Uncovering the Small Community Structure in Large Networks: A Local Spectral Approach

Yixuan Li
Cornell Unversity
yli@cs.cornell.edu

Kun He *
Huazhong University of Science and Technology
brooklet60@hust.edu.cn

David Bindel
Cornell University
bindel@cs.cornell.edu

John E. Hopcroft
Cornell University
jeh@cs.cornell.edu

ABSTRACT

Large graphs arise in a number of contexts and understanding their structure and extracting information from them is an important research area. Early algorithms on mining communities have focused on the global structure, and often run in time functional to the size of the entire graph. Nowadays, as we often explore networks with billions of vertices and find communities of size hundreds, it is crucial to shift our attention from macroscopic structure to microscopic structure when dealing with large networks. A growing body of work has been adopting local expansion methods in order to identify the community from a few exemplary seed members.

In this paper, we propose a novel approach for finding overlapping communities called *LEMON* (Local Expansion via Minimum One Norm). Different from PageRank-like diffusion methods, *LEMON* finds the community by seeking a sparse vector in the span of the local spectra such that the seeds are in its support. We show that *LEMON* can achieve the highest detection accuracy among state-of-the-art proposals. The running time depends on the size of the community rather than that of the entire graph. The algorithm is easy to implement, and is highly parallelizable.

Moreover, given that networks are not all similar in nature, a comprehensive analysis on how the local expansion approach is suited for uncovering communities in different networks is still lacking. We thoroughly evaluate our approach using both synthetic and real-world datasets across different domains, and analyze the empirical variations when applying our method to inherently different networks in practice. In addition, the heuristics on how the quality and quantity of the seed set would affect the performance are provided.

*Corresponding author.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). IW3C2 reserves the right to provide a hyperlink to the author's site if the Material is used in electronic media.
WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741676.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Clustering; G.2.2 [**Discrete Mathematics**]: Graph Theory - Graph Algorithms

General Terms

Algorithms

Keywords

Community detection; Local spectral clustering; Seed set expansion; Random walk; Minimum one norm

1. INTRODUCTION

Analyzing the structure and extracting information from complex networks is an important research area. Significant research has been carried out in finding the structure of networks and identifying communities [10].

In early work, researchers assumed that communities were disjoint and had more internal connections than external connections. Both assumptions have been discarded since it is clear that in most networks a vertex belongs to more than one community. For instance, in social networks, one might belong to a work community, a community of friends, and a community of individuals that share the same hobby such as golf; in co-purchased networks, one item might belong to multiple categories. Also since we are dealing with networks with hundreds of millions of vertices, an individual in a community of size 100[1] will certainly have more links outside the community than inside. These key insights have motivated researchers to identify communities from a new perspective.

Considerable researches on detecting communities have focused on the global structure. And these globally based detection algorithms usually run in time functional to the size of the entire graph, a major drawback in computational cost. Nowadays, we explore networks with billions of vertices to find communities of size a hundred. Thus, taking the entire graph into account might not serve as a practical solution in many situations. It is crucial to shift our attention from

[1]A statistical study on social networks done by Leskovec *et al.* [17] has shown that real-world communities with high quality are quite small and usually consist of no more than 100 vertices.

macroscopic structure to microscopic structure when dealing with large networks, and develop new approaches that enable finding communities in time functional to the size of the community.

Quite recently, there has been a growing interest in finding communities by locally expanding an exemplary seed set in the community of interest [6][14][25][27]. This type of algorithm usually starts with a few members that are already known to be in the target community, and the goal is to uncover the remaining members in the same community as the exemplary members. These known members are usually referred to as *seeds* in the literature, and the process of gradually growing the seed set into a larger set until the target community is revealed is called *seed set expansion*. The setting of seed set expansion can be widely applied to many real world applications. For example, in web search, with a few known pages that share similar information, we could generate a larger group of web pages that contain the relevant contents with respect to a certain search query; in product networks, seed set expansion enables the automatic categorization of products that are discovered to be in the same community as the labeled items.

The random walk technique has been extensively adopted as a subroutine for locally growing the seed set in the literature [6][12][14][21][23][25][27]. The dynamics of random walks are effective in finding a local community since they make non-uniform expansion decisions based on the structure revealed during the exploration of the neighborhood surrounding the seeds [6]. This implies that random walk based local expansion is able to trace the community members in a principled way that resembles the natural process of forming the local community structure. Very recently, Abrahao *et al.* [3] also experimentally verified that random walk produces communities that are most structurally similar to real-world communities amongst various algorithmic communities.

In this paper we propose a novel approach for finding overlapping communities called *LEMON* (Local Expansion via Minimum One Norm)[2] for finding overlapping communities in large networks. We systematically demonstrate that *LEMON* can achieve both high efficiency and effectiveness that significantly stand out amongst state-of-the-art proposals. Specifically, we consider the span of a few dimensions of vectors after the short random walk and use it as the approximate invariant subspace, which we refer as *local spectra*. What makes LEMON distinct from previous PageRank-like diffusion methods is by utilizing the subspace rather than a single probability vector. In contrast to the traditional spectral clustering methods, the local spectral method does not require the burdensome computation of a large number of singular vectors. In addition, as traditional spectral methods usually partition the vertices into disjoint communities, we make another fundamental change. Concretely, we mine the communities from the subspace by seeking a sparse approximate indicator vector in the span of the local spectral such that the seeds are in its support. In practice, this can be mathematically achieved by solving an ℓ^1-penalized linear programming problem.

We aim to develop a comprehensive understanding of the local spectral approach for identifying a community from a small seed set. Following the central idea of our approach, we seek to solve fundamentally important questions such as: what defines "good" communities and when do they emerge as we expand the seed set (Section 4.4)? How to find a small community in time functional to the size of the community rather than that of the entire graph (Section 4.3)? What defines "good" seeds and how many seeds could uniquely define a community (Section 5)? And given that networks are not all similar in nature, how the local expansion approach is suited for uncovering communities in different types of networks (Section 6.4)?

We thoroughly evaluate our approach using both synthetic and real-world datasets across different domains, and analyze the empirical variations when applying our method to inherently different networks in practice. We believe that the insights we gained from researching on these problems would provide valuable guidance for future investigation on this topic.

[2]Our demo code is publicly available at: `https://github.com/yixuanli/lemon`.

2. RELATED WORK

A considerable amount of literature has been published on finding communities in large social and information networks. We highlight a few ideas that have recently emerged in the literature to clarify how our method differs.

Globally based community finding algorithms. Various community detection algorithms have been developed in the past decade. And most of the algorithms fall into the category of global approach. One stream of global algorithms attempt to find communities by optimizing an objective function. For example, OSLOM [16] is based on the optimization of a fitness function, which expresses the statistical significance of clusters with respect to random fluctuations (i.e., the random graph generated by the configuration model [19] during community expansion). However, the communities identified by mathematical construction may structurally diverge from real communities as pointed out in [3]. Another main stream of research adopts the label propagation approach [22], which defines rules that simulate the spread of labels of vertices in the network. The DEMON algorithm [9], for example, democratically lets each vertex vote for the communities it sees surrounding it in its limited view of the global system using a label propagation algorithm, and then merges the local communities into a global collection. Other approaches such as Link Community (LC) [4] partitions the graph by first building a hierarchical link dendrogram according to the link similarity and then cutting the dendrogram at some threshold to yield link communities.

Random walk based detection algorithms. As noted in the preceding section, among the divergent approaches, random walks tend to reveal communities that bear the closest resemblance to the ground truth communities [3]. In the following, we briefly review some methods that have adopted the random walk technique in finding communities. Speaking of methods that focus on the global structure, Pons *et al.* [21] proposed a hierarchical agglomerative algorithm, *Walk-Trap*, that quantified the similarity between vertices using random walks and then partitioned the network into non-overlapping communities. Meilă *et al.* [18] presented a clustering approach by viewing the pairwise similarities as edge flows in a random walk and studied the eigenvectors and values of the resulting transition matrix. A later successful algorithm, *Infomap*, proposed by Rosvall & Bergstrom [23] enabled uncovering hierarchical structures in networks by compressing a description of a random walker as a proxy for

Domain	Dataset	Vertices	Links	Average membership	Maximum membership	Community size mean
Product	Amazon	334,863	925,872	0.11	49	39
Collaboration	DBLP	317,080	1,049,866	0.22	11	251
Social	YouTube	1,134,890	2,987,624	0.05	41	79
Social	Orkut	3,072,441	117,185,083	9.56	504	83

Table 1: Statistics for the real networks.

real flow on networks. Variant of this technique such as biased random walk [28] has also been employed in community finding.

Seed set expansion based approaches. To interpret the problem of community detection from a local perspective, our work is in the same spirit as the seed set expansion algorithms in [6], [7], [11], [13], [14] and [25]. Specifically, Andersen & Lang [6] adapted the theoretical results from [24] to expand a set into a community with locally minimal conductance based on lazy random walks. However, the lazy random walk endured a much slower mixing speed and usually took more than 500 hundred steps before converging to a local structure, which is inefficient compared with several steps of rapid mixing in a regular random walk. Featuring on the seeding strategies, Whang *et al.* [25] established several sophisticated methods for choosing the seed set, and then used similar PageRank scheme as that in [5] to expand the seeds until a community with the optimal conductance is found. Nonetheless, the performance gained by adopting these intricate seeding methods was not significantly better than that by using random seeds. This implies that a better scheme of expanding the seeds is also needed aside from a good seeding strategy. A recent work by Kloumann & Kleinberg [14] provided a systematic understanding of variants of PageRank-based seed set expansions. They showed many insightful findings regarding the heuristics on seed set. However, the drawback of lacking a proper stop criterion has limited its functionality in practice. Even though a recently proposed heat kernel algorithm [13] advances PageRank by introducing a sophisticated diffusion method, the detection accuracy achieved by heat kernel approach is still much lower than that of *LEMON*, which is shown in Section 6.2.

3. PRELIMINARIES

3.1 Problem Statement

Given a network $G = (V, E)$ and a set of members $\mathcal{S}$ in the target community $\mathcal{C}$, where $|\mathcal{C}| \ll |V|$ and $|\mathcal{S}| \ll |\mathcal{C}|$, we are interested in discovering the remaining members in $\mathcal{C}$. Generally speaking, we focus on addressing the question of **how to accurately find a small community in time functional to the size of the community from a seed set?**

3.2 Datasets

3.2.1 Synthetic datasets

The LFR benchmark graphs [15] have been widely adopted for the purpose of evaluating the performance of community detection algorithms. LFR datasets are generated with built-in community structure that resembles the features found in most real-world networks with power-law degree distribution [8]. It provides researchers with rich flexibility to control the network topology through tuning different parameters, including the graph size n, the average degree $\bar{k}$, the maximum degree k_{max}, the minimum and maximum community size $|\mathcal{C}|_{min}$ and $|\mathcal{C}|_{max}$, the mixing parameter μ, the overlapping membership om and the number of vertices with overlapping membership on. Among these parameters, the mixing parameter μ has the most significant impact on the network topology, which controls the fraction of links for each vertex that cross to a community with which the vertex is not associated. Usually, larger μ would result in lower detection accuracy.

Xie et al. [26] have performed a thorough performance comparison of different state-of-the-art overlapping community detection algorithms on LFR benchmark datasets. To make the performance evaluation of our algorithm consistent with that in [26], we adopt the same parameters in our paper. In total, we generate two sets of networks with mixing parameter $\mu = 0.1$ and $\mu = 0.3$ respectively. We vary the parameter om from 2 to 8 for each μ and obtain a total of 14 networks. Table 2 lists the value of the parameters we have used for generating the LFR datasets.

Parameter	Description	Value
n	graph size	5000
μ	mixing parameter	{0.1, 0.3}
$\bar{k}$	average degree	10
k_{max}	maximum degree	50
$\|\mathcal{C}\|_{min}$	minimum community size	20
$\|\mathcal{C}\|_{max}$	maximum community size	100
τ_1	node degree distribution exp.	2
τ_2	community size distribution exp.	1
om	overlapping membership	{2, 3, ..., 8}
on	overlapping node	2500

Table 2: Parameters for the LFR datasets.

3.2.2 Real datasets

For the purpose of testing on real networks, we include four datasets with ground truth community membership from Stanford Network Analysis Project [2]. These datasets span various domains of network applications, including product networks (Amazon), collaboration networks (DBLP), and online social networks (YouTube and Orkut)[3]. Each network can be viewed as an undirected, connected graph. The statistical information of the datasets is summarized in Table 1.

[3]For all the four real datasets, we adopt the top 5000 communities that possess the highest quality according to [27].

3.3 Evaluation Metric

For the evaluation metric, we adopt F1 score to quantify the similarity between the algorithmic community $\mathcal{C}$ and the ground truth community $\mathcal{C}^*$. The F1 score for each pair of $(\mathcal{C}, \mathcal{C}^*)$ is defined by:

$$F_1(\mathcal{C}, \mathcal{C}^*) = \frac{2 \cdot Precision(\mathcal{C}, \mathcal{C}^*) \cdot Recall(\mathcal{C}, \mathcal{C}^*)}{Precision(\mathcal{C}, \mathcal{C}^*) + Recall(\mathcal{C}, \mathcal{C}^*)}, \quad (1)$$

where the precision and recall are defined as:

$$Precision(\mathcal{C}, \mathcal{C}^*) = \frac{|\mathcal{C} \cap \mathcal{C}^*|}{|\mathcal{C}|}, \quad (2)$$

$$Recall(\mathcal{C}, \mathcal{C}^*) = \frac{|\mathcal{C} \cap \mathcal{C}^*|}{|\mathcal{C}^*|}. \quad (3)$$

Throughout the paper, unless otherwise pointed out, the experimental results on synthetic data for each instance are given by the statistical mean and standard deviation based on 24 test cases[4]; and the experimental results on real datasets for each instance are based on 120 test cases. All the ground truth communities for testing are randomly chosen. The randomness of batch tests can guarantee the elimination of potential sampling bias in our tests.

4. LOCAL EXPANSION VIA MINIMIZING ONE NORM

4.1 Algorithm Overview

Spectral clustering makes use of a small number of singular vectors proportional to the number of communities in the network. If a graph has thousands of small communities, it is impractical to calculate a number of singular vectors greater than the number of communities. We are experimenting with a fundamentally new technique, which does not require the burdensome computation of a large number of singular vectors. Before explaining our local spectral approach for finding overlapping communities, it is necessary to make clear what we mean by "local spectra".

In traditional spectral clustering methods, one finds the first few singular vectors of the Laplacian matrix[5] of a graph G with n vertices. Suppose the first d singular vectors are obtained, one can form an $n \times d$ matrix as a latent space. Then one associates with each vertex a point in this latent space whose coordinates are given by the entries of the corresponding row in the matrix. Vertices are clustered using some method such as k-means clustering algorithm. This method is not likely to work well if the communities are small and heavily overlapping with each other.

We make two fundamental changes to this method. The first modification is to overcome the drawback of computing the singular vectors. Intuitively, the vertices around the seed members are more likely to be in the target community, thus a random walk serves as a natural subroutine to reveal these potential members. We start a random walk from several known members in the target community and run for a few steps. The number of random walk steps should be long enough to reach out to the vertices in the target community, but not too long to spread out to the entire graph. Instead of considering a single probability vector, we consider the span of a few dimensions of vectors after the short random walks and use it as the approximate invariant subspace (*local spectra*). The second is to handle the overlapping situation. Instead of using k-means to partition the points in the latent space into disjoint clusters, we look for the minimum 0-norm vector in the span of the invariant subspace obtained above, such that the seed members are in its support. We want to find rows in the invariant subspace that point in nearly the same direction as seed members. We will use 1-norm vector as a proxy for the minimum 0-norm vector since finding the 0-norm vector is an NP-hard problem.

In the following, we give a formal description of our local spectral approach *LEMON* for detecting the target communities from a small seed set. Given the input of a set of few vertices $\mathcal{S}$ that are already known to be in the target ground truth community $\mathcal{C}^*$, our algorithm would output the algorithmic community $\mathcal{C}$ such that the F1 measure for scoring the similarity between $\mathcal{C}$ and $\mathcal{C}^*$ is maximized.

Step 1. Generate the local spectra:

Let $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-1/2}$ be the normalized adjacency matrix of the graph where $\mathbf{D}$ is the diagonal matrix of vertex degree. Consider a random walk starting from exemplary vertices in $\mathcal{S}$. Let $\mathbf{p_0}$ denote the initial probability vector where the total probability is evenly distributed among the seed members. Consider the span of l-dimensional probability vectors which consist of probability vectors in l successive random walks

$$\mathbf{P}_{0,l} = [\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_l]. \quad (4)$$

The initial invariant subspace is then obtained by calculating the orthonormal basis of the span $\mathbf{P}_{0,l}$, which we denote by $\mathbf{V}_{0,l}$. We then use the following recurrence to iteratively calculate the l-dimensional orthonormal basis $\mathbf{V}_{k,l}$ after k steps of random walk

$$\mathbf{V}_{k,l}\mathbf{R}_{k,l} = \mathbf{V}_{k-1,l}\bar{\mathbf{A}}, \quad (5)$$

where $\mathbf{R}_{k,l} \in \mathbb{R}^{n \times l}$ is chosen such that $\mathbf{V}_{k,l}$ is orthonormal. The orthonormal basis $\mathbf{V}_{k,l}$ will be used as the local spectra for clustering.

Step 2. Seek for a sparse vector

With the local spectra $\mathbf{V}_{k,l}$, we then solve the following linear programming problem [1].

$$\begin{aligned} \min \quad & \mathbf{e}^T\mathbf{y} = ||\mathbf{y}||_1 \\ s.t. \quad & \mathbf{y} = \mathbf{V}_{k,l}\mathbf{x}, \\ & \mathbf{y} \geq \mathbf{0}, \\ & \mathbf{y}(\mathcal{S}) \geq 1, \end{aligned}$$

where $\mathbf{e}$ is a vector of all ones, and both $\mathbf{x}$ and $\mathbf{y}$ are unknown vectors. The first constraint indicates that $\mathbf{y}$ is in the space of $\mathbf{V}_{k,l}$. The element in $\mathbf{y}$ indicate the likelihood for the corresponding vertex belong to the target community, which is non-negative. The third constraint enforces that seeds are in the support of sparse vector $\mathbf{y}$.

After sorting the elements in $\mathbf{y}$ in non-ascending order and getting a vector $\hat{\mathbf{y}}$, the vertices corresponding to the top $|\mathcal{C}|$ elements in $\hat{\mathbf{y}}$ are returned as the detected community with respect to the seed set $\mathcal{S}$.

[4]Each local expansion process from a seed set can be viewed as a test case.

[5]In the literature, several different definitions of graph Laplacian exist. Readers can refer to [20] for more details, which serves as a good introductory paper on spectral clustering.

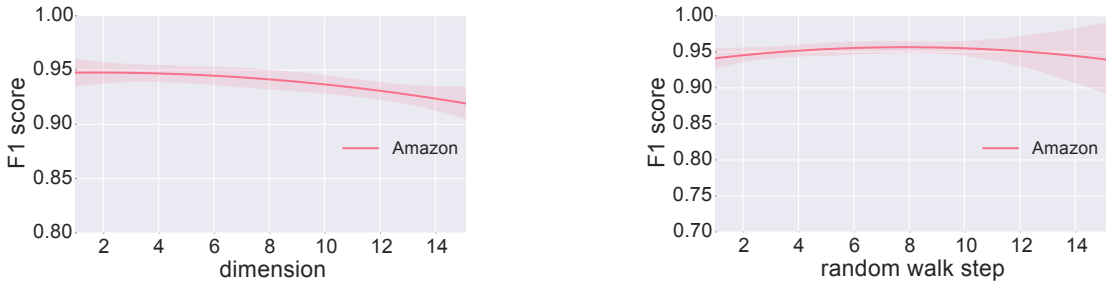

Figure 1: The average F1 score on Amazon network with varying dimensions l and random walk step k, respectively. The plots depict the statistical regression line with a 95% confidence interval.

Step 3. Reseeding

Augment the initial seed set by adding the vertices corresponding to the top t elements of $\hat{\mathbf{y}}$. Denote the augmented seed set as $\mathcal{S}'$. Then repeat step 1 and step 2 using the augmented seed set $\mathcal{S}'$. The detection accuracy can be improved through iterations via increasing t by a constant number s each time. We define s to be the seed expansion step, which is used as a tunable parameter for adjusting the convergence rate. Usually, the larger expansion step would result in lower performance but a faster running speed with less iterations. In the experiments, we fix the seed expansion step to be 6 for both synthetic and real datasets. The number of iterations for the seed expansion is determined by the stop criteria (Section 4.4).

4.2 Parameter Selection

The random walk step k and subspace dimension l are the key parameters in the local spectral clustering algorithm. We conduct parameter sensitivity study for these two parameters on the four real datasets.

4.2.1 *Subspace dimension*

To study the parameter of subspace dimension l, we fix the random walk step to be 3, and vary the number of dimension l from 1 to 15. Figure 1 (left panel) shows that changing the dimension l does not cause significant performance fluctuation. On one hand, choosing a large dimension l is undesirable because it would increase the computation cost of generating local spectra. On the other hand, when dimension degrades to $l = 1$, the standard deviation of F1 score becomes significant, making the detection accuracy unstable. In this paper, we fix $l = 3$ because the experiment suggests that setting $l = 3$ can statistically achieve both high and stable performance. Note that such observation holds not only for Amazon network, but for the remaining real datasets as well.

4.2.2 *Random walk step*

To investigate how the step of random walk affects the algorithm performance, we fix the dimension l to be 3, and vary the random walk step k from 1 to 15. Figure 1 (right panel) shows that the average F1 score plateaus as k increases, and 3-step random walk can yield the algorithm's full potential. The standard deviation, however, significantly increases when k exceeds 10. This indicates that longer random walk is undesirable for stably uncovering the local community structure. Throughout the paper, we fix the random walk step $k = 3$ for the real datasets[6].

4.3 Complexity Reduction by Sampling Method

If one wants to uncover a small community within a large network with billions of vertices, it would be very costly to take all the vertices into account. We want to discover the target community accurately while keeping the number of vertices examined small. Sampling method can effectively solve the memory consumption issue when one wants to find a local community within a large graph, since the whole graph does not have to be loaded into memory.

In practice, the unknown members in the target community are more likely to be around the seed members, and are usually a few steps away from the seeds. This observation motivates us to reduce the complexity by taking only a portion of the graph into consideration. Ideally, this partial graph should contain as many vertices in the target community as possible, and maintains a small size of the same scale as that of the target community.

To sample the graph, we expand the seed set using random walk. After a few steps of the random walk, vertices with large probability are more likely to be in the target community while vertices with small probability being reached would be treated as redundant ones. If the target community exists for the seed set, then according to [6], this target community would serve as a bottleneck for the probability to be spread out. It is worthwhile noting that other expansion methods such as breadth-first-search (BFS) would entirely ignore the bottleneck defining the community and rapidly mix with the entire graph before a significant fraction of vertices in the community have been reached. The subgraph returned by BFS usually contains less vertices in the target community than the subgraph of the same size obtained by random walk technique.

In the experiments on real datasets, we conduct a random walk starting from the seed set until the probability has been spread out to $\alpha \cdot |\mathcal{C}|_{\text{avg}}$ vertices, where α is some constant and $|\mathcal{C}|_{\text{avg}}$ is the average community size in the graph[7]. Note that $\alpha \cdot |\mathcal{C}|_{\text{avg}}$ should be large enough to be able to cover as many vertices in the ground truth community as possible.

[6]For LFR benchmark graphs, we adopt all together 6 combinations for the (step, dimension) tuple: $(2,3),(2,4),(2,5),(3,3),(3,4),(3,5)$ and return the highest F1 score among using these combinations.

[7]A fast implementation for updating the probability vector of the random walks is featured in detail in [6], Section 4.

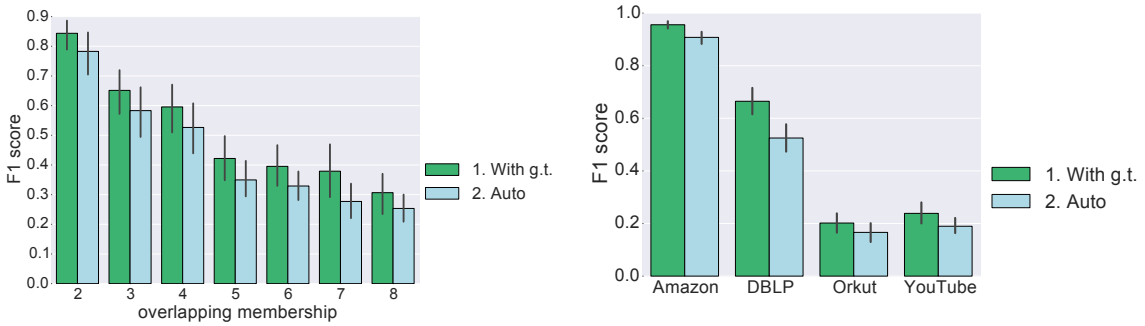

Figure 2: Comparison of the average F1 score with ground truth and automatic size determination. The left corresponds to the LFR datasets when $\mu = 0.3$ and the right corresponds to the real datasets.

This newly obtained subgraph will be used for the remaining computation.

Table 3 gives the statistics after applying sampling method to the real networks. For example, in DBLP network, setting α to be around 10 would yield a subgraph containing on average 98% vertices in the ground truth community. After sampling, we only need to deal with a subgraph of size around 2400 instead of 317,080, bringing a significant reduction of both temporal and spatial complexity.

Dataset	Coverage ratio	Sample rate	$\|\mathcal{C}\|_{\text{avg}}$	Subgraph size
Amazon	1.00	0.0087	39	2913
DBLP	0.98	0.0076	251	2409
YouTube	0.66	0.0033	79	3745
Orkut	0.64	0.0011	83	3379

Table 3: Statistics of the mean values for the sampling method on real datasets.

4.4 Stop Criteria

If there are ground truth communities available, the above algorithm is guaranteed to stop within few iterations since the seed set will no longer augment once its size exceeds that of the ground truth community. The algorithm would then return the community found with the highest F1 score during the iterations as the result. However, in real case, without knowing the exact size of the communities, most locally based detection algorithm has difficulty deciding when is the proper time to terminate expanding such that the discovered community is a "good" community. It is thus important to solve the two issues: 1) how to automatically determine the size of the community given a seed set $\mathcal{S}$, and 2) when to stop growing the seed set during the reseeding process.

4.4.1 *Determine the size of the community*

It has already been shown that random walks produce communities with conductance guarantees and ensure a small boundary defining a natural community in locally based detection algorithms [6]. The intuition is that adding irrelevant vertices to the target community would inevitably cause the conductance to increase, and finding a low-conductance community could ensure the closeness between the detected members and the known seed set. In [25], the authors also follow the same idea and adopt conductance as the metric for defining a good community found by the algorithm around a seed set. As we will see, the local conductance for a small group of vertices in the graph contains valuable information and enables designing effective stop criteria for our algorithm.

The definition of the conductance for a set of vertices $\mathcal{C}$ is given by

$$\phi(\mathcal{C}) = \frac{|\partial(\mathcal{C})|}{\min(\text{Vol}(\mathcal{C}), \text{Vol}(\bar{\mathcal{C}}))}, \tag{6}$$

where $|\partial(\mathcal{C})|$ denotes the cut size, and $\text{Vol}(\mathcal{C})$ is the sum of vertex degree in the set $\mathcal{C}$.

Suppose we have a rough estimation of the lower and upper bound for the size of communities in a graph, which we denote by $|\mathcal{C}|_{\min}$ and $|\mathcal{C}|_{\max}$ respectively. We could modify the original algorithm in the following way.

At step 2, after obtaining the sorted sparse vector $\hat{\mathbf{y}}$, we are hoping to truncate the sorted vector at some point y_g such that all the vertices corresponding to the elements no less than y_g are included in the algorithmic community. The crux lies in that we do not know yet what the best position is to truncate the vector $\hat{\mathbf{y}}$. To solve this issue, we denote Λ_i as the set of vertices corresponding to the top i elements in $\hat{\mathbf{y}}$. We then sweep over the sets from $\Lambda_{|\mathcal{C}|_{\min}}$ to $\Lambda_{|\mathcal{C}|_{\max}}$ and calculate the corresponding conductance for each of the sets. In practice, the value of the conductance with respect to varying size would usually change in a non-monotonic pattern that decreases first and then increases later on. We then adopt the first relative minimum conductance encountered on this curve as the estimated size of the community with respect to the seed set $\mathcal{S}$, which we denote by $\phi_{\mathcal{S}}^{\min}$.

4.4.2 *Stop the reseeding process*

As we keep augmenting the seed set through reseeding at step 3, a different seed set would result in a different sparse vector $\hat{\mathbf{y}}$ and thus lead to potentially different algorithmic communities. Practically, one of these seed sets during the augmenting process would achieve the highest F1 score. And it remains to address the issue of when to stop growing the seed set so that it finds the community that resembles most of the ground truth community. This issue can be solved in a similar fashion as that for determining community size.

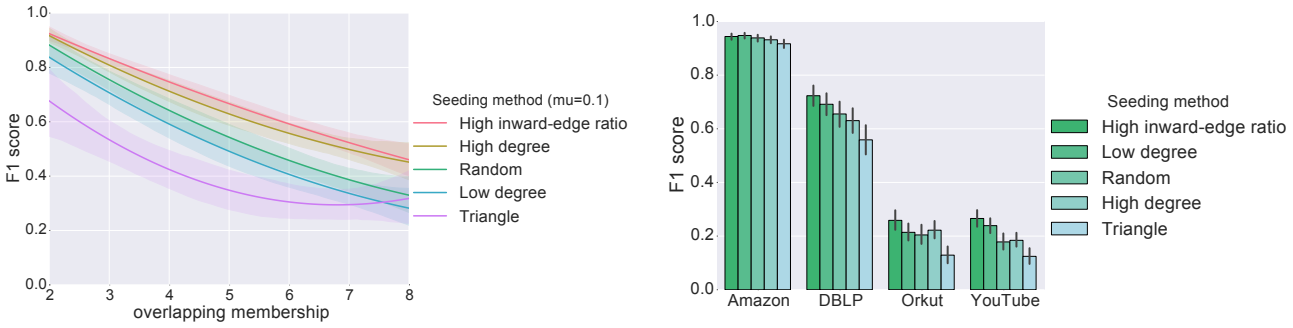

Figure 3: The average F1 score on LFR datasets ($\mu = 0.1$) and real datasets with different seeding methods.

Specifically, we keep track of the value of $\phi_S^{\min}$ for different seed set during the expansion, and stop to grow the seed set when $\phi_S^{\min}$ reaches a local minimum and starts to increase for the first time.

4.4.3 *Auto detect size vs ground truth size*

To verify our method, we compare the performance after applying the stop criteria with that obtained using ground truth communities. Figure 2 shows the statistical result of F1 score on both synthetic and real datasets. On both datasets, the F1 score with automatic size determination is only lowered by 10% on average compared with the performance with available ground truth. This implies that our method is applicable for finding communities that mostly resemble the ground truth communities on both synthetic and real datasets in different domains. It also suggests that our method can be applied in practice to uncover natural communities in the situation when no ground truth is available.

5. SEEDING

Since the initial seed set serves as a key component in our algorithm for uncovering the target community $\mathcal{C}$, it is crucial to consider how the quality of seed set affect the performance. In practice, there is not much control over how the seeds are selected. However, the alternative seeding methods we provide here can be strategically applied by the domain experts based on the availability of candidate seeds in different scenarios. In this section, we will focus on addressing two fundamentally important issues regarding the seed set: 1) What defines "good" seeds? and 2) How many seeds are needed in order to uniquely define a community?

5.1 Seeding Method

To give a well-rounded evaluation on this, we encompass in total five different seeding methods here. In this experiment, we adopt $|\mathcal{S}| = 3$ seeds for each of the seeding method listed below.

- **High degree seeding**: pick $|\mathcal{S}|$ vertices with degree ranked in the top one third among the degree of all vertices in $\mathcal{C}$.
- **Low degree seeding**: pick $|\mathcal{S}|$ vertices with degree ranked in the bottom one third among the degree of all vertices in $\mathcal{C}$.
- **Triangle seeding**: pick $|\mathcal{S}|$ vertices in $\mathcal{C}$ that form a triangle as the initial seed set.
- **Random seeding**: pick $|\mathcal{S}|$ vertices in $\mathcal{C}$ randomly.
- **High inward-edge ratio seeding**: the inward-edge ratio for a vertex v is defined by the fraction of links connecting to another vertex inside the target community $\mathcal{C}$ among all the links coming out from v. We pick $|\mathcal{S}|$ vertices with inward-edge ratio ranked in the top one third among all vertices in $\mathcal{C}$.

Figure 3 (left panel) gives the experimental results on LFR benchmark datasets with mixing parameter $\mu = 0.1$. It is interesting to note that the high-degree seeding method consistently achieves the highest F1 score in both groups of datasets. When $\mu = 0.1$, triangle seeding leads to the worst performance with low F1 score and high standard deviation. This implies that seeding from a compact core structure is less advantageous than seeding sporadically among vertices. The intuitive explanation behind this phenomenon is that it is more difficult for the probabilities to spread out when the random walk initiates from a cohesive structure.

Another interesting observation is that high inward-edge ratio seeding method can consistently lead to the best performance among different seeding methods on both synthetic and real datasets. In [14], the authors have the same observation as ours but did not give an explicit explanation on this phenomenon. In fact, when a large fraction of the seeds links connect to vertices within the same community, random walks starting from these seeds would be more likely to transit probabilities into the vertices within the community rather than spreading out to vertices outside the community. A higher detection accuracy can be thus achieved since the target community contains much of the probability after short random walks.

Moreover, it is also striking to note the difference between the test results on synthetic datasets and that on real datasets. Even though the high-degree seeding method can always bring higher performance than that of random seeding on synthetic datasets, the behavior of these seeding methods on real networks is quite different. In Figure 3 (right panel), we see that low-degree seeds lead to better result than that of high-degree seeds on DBLP and YouTube datasets. The degree of seeds does not have a significant impact on the performance in Amazon and Orkut networks since the performance of high-degree seeding and low-degree

seeding almost tie with each other on these datasets. In [14], the authors compared the detection accuracy of PageRank based seed set expansion algorithm with high-degree seeding and random seeding on real networks, and concluded that random seeding method always outperforms high-degree seeding in all domains of real networks. However, we remark here that this observation does not apply to our algorithm as we find that high-degree seeding works slightly better than random seeding on Orkut and YouTube datasets.

5.2 Seed Set Size

It is also interesting to investigate how the size of the seed set affects the performance of our algorithm.

We first experiment on the LFR benchmark datasets with varying seed set size. We choose seed set of size proportional to the size of the target community $\mathcal{C}$. Specifically, we test with five different seeding ratios r: 2%, 4%, 6%, 8% and 10% respectively, and round $r \cdot |\mathcal{C}|$ to an integer if it is a fraction. Figure 4 shows the F1 scores when $\mu = 0.1$. The algorithm's performance can be improved in general as the seed set size increases. In the case when both mixing parameter and overlapping membership are small, e.g., $\mu = 0.1, om = 2$, increasing the seed set size does not seem to affect the performance significantly, and seed set consisting of a small percentage of vertices are sufficient to discover the target community with high accuracy. This implies that when the structure of a small community is well-defined, our algorithm only needs 2 to 3 seeds to reveal the remaining members in a community of size roughly 100. For general test purpose on LFR benchmark graphs, we adopt an 8% fraction of the vertices in the target community as seeds.

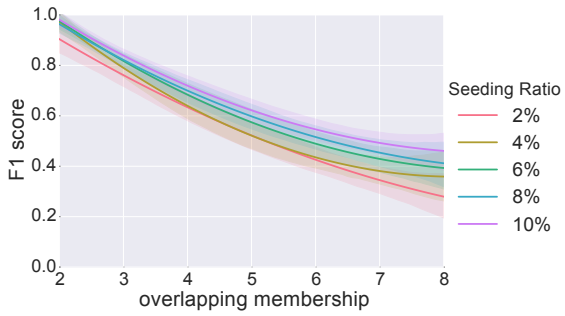

Figure 4: The average F1 score on LFR datasets ($\mu = 0.1$) with different seeding ratio.

We then carry out the similar experiment on the real datasets. The result on real networks is interesting because increasing the seed set size has little affect on the performance. Especially, using only 3 seeds can yield almost the same performance as using an 8% fraction of the vertices in the target community as seeds on real datasets.

Our algorithm is thus advantageous to many other seed set expansion algorithms that usually require a higher fraction of vertices to be known. For example, in [14], the authors perform a similar experiment on DBLP network. The performance of their algorithm achieves the maximum recall of 0.3 when seeding ratio is 10%, while *LEMON* can achieve an average F1 score of 0.66 with 3 vertices. This makes our algorithm practical for real networks when it is impossible to collect a large number of seeds.

5.3 Further Extension

As the results of using different seeding methods suggests, high-degree seeds can heuristically lead to better result on synthetic data. Such heuristic implies that a vertex with higher degree may exert higher impact on shaping the subspace we are looking for, and thus affect the performance by leading to different sparse vectors where we obtain the "candidates" of the target community from.

In practice, we usually have little control on the seed set. The chance we get a seed set of high-degree members is rare. More often than not, the degree of seeds is randomly distributed. We are therefore inspired to tailor our algorithm accordingly in order to emphasize the seeds with high degree. The modification is rather straightforward: when calculating the initial probability vector p_0 to start a random walk from, instead of evenly distributing the amount of probability to each seed, we initialize the probability vector according to the degree of each seed. Formally,

$$p_0(v_i) = \begin{cases} d(v_i)/\mathrm{Vol}(\mathcal{S}) & \text{if } v_i \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $d(v_i)$ denotes the degree of vertex v_i. In other words, we enforce a bias towards the high-degree vertices at the beginning of the random walk. Note that each time after the reseeding process, the initial probability vector also needs to be recalculated in the same way.

Figure 5 (left panel) depicts the experimental results on LFR benchmark graphs with and without degree normalized initialization for the random walk respectively. We can find that degree-normalization of the initial probability vector results in better performance.

We then perform the same experiments on real networks, and find that degree-normalization would on the contrary, lead to slightly worse statistical results (see the right panel of Figure 5). The completely different behavior of using degree normalization on real datasets is rather intriguing. In fact, this phenomenon accords with our previous observation in Section 5.1 that a high-degree seed set is less advantageous than random seeds on real datasets. And this explains why emphasizing on the high-degree vertices would worsen the performance on real datasets.

6. COMPARISON WITH THE STATE-OF-THE-ART ALGORITHMS

6.1 Baseline Algorithms

To give a well-rounded performance comparison with state-of-the-art algorithms, we compared our results to three localized community detection algorithms and four global community detection algorithms.

1. **Localized algorithms:** We encompass three locally based methods, Heat Kernel (HK) [13], PageRank (PR) [14] and Seed Set Expansion (SSE) [25].

2. **Global algorithms:** We also compare our local spectral clustering algorithm with four overlapping community detection methods that are based on the global structure: OSLOM[8] [16], DEMON[9] [9], and LinkCommunity[10] (LC) [4].

[8] `http://www.oslom.org/software.htm`
[9] `http://www.michelecoscia.com/?page_id=42`
[10] `https://github.com/bagrow/linkcomm`

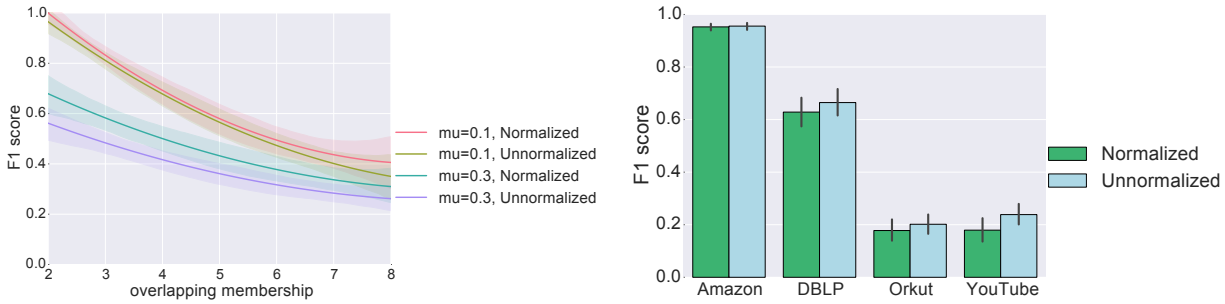

Figure 5: Comparison of the average F1 score with and without normalizing the initial probability vector by each seed's degree. The left corresponds to the LFR datasets and the right corresponds to the real datasets.

6.2 Comparison with Localized Algorithms

We refer to the experimental results reported in some recent publications on localized community detection algorithms [13][14][25]. Figure 6 illustrates the comparison of F1 scores on Amazon, DBLP, YouTube and Orkut datasets. We use "LEMON-auto" to denote the results obtained by applying the stop criteria in Section 4.4. Since the results on Orkut and YouTube datasets are missing in [25] and [14], we use empty bars to indicate them.

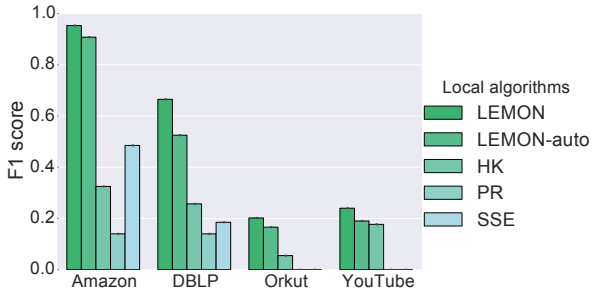

Figure 6: Comparison of the average F1 score with state-of-the-art local detection algorithms on real networks.

Figure 6 shows that *LEMON* achieves an F1 score of 0.910 on the Amazon dataset, far outperforming the other algorithms. The average F1 scores increases the performance by 3 times compared with the heat kernel algorithm [13] on Amazon, DBLP and Orkut networks. To compare with [25], we find that the average F1 score of our algorithm doubles their best performance achieved by the "spread hubs" method on Amazon dataset and triples the performance on the DBLP network. Also, note that in [14], the authors did not have an explicit stop criterion. Instead, they assumed using a budget for predicting the size of the target community. We compare with the F1 score at a budget of 100 for both Amazon and DBLP datasets. From the results on Amazon networks in [14], we notice that even granted a budget of 400, which is far beyond the average community size of 39 in Amazon network, only a recall of 0.45 can be achieved. And we infer the F1 score would be even lower than this value since the precision is dragged down by the large budget set.

It is also worth noting that we only use 3 randomly picked seeds for all the test cases on each dataset. Our algorithm requires very fewer seeds than other algorithms such as [14].

The experiment has verified that our algorithm is able to achieve high accuracy on large networks constituting communities of average size roughly hundred. This implies that our approach is well-suited for the task of detecting small communities in large networks.

6.3 Comparison with Global Algorithms

We also compare local spectral clustering with several state-of-the-art global based algorithms. Table 4 summarizes the running time as well as the average F1 score of each algorithm on real datasets. Among the baselines, OSLOM and LC fail to terminate within 10 days on the YouTube dataset. The OSLOM algorithm can achieve rather good performance but does not scale well.

In contrast, our algorithm can consistently return the result within few seconds irrespective of how large the entire graph is. Besides, our algorithm has small memory consumption, and a machine with 4GB RAM can afford to process networks as large as Orkut since the algorithm does not have to store the whole graph in memory. Moreover, our locally based algorithm is parallelizable because each seed set expansion can be computed independently. Such property can bring a further performance gain on running time with multi-threaded implementation.

Figure 7 compares the average F1 score with some state-of-the-art algorithms on LFR benchmark graphs. During the experimentation, we also incorporate the methods that can effectively improve the performance on synthetic datasets that are addressed in Section 5.3. We notice that our algorithm outperforms the baseline algorithms even when we use the random seeding strategy. When the mixing parameter $\mu = 0.3$, as is shown in Figure 7, *LEMON* brings about $30\% \sim 40\%$ relative improvement compared with the best results among the baselines. And we can expect the performance gain to be even more significant if the seeds possess the qualities discussed in Section 5.1.

Among the four baselines, we notice that LC and DEMON consistently perform poorly on both groups of the synthetic datasets. We further look into the communities found by LC and DEMON respectively, and find that LC tends to partition the graphs into very small pieces while DEMON, on the contrary, usually finds communities that are much larger

Algorithm	Implementation	Amazon		DBLP		YouTube		Orkut	
		F1 score	time	F1 score	time	F1 score	time	F1 score	time
LEMON	Python	0.953	<15s	0.665	<15s	0.240	<15s	0.202	<15s
LEMON-auto	Python	0.910	<15s	0.525	<15s	0.190	<15s	0.170	<15s
DEMON	Python/C++	0.164	4,562s	0.196	727,675s	0.031	22,395s	-	-
OSLOM	C++	0.766	885,867s	0.542	23,262s	-	>10d	-	-
LC	Python/C++	0.815	4,606s	0.527	49,045s	-	>10d	-	-

Table 4: Comparison with global algorithms on real datasets.

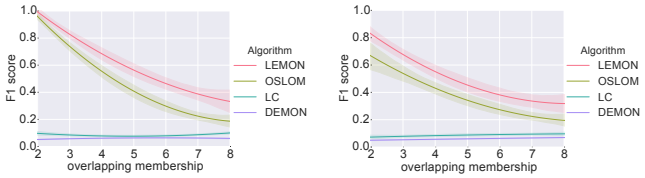

Figure 7: Comparison of the average F1 score on LFR datasets with baseline algorithms. The left corresponds to the datasets with mixing parameter $\mu = 0.1$ and the right corresponds to $\mu = 0.3$.

than the ground truth communities. This implies that both algorithms extract structures from networks that bear little resemblance to the natural formation of the communities. However, we remark here that even LC fails to recognize the communities well on the synthetic data, it perform better on real datasets as we see in Table 4 .

6.4 Empirical Comparison Between Synthetic and Real Data

Networks are not all similar and we cannot assume one algorithm works for finding communities in a network will behave the same on the other networks. Therefore, it is important to develop the understanding of how different types of networks affect the behavior of algorithms.

Our algorithm sustains a consistent performance on both LFR benchmark graphs and real networks though, we still want to summarize and call the attention to several subtle differences here.

First, *LEMON* is less sensitive to the parameter of random walk step k and subspace dimension l on real networks than that on LFR benchmark graphs. In practice, fixing (k, l) to be $(3, 3)$ for real networks can ensure a good performance.

Second, *LEMON* is less sensitive to the seed set size on real networks than that on LFR benchmark. In practice, a seed set size of 3 can guarantee a good performance on real networks. As for LFR, we adopt the seed set size to be proportional to the community size.

Third, *LEMON* is more sensitive to the high-degree seeds on real networks than that on LFR benchmark. In LFR graphs, the degree of a vertex is at most 50. Whereas in some large real networks such as YouTube, the degree of some vertices exceeds 1000, making the degree distribution much more screw than that seen in LFR graphs. And we expect that vertices with unusually high degree in real networks would have a stronger power in controlling the trend for the probabilities to spread out during the random walk, and thus have a higher risk to enter some other neighboring communities. Such an effect can be counterbalanced by putting less initial probabilities on these "super cores".

The above empirical analysis informs us that finding communities in real networks seems to be less parameterized than that on synthetic datasets for our algorithm. This indicates that our algorithm is better suited for uncovering those naturally well-formed communities than the artificially constructed communities in practice.

7. CONCLUSION

The problem of identifying small community structure in large networks has been gaining importance. In this paper, we have presented a method for finding overlapping communities by seeking a sparse vector in the span of local spectra where the seeds are in its support. To overcome the drawbacks of traditional spectral clustering methods, we propose a novel method to construct the local spectra based on the singular vector approximations drawn from short random walks. Our algorithm enables finding a small community in time functional to the size of the community, and it consistently returns the result within seconds even for a network with billions of vertices. We demonstrate the effectiveness and efficiency of our method for discovering communities on both synthetic and real-world datasets. As the experimental result shows, our algorithm achieves the highest detection accuracy amongst the state-of-the-art proposals.

Many other fundamentally important research questions remain to be addressed. First, the community detection algorithm based on local spectral clustering could be potentially applied to the membership detection problem, i.e., finding all the communities that an arbitrary vertex belongs to. Second, during the process of seed set expansion, we adopt the first low-conductance community as the target community, which usually yields a high resemblance to the ground truth community. It would also be interesting to look further into some larger low-conductance communities and see if a hierarchical structure exists. In this case, some large social group consisting of several small cliques is likely to be discovered.

Acknowledgements

The authors would like to thank Isabel Kloumann for valuable comments and discussions. This research has been supported in part by US Army Research Office W911NF-14-1-0477 and National Science Foundation of China 61173180, 61472147.

8. REFERENCES

[1] GNU linear programming kit. `http://www.gnu.org/software/glpk`.

[2] Stanford Network Analysis Project. `http://snap.stanford.edu`.

[3] B. Abrahao, S. Soundarajan, J. Hopcroft, and R. Kleinberg. A separability framework for analyzing community structure. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):5, 2014.

[4] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[5] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, pages 475–486. IEEE, 2006.

[6] R. Andersen and K. J. Lang. Communities from seed sets. In *WWW*, pages 223–232. ACM, 2006.

[7] M.-F. Balcan, C. Borgs, M. Braverman, J. Chayes, and S.-H. Teng. Finding endogenously formed communities. In *SIAM*, pages 767–783. ACM, 2013.

[8] A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[9] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Demon: a local-first discovery method for overlapping communities. In *KDD*, pages 615–623. ACM, 2012.

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

[11] T. J. Hansen and M. W. Mahoney. Semi-supervised eigenvectors for large-scale locally-biased learning. *Journal of Machine Learning Research*, 15:3691–3734, 2014.

[12] D. Jin, B. Yang, C. Baquero, D. Liu, D. He, and J. Liu. A markov random walk under constraint for discovering overlapping communities in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(05):P05031, 2011.

[13] K. Kloster and D. F. Gleich. Heat kernel based community detection. In *KDD*. ACM, 2014.

[14] I. M. Kloumann and J. M. Kleinberg. Community membership identification from small seed sets. In *KDD*. ACM, 2014.

[15] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[16] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[17] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, pages 695–704. ACM, 2008.

[18] M. Meila and J. Shi. A random walks view of spectral segmentation. 2001.

[19] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random structures & algorithms*, 6(2-3):161–180, 1995.

[20] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[21] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer, 2005.

[22] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[23] M. Rosvall and C. T. Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209, 2011.

[24] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90. ACM, 2004.

[25] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using seed set expansion. In *CIKM*, pages 2099–2108. ACM, 2013.

[26] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys (CSUR)*, 45(4):43, 2013.

[27] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, pages 10–13, 2012.

[28] V. Zlatić, A. Gabrielli, and G. Caldarelli. Topologically biased random walk and community finding in networks. *Physical Review E*, 82(6):066109, 2010.