then generate a sweep curve $\Phi(S_k)$ by increasing index $k$. Let $k^*$ be the value of $k$ where $\Phi(S_k)$ achieves a first local minimum. The set $S_{k^*}$ is regarded as the detected community.

We determine a local minima as follows. As shown in Fig. 1, if at some point $k^*$ when we are increasing $k$, $\Phi_k = \Phi(S_k)$ stops decreasing, then this $k^*$ is a candidate point for the local minimum. If $\Phi_k$ keeps increasing after $k^*$ and eventually becomes higher than $\beta \Phi_{k^*}$, then we take $k^*$ as a valid local minimum. We experimented with several values of $\beta$ on a small trial of data and found that $\beta = 1.03$ gives good performance across all the datasets.

Algorithm 6 The overall LBSA algorithm

**Input:** $G = (V,E)$, seed set $S \subseteq V$

**Output:** Community $C = S_k$

1. Get sampled subgraph $G_s = (V_s, E_s)$ by Algorithm 1, Algorithm 2, or Algorithm 3
2. Calculate vector $\mathbf{y}$ by Algorithm 4 or Algorithm 5
3. Sort nodes by the decreasing value of elements in $\mathbf{y}$
4. Find $k_0$ where $S_{k_0}$ contains all the seeds
5. For $k = k_0 : n_s$, compute the conductance $\Phi(S_k)$: $\Phi_k = \Phi(S_k = \{v| i \leq k \text{ in the sorted list}\})$
6. Find $k^*$ with the first local minimum $\Phi(S_{k^*})$

The overall Locally-Biased Spectral Approximation (LBSA) algorithm is shown in Algorithm 6. The overall LBSA algorithm includes six methods:

- rwPISA: adopt Algorithm 1 and Algorithm 4;
- rwLISA: adopt Algorithm 1 and Algorithm 5;
- prPISA: adopt Algorithm 2 and Algorithm 4;
- prLISA: adopt Algorithm 2 and Algorithm 5;
- hkPISA: adopt Algorithm 3 and Algorithm 4;
- hkLISA: adopt Algorithm 3 and Algorithm 5.

Without sampling on small network, LBSA algorithm only contains two methods: PISA (only adopt Algorithm 4) and LISA (only adopt Algorithm 5).

5. Experiments

In this section, we compare LBSA with several state-of-the-art local community detection algorithms, and evaluate the performance by popular $F_1$ score and Jaccard index.

5.1. Data description

Seven synthetic datasets (Table 3) and six real-world datasets (Table 4) are considered for a comprehensive evaluation.

5.1.1. LFR benchmark graphs

Lancichinetti et al. [54, 55] proposed a method for generating LFR benchmark graphs \(^1\) with a built-in binary community structure, which simulates properties of real-world networks on heterogeneity of node degree and community size distributions. The LFR benchmark graphs are widely used for evaluating community detection algorithms, and Xie et al. [2] performed a thorough performance comparison of different community detection algorithms on the LFR benchmark datasets.

We adopt the same set of parameter settings used in [2] and generate seven LFR benchmark graphs. Table 2 summarizes the parameter settings, among which the mixing parameter $\mu$ controls the average fraction of neighboring nodes that do not belong to any community for each node. Each node belongs to either one community or $om$ overlapping communities, and the number of nodes in overlapping communities is specified by $on$. Table 3 summarizes the statistics of the seven LFR networks and their ground-truth communities. For each of the networks, we calculate the number of nodes and edges. For the ground-truth communities, we calculate the average and standard deviation of the community size, and the average conductance.

\(^1\)https://sites.google.com/site/santofortunato/inthepress2

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 5000$</td>
<td>Number of nodes in the graph</td>
</tr>
<tr>
<td>$\mu = 0.3$</td>
<td>Mixing parameter</td>
</tr>
<tr>
<td>$d = 10$</td>
<td>Average degree of the nodes</td>
</tr>
<tr>
<td>$d_{\text{max}} = 50$</td>
<td>Maximum degree of the nodes</td>
</tr>
<tr>
<td>$[20, 100]$</td>
<td>Range of the community size</td>
</tr>
<tr>
<td>$\tau_1 = 2$</td>
<td>Node degree distribution exponent</td>
</tr>
<tr>
<td>$\tau_2 = 1$</td>
<td>Community size distribution exponent</td>
</tr>
<tr>
<td>$om \in {2, 3, ..., 8}$</td>
<td>Overlapping membership</td>
</tr>
<tr>
<td>$on = 500$</td>
<td>Number of overlapping nodes</td>
</tr>
</tbody>
</table>
5.1.2. Real-world networks

We choose six real-world network datasets with labeled ground-truth from the SNAP collection, namely Amazon, DBLP, LiveJ, YouTube, Orkut and Friendster in the domains of product, collaboration and social contact [36]. These datasets are widely used as ground-truth for evaluating community detection algorithms as they have thousands of annotated ground-truth communities. We just use the top 5000 ground-truth communities supplied by the SNAP collection for each dataset. Table 4 summarizes the statistics of the six real-world networks and their ground-truth communities.

### Table 4: Statistics for real-world networks and their ground-truth communities.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Name</th>
<th># Nodes</th>
<th># Edges</th>
<th>Ground-truth communities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Amazon</td>
<td>334,863</td>
<td>925,872</td>
<td>13 ± 18</td>
</tr>
<tr>
<td></td>
<td>DBLP</td>
<td>317,080</td>
<td>1,049,866</td>
<td>22 ± 201</td>
</tr>
<tr>
<td>Collaboration</td>
<td>LiveJ</td>
<td>3,997,962</td>
<td>34,681,189</td>
<td>28 ± 58</td>
</tr>
<tr>
<td></td>
<td>YouTube</td>
<td>1,134,890</td>
<td>2,987,624</td>
<td>21 ± 73</td>
</tr>
<tr>
<td>Social</td>
<td>Orkut</td>
<td>3,072,441</td>
<td>117,185,083</td>
<td>216 ± 321</td>
</tr>
<tr>
<td></td>
<td>Friendster</td>
<td>65,608,366</td>
<td>1,806,067,135</td>
<td>47 ± 70</td>
</tr>
</tbody>
</table>

5.2. Experimental setup

We implement the proposed LBSA method in Matlab through a C mex interface and conduct experiments on a computer with 2 Intel Xeon processors at 2.30GHz and 256GB memory. For the six SNAP datasets, we randomly locate 500 ground-truth communities on each dataset, and randomly pick three exemplary seeds from each target community. For the seven LFR datasets, we deal with every ground-truth community and randomly pick three exemplary seeds from each ground-truth community. We just pre-process all real-world datasets by sampling, and apply the LBSA methods for each network. To make a fair comparison, we run all baseline algorithms using the same set of random seeds.

5.3. Experimental results

5.3.1. Statistics on sampling

Table 5, Table 6 and Table 7 show the statistics for the random walk, personalized PageRank and heat kernel sampling on the real-world datasets, respectively. As shown in Table 5, Table 6 and Table 7, the sampled subgraphs are relatively small with 3155, 3424 and 3503 nodes on average, only sampled about 0.18%, 0.45% and 0.29% of the nodes from the original

For the parameters, we fix \((k_0, n_1) = (3, 5000)\) for Algorithm 1, \((\alpha, \epsilon_1, n_2) = (0.9, 10^{-6}, 5000)\) for Algorithm 2, and \((t, \epsilon_2, n_3) = (3, 10^{-6}, 5000)\) for Algorithm 3 such that the resulting subgraph is large enough to cover almost all the members in the target community. We set \(k_1 = 6\) for Algorithm 4 and \(k_2 = 4\) for Algorithm 5 to have a good trade-off on real-world datasets as well as the synthetic data.

5.2. Experimental setup

We implement the proposed LBSA method in Matlab through a C mex interface and conduct experiments on a computer with 2 Intel Xeon processors at 2.30GHz and 256GB memory. For the six SNAP datasets, we randomly locate 500 ground-truth communities on each dataset, and randomly pick three exemplary seeds from each target community. For the seven LFR datasets, we deal with every ground-truth community and randomly pick three exemplary seeds from each ground-truth community. We just pre-process all real-world datasets by sampling, and apply the LBSA methods for each network. To make a fair comparison, we run all baseline algorithms using the same set of random seeds.

5.2. Experimental setup

We implement the proposed LBSA method in Matlab through a C mex interface and conduct experiments on a computer with 2 Intel Xeon processors at 2.30GHz and 256GB memory. For the six SNAP datasets, we randomly locate 500 ground-truth communities on each dataset, and randomly pick three exemplary seeds from each target community. For the seven LFR datasets, we deal with every ground-truth community and randomly pick three exemplary seeds from each ground-truth community. We just pre-process all real-world datasets by sampling, and apply the LBSA methods for each network. To make a fair comparison, we run all baseline algorithms using the same set of random seeds.

### Table 3: Statistics for LFR networks and their ground-truth communities.

<table>
<thead>
<tr>
<th>Networks</th>
<th>Name</th>
<th># Nodes</th>
<th># Edges</th>
<th>Avg. ± Std. Size</th>
<th>Avg. Cond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFR ((\alpha \omega = 2))</td>
<td>5,000</td>
<td>25,527</td>
<td>48 ± 21</td>
<td>0.363</td>
<td></td>
</tr>
<tr>
<td>LFR ((\alpha \omega = 3))</td>
<td>5,000</td>
<td>25,494</td>
<td>52 ± 24</td>
<td>0.421</td>
<td></td>
</tr>
<tr>
<td>LFR ((\alpha \omega = 4))</td>
<td>5,000</td>
<td>25,241</td>
<td>48 ± 22</td>
<td>0.461</td>
<td></td>
</tr>
<tr>
<td>LFR ((\alpha \omega = 5))</td>
<td>5,000</td>
<td>25,511</td>
<td>51 ± 23</td>
<td>0.512</td>
<td></td>
</tr>
<tr>
<td>LFR ((\alpha \omega = 6))</td>
<td>5,000</td>
<td>25,657</td>
<td>47 ± 23</td>
<td>0.542</td>
<td></td>
</tr>
<tr>
<td>LFR ((\alpha \omega = 7))</td>
<td>5,000</td>
<td>25,699</td>
<td>49 ± 25</td>
<td>0.570</td>
<td></td>
</tr>
<tr>
<td>LFR ((\alpha \omega = 8))</td>
<td>5,000</td>
<td>25,086</td>
<td>49 ± 23</td>
<td>0.609</td>
<td></td>
</tr>
</tbody>
</table>

5.3. Experimental results

5.3.1. Statistics on sampling

Table 5, Table 6 and Table 7 show the statistics for the random walk, personalized PageRank and heat kernel sampling on the real-world datasets, respectively. As shown in Table 5, Table 6 and Table 7, the sampled subgraphs are relatively small with 3155, 3424 and 3503 nodes on average, only sampled about 0.18%, 0.45% and 0.29% of the nodes from the original
Nevertheless, there are very high coverage ratio (ratio of ground-truth nodes covered by the subgraph) of 97.0%, 96.9% and 97.7%, and the sampling procedure are very fast in 1.301, 0.051 and 0.309 seconds. Additionally, Table 5 shows that the sampled subgraph is very small with only 41 nodes on Amazon but with very high coverage ratio of 99.1%. On the other hand, the random walk sampling procedure on the Friendster is longer than that of the other two, as shown in Table 6 and Table 7. In summary, the three methods all work well for sampling; heat kernel is slightly better among the three methods with higher average coverage ratio and short sampling time.

Table 5
Statistics of the average values for the random walk sampling.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Coverage</th>
<th>$n_r$</th>
<th>$n_r/n$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.991</td>
<td>41</td>
<td>$1.2 \times 10^{-4}$</td>
<td>0.002</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.987</td>
<td>1347</td>
<td>$4.2 \times 10^{-3}$</td>
<td>0.002</td>
</tr>
<tr>
<td>LiveJ</td>
<td>0.999</td>
<td>2780</td>
<td>$7.0 \times 10^{-4}$</td>
<td>0.045</td>
</tr>
<tr>
<td>YouTube</td>
<td>0.935</td>
<td>4801</td>
<td>$4.2 \times 10^{-3}$</td>
<td>0.137</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.914</td>
<td>4985</td>
<td>$1.6 \times 10^{-3}$</td>
<td>0.542</td>
</tr>
<tr>
<td>Friendster</td>
<td>0.994</td>
<td>4977</td>
<td>$7.6 \times 10^{-5}$</td>
<td>0.076</td>
</tr>
<tr>
<td>Average</td>
<td>0.970</td>
<td>3155</td>
<td>$1.8 \times 10^{-4}$</td>
<td>1.301</td>
</tr>
</tbody>
</table>

Table 6
Statistics of the average values for the personalized PageRank sampling.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Coverage</th>
<th>$n_r$</th>
<th>$n_r/n$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>1.000</td>
<td>2054</td>
<td>$6.1 \times 10^{-4}$</td>
<td>0.014</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.992</td>
<td>4637</td>
<td>$1.5 \times 10^{-2}$</td>
<td>0.051</td>
</tr>
<tr>
<td>LiveJ</td>
<td>0.999</td>
<td>2661</td>
<td>$6.7 \times 10^{-4}$</td>
<td>0.036</td>
</tr>
<tr>
<td>YouTube</td>
<td>0.966</td>
<td>4973</td>
<td>$4.4 \times 10^{-3}$</td>
<td>0.081</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.882</td>
<td>3493</td>
<td>$1.1 \times 10^{-3}$</td>
<td>0.060</td>
</tr>
<tr>
<td>Friendster</td>
<td>0.977</td>
<td>2727</td>
<td>$4.2 \times 10^{-5}$</td>
<td>0.066</td>
</tr>
<tr>
<td>Average</td>
<td>0.969</td>
<td>3424</td>
<td>$4.5 \times 10^{-4}$</td>
<td>0.051</td>
</tr>
</tbody>
</table>

Table 7
Statistics of the average values for the heat kernel sampling.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Coverage</th>
<th>$n_r$</th>
<th>$n_r/n$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.999</td>
<td>449</td>
<td>$1.3 \times 10^{-4}$</td>
<td>0.002</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.991</td>
<td>3034</td>
<td>$9.6 \times 10^{-3}$</td>
<td>0.028</td>
</tr>
<tr>
<td>LiveJ</td>
<td>0.999</td>
<td>2639</td>
<td>$6.6 \times 10^{-4}$</td>
<td>0.076</td>
</tr>
<tr>
<td>YouTube</td>
<td>0.969</td>
<td>4949</td>
<td>$4.4 \times 10^{-3}$</td>
<td>0.447</td>
</tr>
<tr>
<td>Orkut</td>
<td>0.908</td>
<td>4990</td>
<td>$1.6 \times 10^{-3}$</td>
<td>0.511</td>
</tr>
<tr>
<td>Friendster</td>
<td>0.995</td>
<td>4954</td>
<td>$7.6 \times 10^{-5}$</td>
<td>0.789</td>
</tr>
<tr>
<td>Average</td>
<td>0.977</td>
<td>3503</td>
<td>$2.9 \times 10^{-5}$</td>
<td>0.309</td>
</tr>
</tbody>
</table>

5.3.2. Convergence analysis

As LBSA algorithm involves the power iteration and Lanczos iteration, we experimentally investigate the convergence property of Algorithm 4 and Algorithm 5 on LFR benchmark graphs as well as real-world networks. For the output of each iteration, we calculate the residual value $||r_1||_2$ for Algorithm 4 and $||r_2||_2$ for Algorithm 5.

Fig. 2 shows the convergence analysis on LFR networks with power iteration and Lanczos iteration. One can see that the Lanczos iteration converges more quickly than the power iteration, as shown in Fig. 2. Specifically, Fig. 2 (a) indicates that the residual value $||r_1||_2$ converges to zero for more than 20 iterations. However, Fig. 2 (b) illustrates that the residual value $||r_2||_2$ converges to zero only for more than 10 iterations.

Fig. 3 shows the convergence analysis on real-world networks with power iteration and Lanczos iteration. Overall, as shown in Fig. 3, the Lanczos iteration runs more stably and converges more quickly than the power iteration. Specifically, Fig. 3 (b), Fig. 3 (d) and Fig. 3 (f) show that the residual values $||r_2||_2$ converge to zero only for more than 20 iterations. However, Fig. 3 (a), Fig. 3 (c) and Fig. 3 (e) indicate that the residual values $||r_1||_2$ converge to zero for more than 40 iterations. Additionally, the power iteration runs unstably on YouTube, Orkut and Friendster, as shown in Fig. 3 (a), Fig. 3 (c) and Fig. 3 (e).

As the Lanczos iteration converges more quickly than the power iteration on real-world networks as well as LFR datasets, we set the iteration steps $k_1 = 6$ for Algorithm 4 and $k_2 = 4$ for Algorithm 5 to gain a “local” eigenvector indicating the implicit topology structure of the local region around the seeds.

5.3.3. Accuracy comparison on LFR datasets

Fig. 4 illustrates the average detection accuracy of our methods and the baselines on the LFR datasets. As shown in Fig. 4 (a) and Fig. 4 (b), PISA and LISA significantly outperform all baseline methods evaluated by $F_1$ score and Jaccard index on all the seven synthetic networks, and PISA achieves the best accuracy but HK and PR are the last two place on all LFR datasets. Additionally, one can see that all algorithms tend to achieve a lower accuracy with a larger overlapping membership on $om$. As on $om = 500$ overlapping nodes are assigned to on $om \in \{2, 3, \ldots, 8\}$ communities, a larger $om$ makes the detection more difficult, this may explain why all algorithms tend to achieve a lower accuracy with a larger $om$.

Fig. 5 and Table 8 show more comparisons on LFR datasets. Fig. 5 illustrates the average conductance of the detected communities on LFR datasets. One can see that HK and PR tend to achieve the lower conductance, and PISA and LISA get the most consistent results as compared with conductance of ground-truth communities shown in Table 3. However, as compared with ground-truth communities shown in Table 3, Table 8 shows that PISA uncovers more suitable size of the detected communities compared to LISA, and HK and PR attain much large size of the detected communities. These comparison analysis may explain why PISA and LISA significantly outperform all baseline methods, and PISA achieves the best accuracy but HK and PR perform worse.

For the running time comparison, as the graph loading time is the same for all algorithms, we omit the loading time for our algorithms and all baselines to show their differences on the actual processing. Table 8 shows that all algorithms are very fast and run in seconds. On average, HK is the fastest in 0.025 seconds, and the other five run in 0.1 to 5.1 seconds. PISA and LISA costs several more seconds as compared with other algorithms, as they involve finding a community with the local minimal conductance. Also, different methods are implemented in

10
Fig. 2. Convergence analysis on LFR networks with power iteration and Lanczos iteration.

Fig. 3. Convergence analysis on real-world networks with power iteration and Lanczos iteration.
overlapping membership
0.0
0.2
0.4
0.6
0.8
1.0
F1 score
PISA
LISA
LEMON
LOSP
HK
PR
(a)

overlapping membership
0.0
0.2
0.4
0.6
0.8
1.0
Jaccard index
PISA
LISA
LEMON
LOSP
HK
PR
(b)

Fig. 4. Accuracy comparison on LFR datasets.

Fig. 5. Conductance comparison on LFR datasets.

Table 8
Average size of the detected communities and average running time of the algorithms on LFR datasets.

<table>
<thead>
<tr>
<th>Size</th>
<th>PISA</th>
<th>LISA</th>
<th>LEMON</th>
<th>LOSP</th>
<th>HK</th>
<th>PR</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PISA</td>
</tr>
<tr>
<td>LFR (om = 2)</td>
<td>42</td>
<td>232</td>
<td>16</td>
<td>11</td>
<td>2258</td>
<td>2344</td>
<td>4.637</td>
</tr>
<tr>
<td>LFR (om = 3)</td>
<td>41</td>
<td>269</td>
<td>14</td>
<td>12</td>
<td>2368</td>
<td>2340</td>
<td>5.040</td>
</tr>
<tr>
<td>LFR (om = 4)</td>
<td>144</td>
<td>699</td>
<td>16</td>
<td>12</td>
<td>2267</td>
<td>2343</td>
<td>5.314</td>
</tr>
<tr>
<td>LFR (om = 5)</td>
<td>110</td>
<td>597</td>
<td>18</td>
<td>11</td>
<td>2354</td>
<td>2334</td>
<td>5.193</td>
</tr>
<tr>
<td>LFR (om = 6)</td>
<td>68</td>
<td>926</td>
<td>17</td>
<td>11</td>
<td>2374</td>
<td>2323</td>
<td>5.401</td>
</tr>
<tr>
<td>LFR (om = 7)</td>
<td>186</td>
<td>1137</td>
<td>17</td>
<td>11</td>
<td>2417</td>
<td>2326</td>
<td>5.289</td>
</tr>
<tr>
<td>LFR (om = 8)</td>
<td>229</td>
<td>1305</td>
<td>16</td>
<td>12</td>
<td>2374</td>
<td>2330</td>
<td>5.119</td>
</tr>
<tr>
<td>Average</td>
<td>117</td>
<td>738</td>
<td>16</td>
<td>11</td>
<td>2344</td>
<td>2334</td>
<td>5.142</td>
</tr>
</tbody>
</table>
Table 9
Average size of the detected communities of the algorithms on real-world networks.

<table>
<thead>
<tr>
<th></th>
<th>rwPISA</th>
<th>rwLISA</th>
<th>prPISA</th>
<th>prLISA</th>
<th>hkPISA</th>
<th>hkLISA</th>
<th>LEMON</th>
<th>LOSP</th>
<th>HK</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>9</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>13</td>
<td>8</td>
<td>18</td>
<td>512</td>
</tr>
<tr>
<td>DBLP</td>
<td>42</td>
<td>7</td>
<td>19</td>
<td>7</td>
<td>19</td>
<td>7</td>
<td>20</td>
<td>20</td>
<td>87</td>
<td>9077</td>
</tr>
<tr>
<td>LiveJ</td>
<td>258</td>
<td>98</td>
<td>54</td>
<td>31</td>
<td>73</td>
<td>31</td>
<td>26</td>
<td>28</td>
<td>119</td>
<td>512</td>
</tr>
<tr>
<td>YouTube</td>
<td>1875</td>
<td>113</td>
<td>658</td>
<td>290</td>
<td>750</td>
<td>309</td>
<td>18</td>
<td>17</td>
<td>122</td>
<td>13840</td>
</tr>
<tr>
<td>Orkut</td>
<td>2430</td>
<td>744</td>
<td>618</td>
<td>236</td>
<td>1319</td>
<td>444</td>
<td>9</td>
<td>17</td>
<td>341</td>
<td>1648</td>
</tr>
<tr>
<td>Friendster</td>
<td>2225</td>
<td>1174</td>
<td>409</td>
<td>161</td>
<td>1035</td>
<td>553</td>
<td>9</td>
<td>13</td>
<td>107</td>
<td>752</td>
</tr>
<tr>
<td>Average</td>
<td>1140</td>
<td>357</td>
<td>295</td>
<td>122</td>
<td>535</td>
<td>225</td>
<td>16</td>
<td>16</td>
<td>137</td>
<td>5052</td>
</tr>
</tbody>
</table>

Table 10
Average running time measured by seconds of the algorithms on real-world networks.

<table>
<thead>
<tr>
<th></th>
<th>rwPISA</th>
<th>rwLISA</th>
<th>prPISA</th>
<th>prLISA</th>
<th>hkPISA</th>
<th>hkLISA</th>
<th>LEMON</th>
<th>LOSP</th>
<th>HK</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.002</td>
<td>0.002</td>
<td>0.665</td>
<td>0.496</td>
<td>0.051</td>
<td>0.039</td>
<td>0.039</td>
<td>0.117</td>
<td>0.008</td>
<td>0.008</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.547</td>
<td>0.427</td>
<td>2.556</td>
<td>1.848</td>
<td>1.366</td>
<td>0.990</td>
<td>0.271</td>
<td>0.127</td>
<td>0.026</td>
<td>0.581</td>
</tr>
<tr>
<td>LiveJ</td>
<td>3.654</td>
<td>2.430</td>
<td>1.873</td>
<td>1.167</td>
<td>2.009</td>
<td>1.257</td>
<td>0.553</td>
<td>1.092</td>
<td>0.018</td>
<td>0.279</td>
</tr>
<tr>
<td>YouTube</td>
<td>9.445</td>
<td>7.218</td>
<td>4.098</td>
<td>2.549</td>
<td>6.313</td>
<td>3.742</td>
<td>1.458</td>
<td>0.812</td>
<td>0.035</td>
<td>0.241</td>
</tr>
<tr>
<td>Orkut</td>
<td>13.566</td>
<td>8.651</td>
<td>4.132</td>
<td>2.097</td>
<td>9.913</td>
<td>5.138</td>
<td>1.328</td>
<td>1.960</td>
<td>0.025</td>
<td>0.091</td>
</tr>
<tr>
<td>Friendster</td>
<td>14.693</td>
<td>10.308</td>
<td>5.903</td>
<td>4.681</td>
<td>10.194</td>
<td>4.410</td>
<td>1.649</td>
<td>28.980</td>
<td>0.116</td>
<td>0.098</td>
</tr>
<tr>
<td>Average</td>
<td>6.985</td>
<td>4.839</td>
<td>3.205</td>
<td>2.140</td>
<td>4.974</td>
<td>2.596</td>
<td>0.883</td>
<td>5.515</td>
<td>0.038</td>
<td>0.348</td>
</tr>
</tbody>
</table>

Fig. 6. Accuracy comparison on real-world networks.

Fig. 7. Conductance comparison on real-world networks.
different languages, so the running time could give an indication of the overall trend, and it can not be compared directly.

5.3.4. Accuracy comparison on real-world networks

Fig. 6 illustrates the average detection accuracy on real-world networks. As shown in Fig. 6 (a) and Fig. 6 (b), prLISA and hkLISA outperform all baseline methods evaluated by $F_1$ score and Jaccard index on all six real-world networks. rwPISA, prPISA and hkPISA outperform all baseline methods on Amazon and LiveJ, and rwLISA outperforms all baseline methods on YouTube. On the other hand, LEMON shows the lowest accuracy on LiveJ and Orkut, HK shows the worst performance on YouTube and Friendster, and PR is in the last place on Amazon and DBLP. Though rwPISA, rwLISA, prPISA and hkPISA do not outperform all baseline methods on all six real-world networks, they are the most robust methods and very competitive on average. As a whole, prLISA and hkLISA perform the better on the six real-world datasets while HK and PR are in the last two place.

Fig. 7 and Table 9 show more comparisons on real-world datasets. As shown in Fig. 7, LEMON obtains the highest conductance on LiveJ, which is much bigger than the ground-truth as shown in Table 4. Table 9 shows that LEMON achieves much small size of the detected communities on Orkut compared with ground-truth communities, as shown in Table 4. This may explain why HK achieves the lowest accuracy on YouTube and Friendster, as shown in Table 7. This may explain why HK achieves the lowest conductance on YouTube and Friendster, which are very small as compared with ground-truth communities shown in Table 4. This may explain why HK achieves the lowest accuracy on YouTube and Friendster. Table 9 also illustrates that PR attains much large size of the detected communities on Amazon and DBLP as compared with ground-truth communities shown in Table 4. This may explain why PR is in the last place on Amazon and Friendster.

For the running time comparison, as the graph loading time is the same for all algorithms, we omit the loading time of our algorithms and all baselines to show their differences on the actual processing. Table 10 illustrates that all algorithms are very fast and run in seconds. On average, HK is the fastest in 0.038 seconds, and the other nine algorithms run in 0.3 to 6.9 seconds. Our methods costs several more seconds as compared with HK and PR, as they involve finding a community with the local minimal conductance. Also, different methods are implemented in different languages, so the running time could give an indication of the overall trend, and it can not be compared directly.

6. Conclusions and future works

In this paper, we propose a Locally-Biased Spectral Approximation (LBSA) approach for local community detection. Firstly, we utilize the very fast random walk, personalized PageRank and heat kernel diffusion to get a local sampled subgraph that largely reduces the complexity of the subsequent computation. Based on Rayleigh quotient and conductance, we provide theoretical base for the proposed method. In addition, we adopt power iteration and Lanczos iteration to get a “local” eigenvector. To the best of our knowledge, this is the first time to apply Lanczos method for local community detection. The convergence analysis results show that Lanczos iteration is more robustly and converges more quickly than the power iteration. Extensive experimental results on the real-world networks as well as synthetic LFR datasets also show that the LBSA algorithm could be a new and effective way to detect local communities in large graphs. We wish our work inspire more researches based on the locally-biased spectral approximation method for network analysis and community detection.

Acknowledgements

This work was supported by National Natural Science Foundation of China (61772219, 61472147, 61572221).

Appendix A. The proof of Theorem 1

Before given the proof of Theorem 1, we first prove the following proposition.

Proposition 1. Let $\lambda_2$ be the second smallest eigenvalue of $L_{sym}$ for a graph $G_s$, then

$$\lambda_2 = \min_{x \neq 0} \frac{x^T L_{sym} x}{x^T x},$$

where $D_s$ denote the diagonal degree matrix of $G_s$, and $e_s$ the vector of length $n_s$ with ones as the entries.

Proof. As $L_{sym}$ is a symmetric matrix, let $L_{sym} = V \hat{\Lambda} V^T$ be the eigenvalue decomposition, where $V = [v_1, v_2, ..., v_n]$ is an orthogonal matrix and $\hat{\Lambda} = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$, $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. Then

$$x^T L_{sym} x = \frac{x^T (V \hat{\Lambda} V^T) x}{x^T x} = \frac{(V^T x)^T \hat{\Lambda} (V^T x)}{(V^T x)^T (V^T x)}.$$  \hspace{1cm} (A.2)

Let $w = V^T x$, we have

$$\frac{x^T L_{sym} x}{x^T x} = \frac{w^T \hat{\Lambda} w}{w^T w} = \frac{\sum_{i=1}^{n} \lambda_i w_i^2}{\sum_{i=1}^{n} w_i^2},$$  \hspace{1cm} (A.3)

where $w_i$ denotes the $i$th element of the vector $w$.

As $L_{sym}(D_{sym}^\frac{1}{2} v) = 0$, we have $\tilde{v}_1 = \frac{D_{sym}^{-\frac{1}{2}} v_1}{||D_{sym}^{-\frac{1}{2}} v_1||}$. Let $x \perp D_{sym}^\frac{1}{2} e_s$, $w_1 = \tilde{V}^T x = 0$. Then by Eq. (A.3),

$$\min_{x \neq 0} \frac{x^T L_{sym} x}{x^T x} = \frac{\sum_{i=2}^{n} \lambda_i w_i^2}{\sum_{i=2}^{n} w_i^2} \geq \frac{\sum_{i=2}^{n} \lambda_i w_i^2}{\sum_{i=2}^{n} w_i^2} = \lambda_2.$$  \hspace{1cm} (A.4)

Additionally, as $\tilde{v}_1^T \tilde{v}_2 = 0$, we have $\tilde{v}_2 \perp D_{sym}^\frac{1}{2} e_s$. By $L_{sym} \tilde{v}_2 = \lambda_2 \tilde{v}_2$ and $||\tilde{v}_2|| = 1$,

$$\lambda_2 = \frac{\tilde{v}_1^T L_{sym} \tilde{v}_2}{\tilde{v}_1^T \tilde{v}_2} \geq \min_{x \neq 0} \frac{x^T L_{sym} x}{x^T x}.$$  \hspace{1cm} (A.5)
By Eq. (A.4) and Eq. (A.5), we have
\[ \lambda_i = \min_{x \in \mathbb{R}^n} \frac{x^TI_{1,xy}x}{x^Tf}. \]  
(A.6)

**Proof of Theorem 1.** By Proposition 1, we have
\[ \lambda_i = \min_{x \in \mathbb{R}^n} \frac{x^TI_{1,xy}x}{x^Tf} = \min_{x \in \mathbb{R}^n} \frac{\sum_{j=1}^{\infty} (z_i - z_j)^2}{\sum_j z_j^2}. \]  
(A.7)

where \( z = D_x^{-1}d_x \), \( d_x \) is the degree of the \( i \)th node and \( \sum_j \) denotes the sum over all unordered pairs \( \{i, j\} \) for which \( i \) and \( j \) are adjacent.

We choose an optimum subset \( \hat{V} \subseteq V \), which achieves \( \phi(G_\phi) \) and separates the graph \( G_\phi \) into two parts, \( \hat{V} \) and \( \bar{V} \). Let
\[ z_i = \begin{cases} 
\frac{1}{\text{vol}(V)} & \text{if the } i \text{th node is in } \hat{V}, \\
\frac{1}{\text{vol}(\bar{V})} & \text{if the } i \text{th node is in } \bar{V}.
\end{cases} \]  
(A.8)

Here \( \text{vol}(\hat{V}) \) indicate the total degrees of nodes inside \( \hat{V} \) in graph \( G_\phi \).

As \( z \perp D_x e_s \), by Eq. (A.7) and Eq. (A.8), we have the following:
\[ \lambda_2 \leq \text{cut}(\hat{V}, \bar{V}) (1/\text{vol}(\hat{V}) + 1/\text{vol}(\bar{V})) \leq \frac{2\text{cut}(\hat{V}, \bar{V})}{\min(\text{vol}(\hat{V}), \text{vol}(\bar{V}))} = \frac{2\phi(G_\phi)}{2}, \]  
(A.9)

where \( \text{cut}(\hat{V}, \bar{V}) \) denotes the number of edges between \( \hat{V} \) and \( \bar{V} \).

From Eq. (A.9), we have
\[ \phi(G_\phi) \geq \frac{\lambda_2}{2}. \]  
(A.10)

**References**

[40] F. Chung, O. Simpson, Solving linear systems with boundary conditions


