Global Graph Clustering and Local Graph Exploration for Community Detection in Twitter

Christos Karras*, Aristeidis Karras*, Ioanna Giannoukou†,
Konstantinos C. Giotopoulos†, Dimitrios Tsolis*, Spyros Sioutas*
*Decentralized Systems Computing Group, Computer Engineering and Informatics Department, University of Patras, Greece
{c.karras, akarras, sioutas}@ceid.upatras.gr, dtsolis@upatras.gr
†Department of Management Science and Technology, University of Patras, Greece
{igian, kgiotop}@upatras.gr

2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM) | 979-8-3503-9858-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SEEDA-CECNSM57760.2022.9933012

***Abstract*—In this paper, the concepts and techniques for global graph clustering are examined, or the process of locating related clusters of vertices within a graph. We introduce the construction of a graph clustering technique based on an eigenvector embedding and a local graph clustering method based on stochastic exploration of the graph. Then, the developed implementations of both methods are presented and assessed in terms of performance. In addition, the difficulties associated with assessing clusterings and benchmarking cluster algorithms are explored where PageRank and EigEmbed algorithms are utilized. The experiments show that the EigEmbed outperformed PageRank across all experiments as it detected more communities with the same number of clusters. Ultimately, we apply both algorithms to a real-world graph representing Twitter network and the followers and tweets therein.**

***Index Terms*—Graph Clustering, Stochastic Processes, PageRank, Community Detection, Intelligent Innovation, Social Networks**

I. Introduction

Graph clustering, the segmentation of a graph into groups of similar nodes, has come to prominence during the past few decades as the availability of enormous data sets in consumer and social networks has proliferated. It has become a useful tool for recognising communities in social networks, discovering structure in high-dimensional data sets, and constructing complex recommendation systems [1]–[5]. One specific definition of similar nodes is to split graph vertices into sets with low conductance, as given below. Intuitively, one might conceive of the conductance of a set S as the likelihood of migrating from a random node in S to a node outside S. Lower conductance consequently implies a tighter community inside the graph and may serve as a measure of quality for clustered communities. Unfortunately, selecting partitions of a graph G which minimise the conductance of the clusters is a non-deterministic polynomial-time (NP-hard) problem, although there exist approximation techniques with theoretical guarantees. In the context of this paper, two such approximation ideas are introduced and covered thoroughly.

Spectral graph clustering is a global technique that utilises the eigenvectors and eigenvalues of the graph Laplacian to discover groupings of well-connected nodes. There are several approaches to perform this task using the spectral information of the Laplacian. For example, the eigenvector associated with the lowest non-zero eigenvalue of the Laplacian of a connected network defines a near-optimal low conductance cut. This results in an iterative technique that splits the graph into several clusters, and theoretical recovery guarantees may be obtained based on the conductance between clusters. Alternately, one may utilise a subset of the eigenvectors of the Laplacian corresponding to the lowest eigenvalues to embed the graph in a Euclidean space, after utilizing traditional distance-based clustering algorithms such as k-means as in [6] [7]. Moreover, sampling methods as in [8]–[10] can be utilized effectively for graph clustering along with stochastic optimization schemes [11] [12] for community detection. The method of spectral clustering has intriguing ties to optimization algorithms such as mincut, ratiocut, and ncut, depending on the normalisation scheme selected for the Laplacian, which picks clusters that minimise an acceptable concept of inter-cluster connectedness.

The embedding of eigenvectors is also related to random walks on graphs. For example, the distance among nodes in an eigenvector embedding is highly correlated with the predicted commuting time between the two nodes. In II-A, the eigenvector embedding technique is highlighted, namely the algorithm developed by Shi and Malik in [13]. For very vast graphs, such as those seen in web-browsing and social network applications, it may be too costly to directly approximate them using the Laplacian. Instead of computing eigenvectors of the Laplacian, local clustering approaches explore local structure to recover well-connected communities in a network.

In Section II-B, the local approach established in [14] is investigated, where customised PageRank vectors are used to identify clusters with low conductance for approximation. The objective here is to get eigenvector-like information via local graph walks while the embedding of eigenvectors specifies predicted commuting durations among nodes. This correlation drives us to examine local network structures for eigenvectors approximation by using locally biased random walks.

The objective of a local technique is to outperform global methods in terms of execution speed, particularly for sparse graphs. Approximate Personalized PageRank (APPR) fulfills

this by identifying clusters with theoretical guarantees by spreading mass from a seed set. Rather than simply constructing quick sparse linear algebra calculations, the approaches presented here concentrate on strategies to eliminate matrix operations.

The remainder of this paper is organized as follows. In Section II-A, the eigenvector embedding strategy and the implementation of a spectral clustering algorithm is introduced. In Section II-B, a local method based on the techniques in [14] [15] is surveyed and implemented. In Sections III and IV, the performance of the implementation on the generative planted partition model is presented and its use in a real-world application: community detection in a social network. Ultimately, the concluding remarks are in Section V with a few observations on the performance and implementation of the global and local methods.

The notation utilized in this work is as follows. An unweighted graph with node set V and edge set E is denoted $G = (V, E)$. The degree of a node $v \in V$ is written $d(v)$. The adjacency matrix and degree matrix are A and D, respectively. The complement of a subset of vertices S in V is denoted S^C. The concepts of cut, conductance, and volume throughout are also utilized. The cut, volume, and conductance of a subset of nodes $S \in V$ are defined as:

$$\text{cut}(S) = \#\text{ edge endpoints leaving } S = \sum_{e_{ij} \in E, v_i \in S, v_j \in S^C} \tag{1}$$

$$\text{vol}(S) = \#\text{ edge endpoints in } S = \sum_{u \in S} d(u) \tag{2}$$

$$\Phi(S) = \frac{\text{cut}(S)}{\min(\text{vol}(S), \text{vol}(V \backslash S))} \tag{3}$$

II. Preliminaries

A. Global spectral clustering

Eigenvector Embedding: To represent the eigenvector embedding, consider the problem of partitioning the vertices of a graph $G = (V, E)$ into two distinct clusters, $C \subset V$ and its compliment $C^C \subset V$, such that the conductance $\Phi(C)$ is a minimum over all such partitions. Formally, the aim here to solve the ncut (normalized cut) program.

$$C = \min_M \Phi(M). \tag{4}$$

Before doing so, it is helpful to supply an overview of the connection between ncut and the eigenvectors of the graph Laplacian. As it can be seen below, the minimization problem can be represented as a constrained minimization of the Rayleigh Quotient (RQ) of the Laplacian $L = D - A$. The minimization of the RQ is over structured indicator vectors which store the partition information in M. The indicator constraint makes the minimization problem NP-hard, so it is feasible to consider relaxing this to minimize over a true subspace of real valued vectors with dimension n. This leads to a classical minimization problem in linear algebra, the minimization of the RQ, whose solution is provided by certain eigenvectors of L.

Note that the shape of the Laplacian is quadratic of form $u^T L u = \sum_{i,j=1}^{n} a_{ij}(u_i - u_j)^2$, where u^T denotes the transpose of u. Here, $L \in \mathbb{R}^{n \times n}$, $u \in \mathbb{R}^n$, and a_{ij} are the entries of the adjacency matrix A. To express (4) a characteristic vector f is defined component-wise by (5).

$$f_i = \begin{cases} \sqrt{\frac{vol(M^C)}{vol(M)}} & i \in M, \\ -\sqrt{\frac{vol(M)}{vol(M^C)}} & i \in M^C. \end{cases} \tag{5}$$

The partitioning information is encoded in the characteristic vector f. The aim is to solve (4) by determining an f that minimizes the conductance of M. The key observations in this reformulation are that the conductance $\Phi(M)$ is proportional to $f^T L f$ and that f is D-orthonormal to the first eigenvector of the Laplacian, the vector of ones $\mathbf{1} = (1, \cdots, 1)^T$. Below the results of a few simple calculations where (6) holds are shown.

$$f^T L f = 2\,\text{vol}(V)\Phi(M), \quad (Df)^T \mathbf{1} = 0, \text{ and } f^T D f = \text{vol}(V) \tag{6}$$

The objective and constraints in terms of f and the Laplacian L, can be expressed as in (7).

$$\begin{gathered} \min_M f^T L f, \\ \text{s.t. } f \text{ as in } (5), \text{ with} \\ (Df)^T \mathbf{1} = 0, f^T D f = \text{vol}(V). \end{gathered} \tag{7}$$

As the lowest eigenvalue of the Laplacian is always zero and the corresponding eigenvector is $\mathbf{1}$, this optimization problem now appears strikingly similar to the minimization of the RQ for the generalized eigenvalue problem $Lx = \lambda Dx$, however, the constraint on the entries of f make this problem NP-hard. The strategy is to relax the constraint on f so that its entries are real-valued, in which case (7) is minimized by the eigenvector x of $D^{-1}L$ corresponding to the second lowest eigenvalue (assuming G is connected, D is invertible and the generalized eigenvalue problem $Lx = \lambda Dx$ is equivalent to $D^{-1}Lx = \lambda x$).

Formerly, the j^{th} vertex was within cluster M if the sign of the j^{th} entry of f was positive, and in the cluster M^C if the sign was negative.Based on the items in v, nodes to clusters can be reassigned in order to retrieve the partition information. Alternately, the items of v may be grouped using a more complicated approach. Methods such as k-means may be more resilient for dividing V into more than two clusters in this situation.

To partition into $k > 2$ clusters, a similar analysis can be carried out. The objective function is modified so that (4) is replaced by (8).

$$(C_1, \cdots, C_k) = \min_{(M_1,\cdots,M_k)} \sum_{i=1}^{k} \Phi(M_i). \tag{8}$$

The j^{th} cluster at this point has an indicator vector defined by (9).

$$h_{i,j} = \begin{cases} \sqrt{\frac{1}{vol(M_i)}} & i \in M_j \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

If H is denoted as the matrix with entries $h_{ij} = h_{i,j}$, then the minimization problem (8) is equivalent to minimizing trace$(H'LH)$ subject to $H^T DH = I$, over H as in (9). If this last constraint on H is relaxed so that the entries of H are real-valued, a classical trace minimization problem is obtained. The solution is given by the matrix $X \in \mathbb{R}^{n\times k}$ whose columns are the k eigenvectors of the generalized eigenvalue problem $LX = \Lambda DX$ that correspond to the k lowest eigenvalues after the first zero eigenvalue.

An approach for spectral clustering that attempts to reduce conductance between groups is proposed in Algorithm 1.

Algorithm 1 `SpectralClustering`(G, k)

Require: Graph G, number of clusters k.
Ensure: $\hat{C}_j = \{v_i \in V : y_i \in P_j\}$, for $j = 1, \cdots k$.

1: Compute degree matrix $D \in \mathbb{R}^{n\times n}$ and graph Laplacian $L \in \mathbb{R}^{n\times n}$ associated with graph G.
2: Compute first $k+1$ eigenvectors, $x_0, \cdots, x_k$ of generalized eigenvalue problem $Lx = \lambda Dx$.
3: Form matrix $X \in \mathbb{R}^{n\times k}$ whose i^{th} column is x_i, $1 \le i \le k$, and let $y_j \in \mathbb{R}^k$ be the j^{th} row of X.
4: Use the k-means algorithm to partition the points $\{y_j\}_{j=1}^n$ in $\mathbb{R}^k$ into clusters $P_1, \cdots, P_k$.

Due to the relaxation of the restriction on the characteristic vectors, it cannot be guaranteed that the clusters recovered by dividing the vertices in the eigenvector embedding would minimize the objective function (8). Specifically, there are no assurances that the conductance of these clusters is comparable to that of genuine minimizers. However, the eigenvector issue may be handled using ordinary linear algebra methods, and clustering is often reported to function rather effectively in practise. The relationship between the distance between nodes in the eigenvector embedding and other concepts of connectedness inside a graph justifies the relaxing and reassignment procedure. Specifically, the distance between vertices v_i and v_j in the above-described eigenvector embedding is comparable to the predicted commuting time between v_i and v_j in a random walk on G. Consequently, one intuitive interpretation of clusters provided by Algorithm 1 is that a random walk or diffusion process starting in any of these clusters would prefer to remain inside the cluster for an extended amount of time relative to the frequency of jumps between clusters. Different normalisation techniques for the Laplacian provide analogous linkages with other optimization methods, such as `ratiocut` and `mincut`, which involve modifying the objective function in (4).

Implementation: For the implementation, an identical variant of Algorithm 1 is proposed. Rather than solving the generalized eigenvalue problem $Lx = \lambda Dx$, the computation of the eigenvectors $\hat{v}_1, \cdots, \hat{v}_k$ of the symmetric normalized Laplacian $L_s = D^{-1/2}LD^{-1/2}$ occurs. The eigenvectors $v_1, \cdots, v_k$ of $D^{-1}L$ are then obtained by the scaling $v_i = D^{-1/2}\hat{v}_i$ for each $1 \le i \le k$.

To calculate the eigenvectors of L_s, the sparse hermitian eigensolver `eigsh` is utilized which is available in `scipy.sparse.linalg`. The eigensolver employs the implicitly restarted Lanczos method (IRLM) supported by `ARPACK`, which may be viewed as repeated applications of a truncated, implicitly shifted QR algorithm to the tridiagonal matrix produced by the Lanczos factorization. Dense matrices are anticipated to have a complexity that scales roughly as $\mathcal{O}(n^2)$ due to the matrix-vector product being the major cost inside each iteration. The predicted complexity scaling for Laplacians recorded in sparse format with a limited number of non-zero values is approximately $\mathcal{O}(n)$.

Alternative spectral clustering

As stated in Section I, there is a version of spectral clustering that uses sweeps over the eigenvector corresponding to the second lowest eigenvalue to find low conductance cuts with theoretical guarantees [16]. This results in an iterative approach for clustering the graph into numerous groups. This spectral clustering technique provides conductivity guarantees for recovered clusters and is substantially connected to the Local PageRank clustering approach described in Section II-B.

B. Local PageRank clustering

This section surveys the exposition in [14], which describes the Approximate Personalized PageRank (APPR) algorithm for clustering graphs. APPR improves on a prior Nibble method (see [15]) that used a sequence of random walk vectors. A PageRank vector can be described recursively, so APPR can consider a single PageRank vector in place of a sequence of random walk vectors while retaining theoretical guarantees.

Personalized PageRank is defined as the unique solution to (10).

$$\text{pr}(\alpha, s) = \alpha s + (1-\alpha)\text{pr}(\alpha, s)W, \tag{10}$$

where $\alpha \in (0, 1]$ is a teleportation constant, s is an initial concentrated distribution over the nodes (in our case, concentrated on a single node), and $W = \frac{1}{2}(I + D^{-1}A)$ is the lazy random walk transition matrix. Note that distribution vectors are written as row vectors and so multiplied to the left of W. The customised PageRank defines the distribution of nodes from a random walk where the user is transported back to the beginning node s with probability α at each step. This biases the walk such that local graph structure is explored without

traversing too far. Note that the global PageRank metric may be represented as shown above with s representing the uniform distribution on nodes.

For any α, the PageRank vector can be represented as a linear transformation of s: $\text{pr}(\alpha, s) = sR_\alpha$, where (11) holds.

$$R_\alpha = \alpha \sum_{t=0}^{\infty} (1-\alpha)^t W^t. \tag{11}$$

This means that a PageRank vector is a weighted average of lazy random walk vectors in the form of (12).

$$\text{pr}(\alpha, s) = \alpha \sum_{t=0}^{\infty} (1-\alpha)^t (sW^t). \tag{12}$$

Computing approximate PageRank

An approximate PageRank vector $\text{apr}(\alpha, s, r)$ is defined in terms of a non-negative residual vector r that satisfies the condition (13).

$$\text{apr}(\alpha, s, r) + \text{pr}(\alpha, r) = \text{pr}(\alpha, s). \tag{13}$$

The following proposition, which is similar but distinct from the definition of PageRank, allows an iterative algorithm to compute approximate PageRank vectors as defined in (14).

$$\text{pr}(\alpha, s) = \alpha s + (1-\alpha)\text{pr}(\alpha, sW). \tag{14}$$

Proof. The linear transformation R_α can be written recursively as in (15).

$$R_\alpha = \alpha \sum_{t=0}^{\infty} (1-\alpha)^t W^t = \alpha I + (1-\alpha) W R_\alpha. \tag{15}$$

The result is derived by applying R_α to the starting distribution s:

$$\begin{aligned} \text{pr}(\alpha, s) &= sR_\alpha \\ &= \alpha s + (1-\alpha) sW R_\alpha \\ &= \alpha s + (1-\alpha)\,\text{pr}(\alpha, sW) \end{aligned} \tag{16}$$

□

An algorithm for computing approximate PageRank is constructed by step-wise 'pushing' mass from a residual vector r to the associated approximate PageRank vector p. Initially, the value of p is set to $p = 0$ and all mass are put in r on the starting node–i.e., $r = \chi_v$ where

$$\chi_v(u) = \begin{cases} 1 & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}.$$

Then, a number of push operations based on 14 is applied to spread mass from a single node u. Therefore, α fraction of $r(u)$ is moved to $p(u)$, representing the teleportation move, and spread the remaining $(1-\alpha)$ share within r based on single step transitions of the lazy random walk from u. Throughout each push, p maintains the approximate PageRank equation $P + \text{pr}(\alpha, r) = \text{pr}(\alpha, s)$. This 'push' subroutine is described formally in Algorithm 2.

Algorithm 2 $\texttt{push}_u(G, p, r)$

Require: Graph G, node u, approximate PageRank p, corresponding residual r
Ensure: Updated PageRank p' and corresponding residual r'

Let $p' = p$, $r' = r$, then make the following changes
$p'(u) = p(u) + \alpha r(u)$
$r'(u) = (1-\alpha) r(u)/2$
for v s.t. $(u, v) \in E$ **do**
    $r'(v) = r(v) + (1-\alpha) r(u)/(2d(u))$
end for

After performing a large number of pushes, the residual r (or, more precisely, the degree-normalized version of r) becomes minimal, making p an excellent approximation of the PageRank vector. This is specified by the Algorithm 3 where Proposition 1 limits the required number of pushes.

Algorithm 3 $\texttt{ApproximatePageRank}(G, v, \alpha, \varepsilon)$

Require: Graph G, starting node v, $\alpha \in (0, 1]$, ε
Ensure: $p = \text{apr}(\alpha, \chi_v, r)$ with $\max_{u \in V} \frac{r(u)}{d(u)} < \varepsilon$

Let $p = 0$, $r = \chi_v$
while $\max_{u \in V} \frac{r(u)}{d(u)} \geq \varepsilon$ **do**
    Choose any vertex u where $\frac{r(u)}{d(u)} \geq \varepsilon$
    $p, r = \texttt{push}_u(G, p, r)$
end while

Proposition 1. Let T be the total number of push operations performed by ApproximatePageRank and let d_i be the degree of the vertex used in the ith push. Then (17) holds.

$$\sum_{i=1}^{T} d_i \leq \frac{1}{\varepsilon \alpha}. \tag{17}$$

Proof. Note that the amount of probability on the vertex pushed at time i is at least εd_i. Then $|r|_1$ decreases by at least $\alpha \varepsilon d_i$ over the ith push. Initially, $|r|$ is set to $|r|_1 = 1$, so inequality (18) holds.

$$\alpha \varepsilon \sum_{i=1}^{T} d_i \leq 1. \tag{18}$$

□

Based on the preceding proposition introduced in Prop. 1, ApproximatePageRank(v, α, ε) runs in time $\mathcal{O}\left(\frac{1}{\varepsilon\alpha}\right)$. To accomplish this temporal complexity, the system must maintain a queue that contains all push-eligible vertices. When an element of r is modified during a push, if required, the queue is updated.

Finding low conductance cuts

To find cuts with low conductance, a search over sweep sets of the PageRank vector occurs. This is similar to a variant of the traditional spectral clustering, as described at the conclusion of the preceding section, in which a sweep over an eigenvector is guaranteed to generate a cut with conductance close to the global minimum. In the PageRank scenario, the cut generated by the sweep relies on the initial vertex v and the teleportation constant α.

Given the distribution p with support $\text{supp}(p) = N_p$, let $v_1, \ldots, V_{N_p}$ be an ordering of vertices such that:

$$\frac{p(v_i)}{d(v_i)} \geq \frac{p(v_{i+1})}{d(v_{i+1})} \tag{19}$$

The jth sweep set is defined to be $S_j^p = \|v_1, \ldots, v_j\}$ for each $0 \leq j \leq N_p$. Given our approximate PageRank vector p, a search for the sweep set with minimum conductance is the aim, which can be found by sorting p and computing conductance of each sweep set in time $\mathcal{O}\Big(\text{vol}(\text{supp}(p)) \log n\Big)$. Notice that this involves searching over at most n sets as opposed to the very naive search over all subsets of V.

A mixing result for PageRank vectors provides theoretical guarantees (see [14]). Specifically, if a sweep over a pagerank vector does not yield a cut with low conductance, then the PageRank vector is near to the stationary distribution. Alternatively, if there is a set with a small conductance C and a large number of beginning vertices, the resultant PageRank vector is not near to being stable since it has a substantially higher probability inside C. This results in a localised form of the Cheeger inequality for PageRank vectors (a similar Cheeger-like inequality guarantees that a sweep over the second eigenvector of the Laplacian produces a cut with low conductance).

The sweep process is described formally in Algorithm 4 and can be implemented in time $\mathcal{O}\left(2^b \frac{\log^3 m}{\varphi^2}\right)$. Note that the `PageRank-Nibble` algorithm described in [14] puts further volume conditions on the sweep sets in order to provide control over the volumes of the partitions obtained. Experiments with this occurred by our side but decided to work with the more basic version of the algorithm.

Algorithm 4 $\texttt{PageRank} - \texttt{Nibble}(G, v, \varphi, b)$

Require: Graph G, node v, $\varphi \in [0,1]$, $b \in [1, B]$ where $B = \lceil \log_2 m \rceil$

1: Let $\alpha = \frac{\varphi^2}{225 \ln(100\sqrt{(m)})}$, $\varepsilon = 2^{-b}\frac{1}{48B}$
2: $p = \texttt{ApproximatePageRank}(G, v, \alpha, \varepsilon)$
3: Find set S_j^p for $j \in [1, |\text{supp}(p)|]$ with minimum conductance
4: **if** $\Phi(S_j^p) < \varphi$ **then return** S_j^p
5: **else return** none
6: **end if**

Since the success of `PageRank-Nibble` depends on a good choice of v and b, the partitioning algorithm will use a randomized version as given in Algorithm 5.

Algorithm 5 $\texttt{RandomPageRank} - \texttt{Nibble}(G, \varphi)$

Require: Graph G, conductance threshold $\varphi \in [0,1]$
Ensure: $\texttt{PageRank} - \texttt{Nibble}(G, v, \varphi, b)$

1: Choose a vertex v with probability $\Pr(v = i) = \frac{d(i)}{\text{vol}(V}$
2: Choose b in $1, \ldots, \lceil \log_2 m \rceil$ with probability $\Pr(b = i) = \frac{2^{-i}}{1 - 2^{-\lceil \log_2 m \rceil}}$

The partition algorithm is then described in Algorithm 6. If a low conductance set exists, this algorithm finds a set S with $\Phi(S) < \varphi$ with probability $1 - q$; Theorem 1 states this more formally. Note that $G(S)$ means the subgraph induced by S and $\text{vol}_S(T)$ means the volume of T in the subgraph induced by S. The expected runtime of `PageRank-Partition` is $\mathcal{O}\left(m \frac{\log^4 m}{\varphi^3} \log \frac{1}{q}\right)$.

Algorithm 6 $\texttt{PageRank} - \texttt{Partition}(G, \varphi, q)$

Require: Graph G, conductance threshold $\varphi \in [0,1]$, probability $q \in (0,1)$

1: Set $W_1 = V$
2: **for** $j = 1, \ldots, 56m\lceil \log_{10}(1/q) \rceil$ **do**
3: $\quad D_j = \texttt{RandomPageRank} - \texttt{Nibble}(G(W_j), \varphi)$
4: $\quad W_{j+1} = W_j \setminus D_j$
5: $\quad$ **if** $\text{vol}_{W_{j+1}}(W_{j+1}) \leq \frac{5}{6} 2m$ **then return** $D = V \setminus W_{j+1}$
6: $\quad$ **end if**
7: **end for**
8: **return** none

Theorem 1. If there is a set C with $\text{vol}(C) \leq \frac{1}{2}\text{vol}(G)$ and $\Phi(C) \leq \frac{\varphi}{1845000 \log^2 m}$, then `PageRank-Partition` produces a set S satisfying $\Phi(S) \leq \varphi$ and $\text{vol}(S) \leq \frac{5}{6}\text{vol}(G)$ with probability at least $1 - q$.

III. Methodology

In this section, we apply our global and local clustering algorithms to a graph of one of the authors' Twitter friends. These algorithms may be used to identify community structure and spheres of influence in a social network. Identifying these community structures may play a role in improved recommender and filtering systems. Our results are displayed in Table I. With 3 clusters, neither EigEmbed or PPRank is perfect, but each algorithm picks up on interesting features. In particular, they both find low conductance sets, many of which correspond well with ground truth clusters. With that being said, PageRank clustering tended to break up some groups that perhaps should not have been broken up. On the other hand, spectral clustering did not break up such groups, but left some groups too large that should have been broken down. When using a higher number of clusters (10 instead of 3), spectral clustering successfully breaks down these larger clusters.

IV. Experimental results

For the experiments the local and global algorithms are compared for recovering ground truth in the planted partition model. The planted partition model is a generalization of the Erdos-Renyi random graph model in which communities are planted within the node set. Each conceivable edge occurs/-does not occur as the result of a Bernoulli trial (independent of other edges); nonetheless, the probability of an edge between nodes in the same community is greater than that of an edge across communities.

Specifically, the employed planted partition model is detailed below. Let k represent the number of communities, and n_0 represent the number of nodes included inside each community. Thus, all communities have a size of n_0 and the graph has kn_0 nodes. Let p represent the probability of an edge between nodes belonging to the same community, and q represent the likelihood of an edge between nodes belonging to separate communities.

If $q = 0$, then the adjacency matrix of a planted partition graph is block diagonal with a block for each cluster. For $q < p$, the the adjacency matrix is more highly concentrated in the block diagonal than the off-block diagonal. The community mixing level which is shown in (20),

$$\mu = \frac{(k-1)q}{p + (k-1)q} \tag{20}$$

measures the expected fraction of edges that cross community boundaries. The value of μ represents the expected value of the conductance of the ground truth clusters, and so informs about reasonable values of the threshold φ. In our experiments, p is set to $p = 0.5$, $n_0 = 50$ and consider of $k = 2, 3, 4,$ and 5. The aim is to look at a range of q between 0.1 and 0.5 and use $\varphi = 1.1 * \mu$.

In Figure 1, the ability of spectral clustering and PageRank is compared for partitioning, to find low conductance clusters. Note also that, for the planted partition model, the variance in the cluster conductance serves as a measure of when the clusters returned by the spectral clustering algorithm are no longer reliable, as verified by comparison with the F1 scores for spectral clustering in Figure 2.

In Figure 2, the ability of spectral clustering and PageRank are compared for partitioning, to recover the ground truth clusters. This is measured using the average F1 score of all clusters. The F1 score of a cluster is defined as in (21).

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{21}$$

where precision is defined as in (22). and recall as in (23).

$$\text{precision} = \frac{\#\text{true positive}}{\#\text{true positive + false positives}} \tag{22}$$

$$\text{recall} = \frac{\#\text{true positives}}{\#\text{true positives + false negatives}} \tag{23}$$

To calculate the F1 score for a cluster, first there is a need to match a given recovered cluster to a ground truth cluster. The F1 score is calculated for each possible match and use the maximum score as the F1 score of the cluster.

The experiments shown in Figures 1 were conducted on the planted partition model with three 50 node communities. This situation was particularly intriguing since the spectral clustering technique (EigEmbed) correctly recovered clusters up to $q \approx .35$, yet the F1 score for clusters recovered by the local clustering approach (PPRank) was relatively low (see Figure 2). Nonetheless, as indicated by the conductance plots in Figure 1, PPRank recovered clusters with lower average conductance. In fact, although EigEmbed clusters 150 nodes into three balanced communities, PPRank consistently partitioned 150 nodes into two balanced communities and then further subdivided one of these communities, producing a lower mean conductance than the ground truth clusters.

For the two block scenario (See the left panel of Figure 2), PPRank recovers the clusters correctly when the probability q of edge connections between clusters is low, but the quality quickly degrades as q is raised. In both instances, spectral clustering preserves high-quality recovery until $q \approx .35$.

Figure 3 shows empirical time complexity on the planted partition model. Computation quickly became unwieldy for larger k using PageRank clustering. This is anticipated, as the `PageRank-Partition` time complexity depends on m, which is expected to scale with n^2 in planted partition. For spectral clustering one can observe an approximately quadratic time complexity as the graph size ranges from $nk = 100$ nodes to $nk = 1000$ nodes. This occurs because the dominant cost of implicitly restarted Lanczos algorithm, is the iterative application of matrix-vector products. As the number of non-zeros m scales with n^2 in the planted partition model with fixed q, the Laplacian becomes increasingly dense as k increases. This assessment supports the observed quadratic complexity.

In Figure 4, it is shown the empirical time complexity on sparse graphs generated from random trees. At this point, sparse graphs are considered since this should better reflect the strength of PageRank clustering. In this case, PageRank clustering is more competitive with spectral clustering but still increases faster as n increases. This might be reduced with a more optimized implementation. Finally, a summary of the Twitter graph after the experiments is given in Table I.

V. Discussion and Conclusions

In this section, we highlight a few observations made while using these strategies. Initially, we see that `PageRank-Partition` is easily parallelizable. However, Eigenvector computations are notoriously difficult to parallelize. While certain spectral projection-based eigensolvers, such as [17], are extremely parallelizable, they often need some localization of the required eigenvalues in the complex plane. This may be difficult to retrieve without

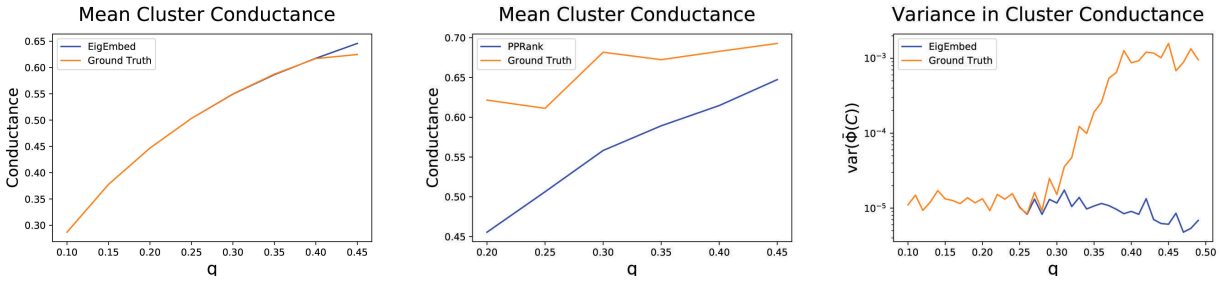

Fig. 1. Mean conductance of calculated clusters for **Left:** spectral clustering and **Middle:** local PageRank algorithms in the planted-partition model with 50 vertices and three communities. **Right:** Conductance variance as a sign of low cluster quality.

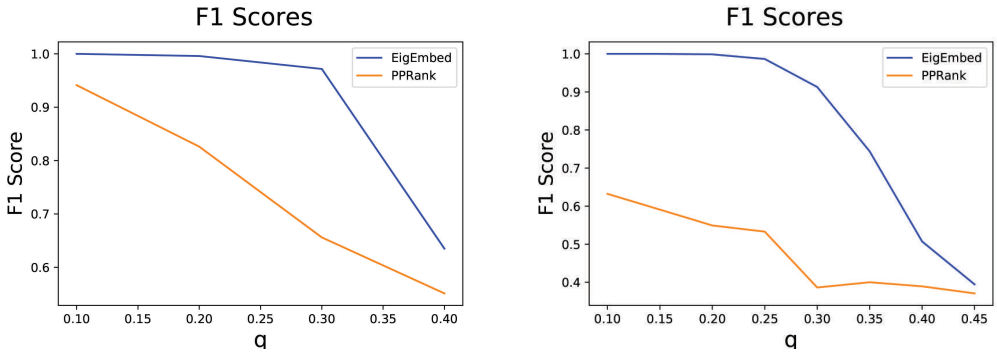

Fig. 2. F1 scores achieved by spectral clustering and local PageRank for the planted-partition model with **Left:** two communities of 50 vertices and **Right:** three communities of 50 vertices.

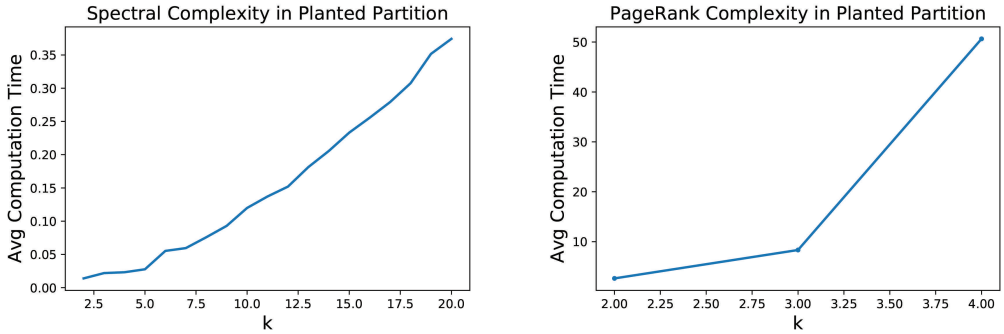

Fig. 3. Average computation time over 20 trials to split planted partition graphs into two cluster. Parameter values: $n_1 = 50, p = 0.5, q = 0.2, \varphi = \max(0.5, 1.2\mu)$ and k on the x-axis. **Left:** spectral clustering, **Right:** PageRank clustering (with `PageRank-Partition` input $q = 0.8$

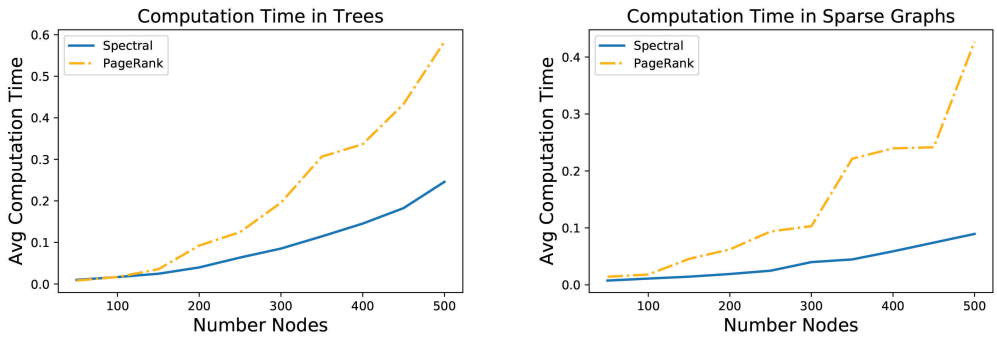

Fig. 4. Average computation time over 20 trials to split sparse graphs into two clusters. Sparse graphs generated using `NetworkX random_powerlaw_tree` with $\gamma = 3$. **Left:** single tree, **Right:** two tree edge sets added together. `PageRank-Partition` implemented with $\varphi = 0.5$, $q = 0.8$

TABLE I
RESULTS OF THE TWITTER GRAPH AFTER GRAPH EXPLORATION

Symbol	Statistic	PPRank	EigEmbed
$\|V\|$	Number of Nodes	839	862
$\|E\|$	Number of Edges	10614	11427
D	Diameter	8	8.1
PL	Average path length	3.439	3.881
AD	Average degree	12.651	14.498
DEN	Density	0.015	0.129
$WConn$	Weak Connected Components	72	98
$SConn$	Strong Connected Components	201	175
Mod	Modularity	0.625	0.712
Com	Communities Detected	76	102
$Init$	Initial Communities	270	244

knowledge of the specific spectral features (such as spectral gap information) of the graph. PageRank clustering tends to create one cluster holding half the nodes and two clusters each containing one quarter of the nodes for three clusters in the planted partition model. Using $p = 0.5, q = 0.2$ and a range of φ values around 0.5 (the predicted conductance of the clusters is around 0.44). This may help explain why higher order approaches (such as evaluating triangles in the graph) may retrieve more information on the seeded partitions. According to the planted partition model, the difference between edge probabilities may be minimal, while triangle probabilities will be more different. Choosing a value for the threshold φ is a key challenge when using PageRank clustering. Choosing a number that is too low causes the PageRank algorithm to run too slowly (even if there is a set with conductance smaller than φ), while a value that is too high results in poor partitioning. This is predicted, given the temporal complexity is proportional to $1/\varphi^3$. In accordance with the above statement, the theoretical guarantee for PageRank clustering seems to be poor. In particular, the planted partition model seems to be applicable only when there is a substantial difference between p and q. To guarantee finding a cut with conductance less than φ, the theorem necessitates a cut with conductance so much less than φ. If the lowest conductance in the graph is not very low, the corresponding value of φ may be more than 1, making it unusable. Therefore, the theoretical guarantee does not seem to be applicable in practise.

In conclusion, we have presented the relevant theoretical and computational concepts connected with a global graph clustering strategy based on an eigenvector embedding and a local graph clustering technique based on local stochastic exploration of the network. Both approaches have been implemented and their performance on well-studied generative cluster models evaluated. Both algorithms were then applied to a real-world graph depicting an online social network.

REFERENCES

[1] S. Wu, X. Feng, and W. Zhou, "Spectral clustering of high-dimensional data exploiting sparse representation vectors," *Neurocomputing*, vol. 135, pp. 229–239, 2014.

[2] F. Rezaeimehr, P. Moradi, S. Ahmadian, N. N. Qader, and M. Jalili, "Tcars: Time-and community-aware recommendation system," *Future Generation Computer Systems*, vol. 78, pp. 419–429, 2018.

[3] L. Sheugh and S. H. Alizadeh, "A novel 2d-graph clustering method based on trust and similarity measures to enhance accuracy and coverage in recommender systems," *Information Sciences*, vol. 432, pp. 210–230, 2018.

[4] S. Ahmadian, N. Joorabloo, M. Jalili, M. Meghdadi, M. Afsharchi, and Y. Ren, "A temporal clustering approach for social recommender systems," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 1139–1144, IEEE, 2018.

[5] A. Karras and C. Karras, "Integrating User and Item Reviews in Deep Cooperative Neural Networks for Movie Recommendation," *arXiv preprint arXiv:2205.06296*, 2022.

[6] C. Karras, A. Karras, and S. Sioutas, "Pattern recognition and event detection on iot data-streams," *arXiv preprint arXiv:2203.01114*, 2022.

[7] C. Karras, A. Karras, G. Drakopoulos, K. Tsakalidis, P. Mylonas, and S. Sioutas, "Weighted Reservoir Sampling On Evolving Streams: A Sampling Algorithmic Framework For Stream Event Identification," in *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*, SETN '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[8] C. Karras, A. Karras, M. Avlonitis, and S. Sioutas, "An Overview of MCMC Methods: From Theory to Applications," in *Artificial Intelligence Applications and Innovations. AIAI 2022 IFIP WG 12.5 International Workshops* (I. Maglogiannis, L. Iliadis, J. Macintyre, and P. Cortez, eds.), (Cham), pp. 319–332, Springer International Publishing, 2022.

[9] C. Karras, A. Karras, M. Avlonitis, I. Giannoukou, and S. Sioutas, "Maximum Likelihood Estimators on MCMC Sampling Algorithms for Decision Making," in *Artificial Intelligence Applications and Innovations. AIAI 2022 IFIP WG 12.5 International Workshops* (I. Maglogiannis, L. Iliadis, J. Macintyre, and P. Cortez, eds.), (Cham), pp. 345–356, Springer International Publishing, 2022.

[10] C. Karras and A. Karras, "Dbsop: An efficient heuristic for speedy mcmc sampling on polytopes," *arXiv preprint arXiv:2203.10916*, 2022.

[11] C. Karras, A. Karras, L. Theodorakopoulos, I. Giannoukou, and S. Sioutas, "Expanding Queries with Maximum Likelihood Estimators and Language Models," in *Proceedings of the ICR'22 International Conference on Innovations in Computing Research* (K. Daimi and A. Al Sadoon, eds.), (Cham), pp. 201–213, Springer International Publishing, 2022.

[12] A. Karras, C. Karras, K. C. Giotopoulos, I. Giannoukou, D. Tsolis, and S. Sioutas, "Download Speed Optimization in P2P Networks Using Decision Making and Adaptive Learning," in *Proceedings of the ICR'22 International Conference on Innovations in Computing Research* (K. Daimi and A. Al Sadoon, eds.), (Cham), pp. 225–238, Springer International Publishing, 2022.

[13] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[14] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 475–486, IEEE, 2006.

[15] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 81–90, 2004.

[16] D. A. Spielman and S.-H. Teng, "Spectral partitioning works: Planar graphs and finite element meshes," *Linear Algebra and its Applications*, vol. 421, no. 2-3, pp. 284–305, 2007.

[17] E. Polizzi, "Density-matrix-based algorithm for solving eigenvalue problems," *Physical Review B*, vol. 79, no. 11, p. 115112, 2009.