Chapter 1

Spectral Community Detection

Contributed by Satyaki Sikdar

1.1 Introduction

Humans are highly selective when choosing who they interact with. This is visible while people choose who they want to be friends with from a group of people, or when researchers choose their collaborators. Social scientists call this process link formation. As expected, the process of link formation in complex networks, is deliberate and non-random. As a result, not every node gets the same share of edges to connect. A small number of nodes, called hubs, become part of the majority of the links, while the other nodes share a modest number of links between them [3]. In the world wide web and social networks, hubs hold a position of authority and influence. For example, Facebook, Google, and Amazon dominate their respective domains despite the presence of hundreds of competitors [20]. This is partly attributed to them being hubs in the complex network of the World Wide Web.

Sociologists first studied the theory of link formation in social networks, looking at the interaction between groups of people. The findings identified *homophily* [34], or the tendency of individuals to bond with other individuals they share a collective identity — gender, race, class, political views, and social roles, as the main driving force [9, 1, 35]. This not only results in the formation of links but also determines their strength. More frequent, stronger ties form between more similar users, and weaker ties which keep the network together form sporadically [17]. This leads to the formation of communities or clusters in the network, with nodes in each community having more links to other nodes in the same community than to the nodes in the rest of the network. This effect is not restricted to just social networks. In specific networks, communities make intuitive sense. For example, in social and telecommunication networks, clusters represent social circles; in collaboration networks, the clusters represent researchers working on similar areas of research, and so on. Moreover, communities are often nested, with smaller communities combining to form larger communities [26]. In the collaboration network of researchers across multiple disciplines, each discipline could be separable into large individual clusters, and inside each of them, there could be smaller clusters representing sub-disciplines. Extracting this higher-order interaction between the nodes is very useful in tasks like graph mining and graph compression. Community detection techniques therefore are used widely in many graph compression [22] and summarization algorithms [23].

While the presence of community structure in complex networks is ubiquitous, the degree of expression varies. In networks like the collaboration network among researchers [37], the clusters are highly separable. Researchers are more much more likely to collaborate with people who work in similar areas as them. The small group of researchers who do inter-disciplinary research keep the entire network connected. However, in some cases, as in friendship networks, the clusters represent social circles, which by nature are overlapping and messy [30]. Most community detection algorithms usually are more effective in graphs where the clusters are well defined.

1.2 The Problem as a Graph

In a graph $G(V, E)$ where V is the set of nodes and E is the set of edges, a community detection algorithm generates a cover $\mathbb{C} = \{C_1, C_2, \cdots, C_k\}$ of k communities where nodes lying in the same community are placed in the same set, and $\bigcup_i C_i = V$, that is, every node in the network is assigned to at least one community.

In disjoint community detection, each node is assigned to exactly one community. Formally, $\forall\ i, j,\ C_i \cap C_j = \emptyset$. In comparison, in overlapping community detection, each node can be a part of multiple communities, that is, $\exists\ i, j,\ C_i \cap C_j \neq \emptyset$.

The intuition behind community structures given in Section 1.1 can be formalized as follows. For a community C with n_C nodes, let $n_{int}(C)$ and $n_{ext}(C)$ denote the internal and external edge counts, signifying the number of edges having both endpoints and exactly one endpoint in C respectively. Both these counts are normalized by their maximum values to get the internal and external edge densities represented by $\delta_{int}(C)$ and $\delta_{ext}(C)$. The numerator of $\delta_{ext}(C)$ is also called the cut value of C (cut(C)). For a good clustering $\mathbb{C}$, the internal edge density for each cluster should be much greater than the external edge density. Mathematically,

$$\delta_{int}(C) = \frac{\sum_{\substack{i \in C \\ j \in C}} A_{ij}}{\binom{n_C}{2}} \qquad \delta_{ext}(C) = \frac{\sum_{\substack{i \in C \\ j \notin C}} A_{ij}}{n_C \cdot (n - n_C)} \qquad \text{cut}(C) = \sum_{\substack{i \in C \\ j \notin C}} A_{ij}$$

where A and n represent the adjacency matrix and the number of nodes in the graph G.

In this chapter, only disjoint community detection techniques are analyzed.

1.3 Some Realistic Data Sets

There are several repositories of real-life networks on the internet [10, 29, 24]. Almost all complex networks are expected to have some amount of community structure when compared to a random graph of similar size. As described in Section 1.1, graphs with well defined clusters are expected to contain a large number of triangles [6] as they are indicative of the presence of local cliques.

Due to the universality of the phenomena, it is observed in graphs of all scales. In the case of Facebook, the famous social network, their user graph as of 2014 had 1.39 billion active users and 400 billion edges [8]. Co-authorship networks are another class of networks that are of great interest to researchers. In Computer Science, for example, DBLP stores the information of 4.3 million publications made by 2.1 million authors across over 5,000 conferences as of September 2018. This sheer scale dramatically magnifies the difficulty level of the problem. However, through the advances in research in distributed computing and using novel computing paradigms [43, 31, 46, 32], researchers can crunch these massive networks and run the necessary algorithms.

There also exists several artificial graph generation algorithms which produce synthetic graphs having a defined community structure which are frequently used to validate the effectiveness of community detection algorithms. The *planted partition model* [11] for example, takes in the number of nodes n, the number of communities l, and the mixing parameter μ as an input. A node shares a fraction $1 - \mu$ of its links with the other nodes of its community and a fraction μ with the other nodes in the network. A lower μ signifies a more prominent community structure. The planted partition model generates a network having l groups of (nearly) equal size. Benchmarks like the *LFR benchmark graph generator* [27] is more flexible than the planted partition model. In addition to the parameters in the previous model, it allows the user more control over the degree distribution as well as the size distribution of the communities. This model has been extended to generate directed graphs with possibly overlapping community structures as well [25].

1.4 SCD - A Key Graph Kernel

Community detection is a widely studied area of research. Researchers have chosen multiple approaches to tackle this problem. A few popular ones involve using spectral techniques [36], graph sparsification [5, 44], traversals [42, 4], and greedy optimization of quality measures like modularity [7]. For a more comprehensive review of existing methods, see [15, 16].

For this report, two spectral clustering techniques are now described in detail.

The core premise of spectral clustering is that the eigenvectors of the matrices associated with a graph encode local information which can be used for clustering the nodes. The advantages of the spectral clustering methods come from their efficiency and mathematical elegance. Additionally, they usually have provable bounds of the quality of clusters produced. For a survey on spectral graph clustering methods, see [36].

The nodes and edges in a graph are described in an abstract space where the conventional notion of distance between objects does not apply. This is unlike a metric space, where each object is embedded in d-dimensions. Conventional machine learning tasks like classification or clustering expect the input data to be in a metric space, so they cannot be directly used for data represented as graphs. Spectral clustering, however, generates a d-dimensional metric space embedding of the nodes, i.e., each node gets assigned a d-dimensional coordinate. In addition to that, it ensures that the nodes that share direct links, or who are part of the same cluster, are spatially closer too. This results in the transfer of the link and community information from the abstract space to the metric space. In the bipartition algorithm described in Section 1.4.1, each node is embedded in 1-dimensional (metric) space, while in the algorithm in Section 1.4.2, it is k-dimensional.

1.4.1 Spectral bipartition

Fiedler [14] described how the eigenvector corresponding to the second smallest positive eigenvalue of the Laplacian matrix, known as the Fiedler vector, can be used to find an approximation for the graph bipartitioning problem.

Hagen et al. [18] proposed an algorithm whose pseudocode is given in Algorithm 1. Nodes are divided into two clusters p_1 and p_2 depending on whether the corresponding entry in the Fiedler vector is above or below the given threshold r. The choice of r therefore influences the quality of clusters. Popular choices include 0 and the median value of the Fiedler vector.

The computation of the Fiedler vector dominates the computational complexity of the algorithm. The fastest known method, the Lanczos method [28] takes linear time i.e., $O(|V|+|E|)$. So, the overall time complexity of Algorithm 1 is also $O(|V|+|E|)$.

1.4.2 k-way spectral partition

Ng et. al [40] extends the idea of bipartitioning described above into k-way partitioning as follows. Instead of using just the Fiedler vector, they use the k smallest non-trivial eigenvectors. Additionally, each row of the eigenvectors is normalized by its L_2 norm. By doing so, they generate a k-dimensional embedding for each of the nodes. These embeddings are then clustered using any conventional spatial clustering algorithm like K-means [19] to find k clusters. The pseudocode of this algorithm is given in Algorithm 2.

The running time of the algorithm is dominated by the eigendecomposition and the time taken by K-means to converge. In practice, the method seems to work fast and scales linearly with the size of the graph.

1.4.3 Metrics and Quality Measures

Erdös-Rényi(ER) graphs [13] where the process of edge formation is entirely random, serves as an essential baseline when it comes to clustering algorithms. Following the intuition provided in Section 1.1 behind cluster formation, community structure should be absent in ER graphs. However, as spatial clustering techniques identify spurious clusters in randomly generated data, community detection algorithms also fall into the same trap when running on ER graphs. This potentially defeats the purpose of extracting meaningful clusters from any graph. Therefore, being able to quantify the performance of a community detection for a given graph is essential.

1.4.3.1 Quality Measures

Given a network and a clustering of nodes, the quality measures quantify how *good* is the clustering, by assigning a score. This score allows for performance comparisons across runs of a community detection algorithm, or among multiple algorithms run on the same network. However, there is no universal notion of goodness in this context. Researchers have proposed several such measures, each with its caveats. The measures that are of importance to us in the context of this report are conductance [21] and modularity [38].

For the problem of bipartition, i.e., splitting a graph into two nearly balanced disjoint sets of nodes such that the connections between the sets are minimized, conductance is a good choice. The mathematical formulation follows this intuition. For a cluster $C \in \mathbb{C} = \{C_1, C_2\}$, the conductance score $\Phi(C)$ is defined as follows.

$$\Phi(C) = \frac{\mathrm{cut}(C)}{\min\{\mathrm{vol}(C),\ \mathrm{vol}(V \setminus C)\}}$$

where $\mathrm{vol}(S)$ is the sum of degrees of nodes in S. The numerator of Φ counts the edges that span from C to the rest of the graph while the denominator ensures the fairness of the split. So, a smaller ratio is indicative of a good split, since it implies the numerator is small and the denominator is large. One of the drawbacks of using conductance is that it is unsuitable for scoring multi-way partitions when the graph is split into more than two clusters.

Modularity, on the other hand, is more suitable for scoring k-partitions. The rationale behind stems from the idea of the deliberate and non-random nature of link formation behind community formation. For every cluster $C \in \mathbb{C} = \{C_1, \cdots, C_k\}$, it computes the difference of the fraction of the number of edges in C and the expected fraction if edges were distributed randomly. Mathematically, the modularity score of a partition $Q(\mathbb{C})$ is

$$Q(\mathbb{C}) = \sum_{C \in \mathbb{C}} \left[\frac{l_C}{m} - \left(\frac{k_C}{2m} \right)^2 \right]$$

where l_C and k_C is the number of edges inside and the sum of degrees of nodes in C, and m is the number of nodes in the graph. A higher modularity represents a better clustering since it means that the actual fractions of edges lying inside the communities is more than the expected values.

1.4.3.2 Comparing Partitions

In certain cases, being able to compare the clusterings in a more granular scale is advantageous over computing aggregate scores. For benchmark graphs, ground truth cluster assignments are available. So, the effectiveness of a clustering algorithm can be judged by comparing how closely the extracted clustering resembles the actual clustering. Borrowing ideas from information theory, one popular method is to compute the Mutual Information between the two clusterings [12], with higher scores signifying more similarity. Sometimes, the metadata can be used to guide the process of finding communities [41] to produce more meaningful clusters.

Algorithm 1 Approximate minimum cut of a connected graph G for a given threshold r

procedure `approx_min_cut`($G(V, E)$, r)
    `clusters` $\leftarrow \emptyset$
    if G has fewer than 2 nodes **then**
        `clusters` $\leftarrow V$
    else
        `fiedler` $\leftarrow$ Fiedler vector of G
        $\mathtt{p}_1 \leftarrow$ nodes ids with value less than r in `fiedler`
        $\mathtt{p}_2 \leftarrow$ rest of the nodes in G
        `clusters` $\leftarrow$ `clusters` $\cup \{ \mathtt{p}_1 \}$
        `clusters` $\leftarrow$ `clusters` $\cup \{ \mathtt{p}_2 \}$
    return `clusters`

Algorithm 2 k-way spectral partitioning of a connected graph G

procedure `k_way_spectral`($G(V, E)$, k)
    `clusters` $\leftarrow \emptyset$
    if G has fewer than k nodes **then**
        `clusters` $\leftarrow V$
    else
        `L` $\leftarrow$ Laplacian matrix of G
        `evecs` $\leftarrow$ k-smallest non-trivial eigenvectors of `L`
        Normalize each row of `evecs` by its norm
        Run `K-means` clustering on `evecs` to find k clusters $\mathbb{C} = \{C_1, \cdots, C_k\}$ using Euclidean distance
        `clusters` $\leftarrow \mathbb{C}$
    return `clusters`

1.5 Prior and Related Work

Spectral clustering remains a popular choice for finding clusters for reasons mentioned in Section 1. While the core premise of using eigendecompositions to encode structural similarity is shared across all the methods, each algorithm has its own uniqueness built into it and sees applications in a variety of domains, like computer vision [45, 33] and VLSI design [2]. In each of these methods, some variant of the minimum-cut problem is solved. Additionally, there have been methods like [39] which maximizes the modularity of the partitions. For a more detailed overview of spectral clustering techniques, see [36].

1.6 A Sequential Algorithm

For implementing Algorithms 1 and 2, we use Python with `NetworkX`[1], `numpy`[2], `scipy`[3], and `scikit-learn`[4] libraries. `NetworkX` provides easy to use containers for graphs as well as supporting functions like computing the Laplacian matrix and the Fiedler vector of a graph. `SciPy` facilitates easy computation of eigenvectors of arbitrary matrices and `scikit-learn` has a built-in implementation of the K-means algorithm.

1.7 A Reference Sequential Implementation

Python implementation of Algorithm 1

```
import networkx as nx

def approx_min_cut(G, r):
    assert nx.is_connected(G), "the graph must be connnected"

    clusters = []
    if G.order() < 2:
        clusters = list(G.nodes())
    else:
        # compute the Fiedler vector
        fiedler_vec = nx.fiedler_vector(G, method='lanczos')

        # p1 and p2 stores the nodes in each partition
        p1, p2 = set(), set()

        for node_id, fiedler_val in zip(G.nodes(), fiedler_vec):
            if fiedler_val < r:
                p1.add(node_id)
            else:
                p2.add(node_id)

        clusters.append(p1)
        clusters.append(p2)
    return clusters
```

Python implementation of Algorithm 2

```
import networkx as nx
import numpy as np
import scipy.sparse.linalg
from sklearn.cluster import KMeans
import sklearn.preprocessing

def k_way_spectral(G, k):
    assert nx.is_connected(G), "the graph must be connnected"

    clusters = []
```

[1] `https://networkx.github.io`
[2] `http://www.numpy.org`
[3] `https://www.scipy.org`
[4] `http://scikit-learn.org`

```
    if G.order() < k:
        clusters = list(G.nodes())
    else:
        L = nx.laplacian_matrix(G)

        # compute the first k + 1 eigenvectors
        _, eigenvecs = scipy.sparse.linalg.eigs(L.asfptype(), k=k+1, which='SM')

        # discard the first trivial eigenvector
        eigenvecs = eigenvecs[:, 1:]

        # normalize each row by its L2 norm
        eigenvecs = sklearn.preprocessing.normalize(eigenvecs)

        # run K-means
        kmeans = KMeans(n_clusters=k).fit(eigenvecs)
        cluster_labels = kmeans.labels_

        clusters = [[] for _ in range(max(cluster_labels) + 1)]

        for node_id, cluster_id in zip(G.nodes(), cluster_labels):
            clusters[cluster_id].append(node_id)
    return clusters
```

1.8 Sequential Scaling Results

Table 1.1: Average running time in seconds across 5 runs of algorithms 1 and 2 on graphs generated using the LFR benchmark using default parameters.

			k-way spectral		
$\|V\|$	$\|E\|$	approx min-cut	$k = 3$	$k = 4$	$k = 5$
100	788	0.018	0.349	0.259	0.068
1,000	7,339	1.556	1.867	1.464	2.051
10,000	62,029	5.997	3.757	8.998	6.109
100,000	765,073	60.654	71.142	72.926	71.33

The experiments are run on one node of the CRC cluster with 64 cores and 128 GB memory. The graphs are generated using the LFR benchmark using the default parameters ($\langle k \rangle = 16,\ \gamma = -2,\ \beta = -1,\ \mu = 0.1$) except for the number of nodes which are set to powers of 10.

The results are summarized in Table 1.1 and plotted on a log-log scale in Figure 1.1. The lines in Figure 1.1 are fairly straight, thus empirically the algorithms scale up linearly with the number of nodes as expected in Section 1.6.

1.9 Conclusion

This report covers the Spectral Community Detection kernel. It starts by looking at the causality behind the formation of communities, and then formalizes the problem in the language of graph

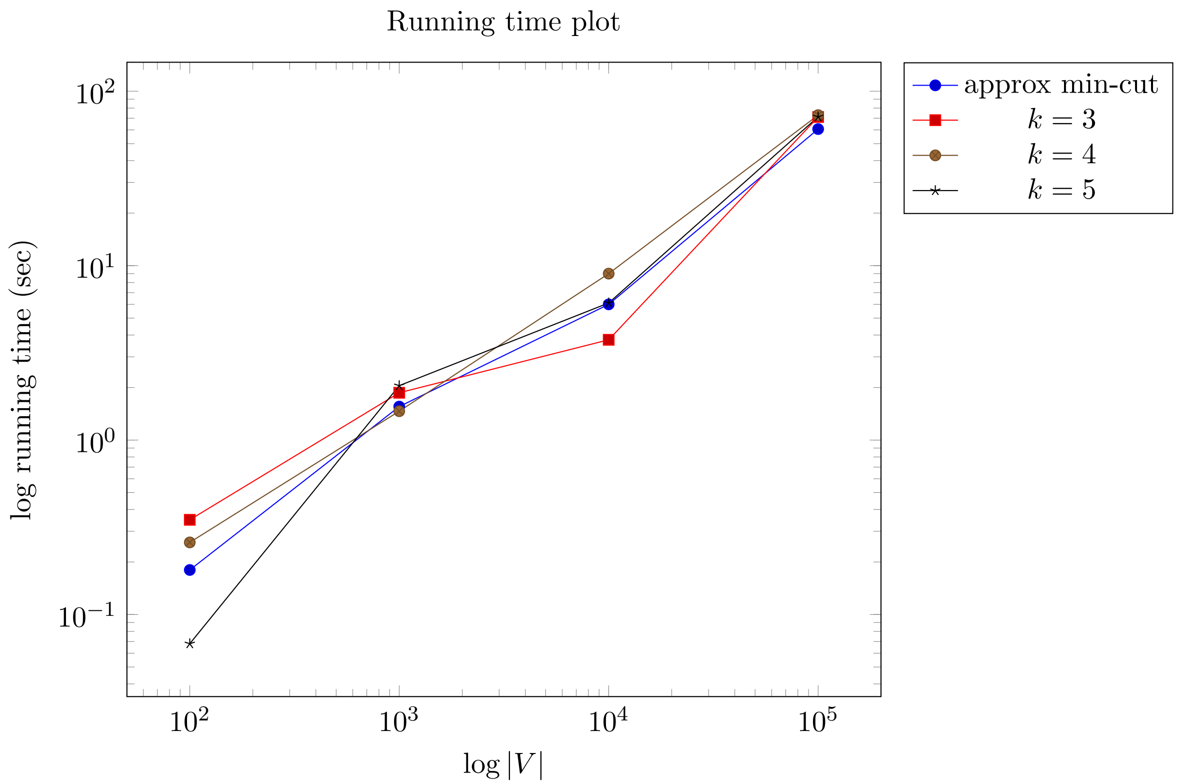

Figure 1.1: Running time plot on a log-log scale based on the data in Table 1.1

theory. It then discusses two popular algorithms, followed by their sequential implementation and scaling results.

Future work is summarized below.

- Testing the running times on more benchmark and real-world networks.
- Testing the quality of partitions using the measures described in Section 1.4.3.

1.10 Response to Reviews

- Added explanation for link formation in Section 1.1
- Added explanation for a hub.
- Added a paragraph in Section 1 explaining what embeddings are.
- Corrected the error in line 7 of algorithm 1.
- Added a citation for K-means instead of explaining it in short.

Bibliography

[1] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005.

[2] Charles J Alpert, Andrew B Kahng, and So-Zen Yao. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90(1-3):3–26, 1999.

[3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[4] Partha Basuchowdhuri, Satyaki Sikdar, Varsha Nagarajan, Khusbu Mishra, Surabhi Gupta, and Subhashis Majumder. Fast detection of community structures using graph traversal in social networks. *Knowledge and Information Systems*, pages 1–31, 2017.

[5] Partha Basuchowdhuri, Satyaki Sikdar, Sonu Shreshtha, and Subhashis Majumder. Detecting community structures in social networks by graph sparsification. In *Proceedings of the 3rd IKDD Conference on Data Science, 2016*, page 5. ACM, 2016.

[6] Ginestra Bianconi, Richard K Darst, Jacopo Iacovacci, and Santo Fortunato. Triadic closure as a basic generating mechanism of communities in complex networks. *Physical Review E*, 90(4):042806, 2014.

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

[8] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. One trillion edges: Graph processing at facebook-scale. *Proceedings of the VLDB Endowment*, 8(12):1804–1815, 2015.

[9] Nicholas A Christakis and James H Fowler. *Connected: The surprising power of our social networks and how they shape our lives.* Little, Brown, 2009.

[10] Aaron Clauset, Ellen Tucker, and Matthias Sainz. The colorado index of complex networks. `https://icon.colorado.edu/`, 2016. Accessed: 2018-09-10.

[11] Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18(2):116–140, 2001.

[12] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[13] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[14] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

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

[16] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[17] Mark S Granovetter. The strength of weak ties. In *Social networks*, pages 347–367. Elsevier, 1977.

[18] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

[19] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[20] Justus Haucap and Ulrich Heimeshoff. Google, facebook, amazon, ebay: Is the internet driving competition or market monopolization? *International Economics and Economic Policy*, 11(1-2):49–61, 2014.

[21] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004.

[22] Nikhil S Ketkar, Lawrence B Holder, and Diane J Cook. Subdue: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 71–76. ACM, 2005.

[23] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. Summarizing and understanding large graphs. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(3):183–202, 2015.

[24] Jérôme Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350. ACM, 2013.

[25] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[26] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[27] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[28] Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators.* United States Governm. Press Office Los Angeles, CA, 1950.

[29] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[30] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[31] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041*, 2014.

[32] Junnan Lu and Alex Thomo. An experimental evaluation of giraph and graphchi. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 993–996. IEEE Press, 2016.

[33] Bin Luo, Richard C Wilson, and Edwin R Hancock. Spectral embedding of graphs. *Pattern recognition*, 36(10):2213–2230, 2003.

[34] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[35] James Moody. Race, school integration, and friendship segregation in america. *American journal of Sociology*, 107(3):679–716, 2001.

[36] Maria CV Nascimento and Andre CPLF De Carvalho. Spectral methods for graph clustering–a survey. *European Journal of Operational Research*, 211(2):221–231, 2011.

[37] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the national academy of sciences*, 98(2):404–409, 2001.

[38] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[39] Mark EJ Newman. Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4):042822, 2013.

[40] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[41] Leto Peel, Daniel B Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science advances*, 3(5):e1602548, 2017.

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

[43] Sherif Sakr, Faisal Moeen Orakzai, Ibrahim Abdelaziz, and Zuhair Khayyat. *Large-Scale graph processing using Apache giraph*. Springer, 2016.

[44] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 721–732. ACM, 2011.

[45] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[46] Reynold S Xin, Joseph E Gonzalez, Michael J Franklin, and Ion Stoica. Graphx: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, page 2. ACM, 2013.