Skew-Symmetric Adjacency Matrices for Clustering Directed Graphs

1st Koby Hayashi
School of Computational Science and Eng.
Georgia Institute of Technology
Atlanta, USA
khayashi9@gatech.edu

2nd Sinan G. Aksoy
Pacific Northwest National Lab
Richland, USA
sinan.aksoy@pnnl.gov

3rd Haesun Park
School of Computational Science and Eng.
Georgia Institute of Technology
Atlanta, USA
hpark@cc.gatech.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020413

***Abstract*—Cut-based directed graph (digraph) clustering often focuses on finding dense within-cluster or sparse between-cluster connections, similar to cut-based undirected graph clustering. In contrast, for flow-based clusterings the edges between clusters tend to be oriented in one direction and have been found in migration data, food webs, and trade data. In this paper we introduce a spectral algorithm for finding flow-based clusterings. The proposed algorithm is based on recent work which uses complex-valued Hermitian matrices to represent digraphs. By establishing an algebraic relationship between a complex-valued Hermitian representation and an associated real-valued, skew-symmetric matrix the proposed algorithm produces clusterings while remaining completely in the real field. Our algorithm is more memory efficient, requires less computation, and provably preserves solution quality. We also show the algorithm can be easily implemented using standard computational building blocks, possesses better numerical properties, and loans itself to a natural interpretation via an objective function relaxation argument.**

***Index Terms*—Spectral Clustering, Digraph, Oriented Graph**

I. Introduction

Many methods for undirected graph clustering focus on finding minimal cuts between dense clusters [1] and many directed graph (digraph) clustering methods seek to extend this idea [2], [3]. However, there has also been attention paid to finding large 'imbalanced cuts' in digraphs. These are cuts where most of the edges are oriented from one cluster to the other with few oriented in the reverse direction. Such cuts are present in migration data [4], food webs [5], [6], and trade data [7]. We refer to this dichotomy as density versus flow-based clustering.

Spectral approaches are frequently used for density-based graph clustering [1], [8]–[10]. More recently, researchers have also applied spectral techniques for finding flow-based clusterings. Spectral algorithms for mining flow-based patterns vary primarily in the matrix used to represent the digraph. The matrix representations used can be broadly classified as general non-symmetric matrices, symmetrizations of the adjacency matrix, and complex-valued Hermitian matrices. In particular, complex-valued Hermitian matrices have received much recent attention for encoding a wide variety of digraph structural properties. [4], [7], [11]–[13].

Complex-valued Hermitian adjacency or Laplacian matrices are appealing because they have nice theoretical properties. Like real-valued symmetric matrices, complex-valued Hermitian matrices are subject to the spectral theorem, min-max theorem for eigenvalues, eigenvalue interlacing properties, and more. In these complex-valued Hermitian matrix representations, complex numbers are used to encode edge direction. This is a potential advantage over symmetrization approaches which often lose information related to edge direction and asymmetric representations which often lack useful theoretical properties. Some recent applications of Hermitian representations include the so-called magnetic digraph Laplacian's utilization in signal processing [12] and node classification and link prediction [13]. There has also been interest in developing a spectral theory for other, closely related complex-valued Hermitian matrices [14], [15] . For flow-based clustering, Cucuringu et al. [4] proposed an algorithm for finding imbalanced cuts based on a complex-valued Hermitian digraph adjacency matrix whose effectiveness they demonstrate via an analysis of a Directed Stochastic Block Model (DSBM) and empirical studies on real data.

In this work, we propose and compare spectral clustering algorithms for finding imbalanced cuts in digraphs. The main results are facilitated by an algebraic relationship between Cucuringu et al.'s complex-valued matrix and an associated real-valued matrix, which we analyze via application of the Real Schur Decomposition. This relationship enables an alternative algorithm to the one proposed in [4]. Our proposed algorithm uses less computation and asymptotically less memory, while provably preserving solution quality. Additionally, it can be easily implemented using standard computational building blocks, possesses better numerical properties, and loans itself to a natural interpretation via an objective function relaxation argument. Lastly, we empirically demonstrate these advantages on both synthetic and real world data. In the later case it is demonstrated that the method can find meaningful flow-based cluster structures.

The paper is organized as follows: in Section II we establish notation, review relevant background, and review prior work on complex-valued digraph matrices. In Section III the algorithm is derived using the aforementioned algebraic relationship and motivated using a heuristic relaxation argument which aides in the interpretability of the method. Finally, in Section IV we present experimental results on the DSBM and

Symbols	Meaning	Symbols	Meaning
$\mathcal{E}$	Edge set	$\mathrm{Re}(\cdot)$	Real part
$\mathcal{V}$	Vertex set	$\mathrm{Im}(\cdot)$	Imaginary part
n	Number of vertices	$\oplus$	Direct matrix sum
$[\mathbf{A}]_+$	Proj. to nonnegative orthant	$\mathcal{A}$	A set, Euler script
$i = \sqrt{-1}$	Complex unit	$\mathbf{A}$	A matrix, bold-uppercase
$\mathbf{E}^\mathsf{T} = [\mathbf{I} \quad \mathbf{0}]$	Truncation matrix	$\mathbf{a}$	A vector, bold-lowercase
$\mathbb{1}$	Vector of all ones	$\lvert\cdot\rvert$	Absolute value or cardinality
$\mathbf{A}^\mathsf{T}$	Transposition	$\mathbf{M}$	Directed Graph Adj.
$\bar{\mathbf{A}}$	Complex conjugation	$\mathbf{H}$	Purely Hermitian Adj.
$\mathbf{A}^* = \bar{\mathbf{A}}^\mathsf{T}$	Conjugate Transpose	$\mathbf{K}$	Real Skew-Symmetric Adj.
$\lVert\cdot\rVert_F$	Frobenius norm	$\lVert\cdot\rVert_2$	2-norm

TABLE I: Notation

Food Web data sets.

II. Preliminaries

a) Definitions and notation: We use $a \in \mathbb{C}$ to denote scalars, $\mathbf{a} \in \mathbb{C}^n$ for vectors, $\mathbf{A} \in \mathbb{C}^{m\times n}$ for matrices, $\mathcal{A}$ for sets, and the superscripts T and $*$ for transposition and conjugate transpose, respectively. A directed graph or digraph $G = (\mathcal{V}, \mathcal{E})$ is a set of vertices $\mathcal{V}$ and a set $\mathcal{E}$ of *ordered* pairs of vertices, called edges. We assume a digraph is accompanied by an edge-weighting function $w : \mathcal{E} \to \mathbb{R}_{\geq 0}$, and denote the weight of edge (u, v) by w_{uv}. Further, if $(u, v) \in \mathcal{E}$, we sometimes write $u \to v$. If the digraph does not contain reciprocal edges, meaning $(u, v) \in \mathcal{E}$ implies that $(v, u) \notin \mathcal{E}$, then we call the digraph an *oriented graph*. A k-partition of the vertex set of a graph is a set of non-empty, disjoint sets $\{\mathcal{A}_1, \cdots \mathcal{A}_k\}$ such that $\bigcup_{j=1}^{k} \mathcal{A}_j = \mathcal{V}$. The directed adjacency matrix associated with G is $\mathbf{M} \in \mathbb{R}^{n\times n}$, where $\mathbf{M}_{uv} = w_{uv}$ if $u \to v \in \mathcal{E}$ and 0 otherwise. We frequently abuse notation by using vertex and cluster symbols as indices, e.g. $\mathbf{M}_{uv} = w_{uv}$ as stated in the previous line. We use $n = |\mathcal{V}|$ and as a general positive integer, for example a square matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$. For ease of reference and other notation, we provide Table I.

b) Linear Algebra: Some basic but relevant results in linear algebra, of which we make frequent use are as follows. A normal matrix $\mathbf{B} \in \mathbb{C}^{m\times m}$ has an eigenvalue decomposition $\mathbf{B} = \mathbf{U}^*\mathbf{\Lambda}\mathbf{U}$, where $\mathbf{\Lambda} \in \mathbb{C}^{m\times m}$ and diagonal and $\mathbf{U} \in \mathbb{C}^{m\times m}$ is unitary so $\mathbf{U}\mathbf{U}^* = \mathbf{I}$. We enforce that $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \cdots, \lambda_m)$ where $|\lambda_i| \geq |\lambda_j|$ if $i \leq j$. All eigenvalues of a Hermitian matrix are real. A matrix $\mathbf{K}$ where $-\mathbf{K} = \mathbf{K}^\mathsf{T}$ is called skew-symmetric and real skew-symmetric if $\mathbf{K} \in \mathbb{R}^{n\times n}$. If $\mathbf{K}$ is real skew-symmetric all its eigenvalues are either 0 or purely imaginary of the form αi where $\alpha \in \mathbb{R}$. If $(\alpha i, \mathbf{x})$ is an eigenpair of $\mathbf{K}$ so its $(\alpha\bar{i}, \bar{\mathbf{x}})$. The direct sum of a set of matrices is written as $\mathbf{B} = \mathbf{B}_1 \oplus \mathbf{B}_2 \oplus \cdots \oplus \mathbf{B}_m = \mathrm{diag}(\mathbf{B}_1, \cdots, \mathbf{B}_m)$, where $\mathbf{B} \in \mathbb{C}^{n\times n}$, $\mathbf{B}_j \in \mathbb{C}^{n_j\times n_j}$, and $\sum_{j=1}^{m} n_j = n$.

A. Complex-valued digraph matrices

Researchers have introduced a variety of different complex-valued adjacency matrices for studying digraphs, utilizing them for different purposes. Liu and Li [15] proposed a Hermitian adjacency matrix $\mathbf{A}$ where $\mathbf{A}_{uv} = 1$ if $(u, v) \in \mathcal{E}$ and $(v, u) \in \mathcal{E}$, i if $(u, v) \in \mathcal{E}$ and $(v, u) \notin \mathcal{E}$, $-i$ if $(u, v) \notin \mathcal{E}$ and $(v, u) \in \mathcal{E}$ and 0 otherwise.

Their motivation for proposing this matrix is that it encodes the directionality of the digraph while possessing strictly real eigenvalues. This enabled a meaningful definition of *Hermitian energy* of digraphs for applications in theoretical chemistry for computing the π-electron energy of a conjugated carbon molecule [15]. Concurrently, Guo and Mohar [14] proposed the same matrix for the purposes of establishing a basic spectral theory of digraphs. Later, Mohar [11] proposed a modification of the matrix, identically defined except that the complex unit i is replaced with a sixth root of unity $\omega = (1 + i\sqrt{3})/2$. ω is chosen due to the fact that $\omega \cdot \bar{\omega} = \omega + \bar{\omega} = 1$, which ensures the matrix encodes combinatorial properties of digraphs, such as $\mathbf{A}^k$ counting directed walks of length k. In Mohar's work and others, the choice of complex number is a parameter used to define the matrix, and is left to the user. For instance, in the q-adjacency matrix used to define magnetic Laplacian [13], the parameter q controls the choice of complex number in polar form. Taking $q = 1/4$ and $q = 1/6$ yields matrices almost identical to the aforementioned matrices, respectively, differing only in that non-reciprocal edges are weighted by a factor of $1/2$.

For this work, we focus on a related, but simplified complex-valued digraph adjacency matrix using the imaginary unit i which is used by Cucuringu et al. [4]. This matrix $\mathbf{H} \in \mathbb{C}^{n\times n}$, most properly defined for weighted, oriented digraphs, is given element-wise by

$$\mathbf{H}_{uv} = \begin{cases} w_{uv} \cdot i & \text{if } u \to v \\ -w_{vu} \cdot i & \text{if } v \to u \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Clearly $\mathbf{H}$ is Hermitian by definition. While defined for oriented graphs, this matrix can be naturally applied to general digraphs by replacing any pair of reciprocal edges between u and v with a single edge (u, v) having weight $w_{uv} - w_{vu}$ if $w_{uv} \geq w_{vu}$.

B. Imbalanced Cuts

Cucuringu et al. [4] use the matrix $\mathbf{H}$ as the adjacency matrix of an oriented graph. Via a statistical argument based on a proposed Directed Stochastic Block Model (DSBM) the authors argue that the eigenvectors of $\mathbf{H}$ can recover flow-based clusterings. The proposed DSBM is designed such that digraphs generated from it are expected to have large 'imbalanced cuts' between them. That is, cuts where most of the edges are oriented from one cluster to the other and few in the reverse direction. To measure the quality of a cut [4] uses a quantity called the Cut Imbalance (CI), which is defined as:

$$\mathrm{CI}(\mathcal{X}, \mathcal{Y}) = \frac{w(\mathcal{X}, \mathcal{Y})}{w(\mathcal{X}, \mathcal{Y}) + w(\mathcal{Y}, \mathcal{X})}, \tag{2}$$

where $\mathcal{X}, \mathcal{Y}$ are a partition and $w(\mathcal{X}, \mathcal{Y}) = \sum_{u\in\mathcal{X}, v\in\mathcal{Y}} \mathbf{M}_{uv}$, i.e., the sum of edges oriented from $\mathcal{X}$ to $\mathcal{Y}$. One can see that a large $\mathrm{CI}(\mathcal{X}, \mathcal{Y})$ value means that most of the edges are oriented from $\mathcal{X}$ to $\mathcal{Y}$, a small CI value means most edges are oriented from $\mathcal{Y}$ to $\mathcal{X}$ and a CI value close to $\frac{1}{2}$ means that the cut is balanced in the sense that $w(\mathcal{X}, \mathcal{Y}) - w(\mathcal{Y}, \mathcal{X})$ is close to 0. In order to make this a maximization problem, the equation $|\mathrm{CI}(\mathcal{X}, \mathcal{Y}) - \frac{1}{2}|$ is considered instead.

Algorithm 1 Hermitian Clustering (Herm)

input: A directed graph and desired number of clusters k.
Construct $\mathbf{H} \in \mathbb{C}^{n\times n}$ as described by Eqn. 1
Assign $l = k$ if k is even and $l = k-1$ if k is odd
Compute the l largest magnitude eigenvalues and their corresponding eigenvectors of the matrix $\mathbf{H}$ $\{(\lambda_1, \mathbf{w}_1), \cdots, (\lambda_l, \mathbf{w}_l)\}$
Compute $\mathbf{P} = \sum_{j=1}^{l} \mathbf{w}_j \mathbf{w}_j^* \in \mathbb{R}^{n\times n}$
Compute a second EVD of $\mathbf{P}$ to obtain the matrix $\mathbf{G} \in \mathbb{R}^{n\times l}$ whose columns are the first l eigenvectors of $\mathbf{P}$.
Run k-means on n rows of $\mathbf{G}$ with k clusters
return k vertex clusters

Cucuringu et al.'s statistical analysis of the DSBM bounds, with a certain probability, the number of vertices misclassified by their spectral algorithm. As previously mentioned, this algorithm uses the eigenvectors corresponding to the largest magnitude eigenvalues of $\mathbf{H}$. However, Cucuringu et al. provide no direct connection between the CI, Eqn. 2, and their spectral algorithm. Algorithm 1 presents details of this spectral algorithm, which we refer to as Hermitian Clustering (Herm).

III. Proposed Algorithm

We now motivate and derive our proposed algorithm. First we discuss aspects of Hermitian Clustering, highlighting properties of the matrix $\mathbf{H}$ which the algorithm utilizes. Then, we explore some matrix decompositions related to $\mathbf{H}$. From these matrix decompositions and their relationships, we derive a new, improved clustering algorithm and motivate its applicability to flow-based clustering using a relaxation argument.

A. Motivation

Observe the matrix $\mathbf{H} \in \mathbb{C}^{n\times n}$ utilized in Algorithm 1 is not only Hermitian but also skew-symmetric. Further, given any digraph with adjacency matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$, $\mathbf{H}$ may be written as $\mathbf{H} = i\mathbf{K} = i(\mathbf{M} - \mathbf{M}^\mathsf{T})$ where $\mathbf{K} = (\mathbf{M} - \mathbf{M}^\mathsf{T}) \in \mathbb{R}^{n\times n}$ is real skew-symmetric. This implies that $\mathbf{K}$ and $\mathbf{H}$ have the same eigenvectors and there is a relationship between clustering based on $\mathbf{H}$ or $\mathbf{K}$. Denote the Hermitian Eigenvalue Decomposition (EVD) of $\mathbf{H} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^*$. Then the EVD of $\mathbf{K}$ is

$$\mathbf{K} = \mathbf{W}(\bar{i}\mathbf{\Lambda})\mathbf{W}^*. \tag{3}$$

Note that while $\mathbf{K}$ is a real-valued matrix, its EVD requires complex-valued matrices.

Algorithm 1 utilizes an even number of the leading eigenvectors of $\mathbf{H}$ to form a low-rank representation of a graph. Specifically, the matrix $\tilde{\mathbf{W}} = [\mathbf{w}_1, \cdots, \mathbf{w}_l] \in \mathbb{C}^{n\times l}$ is formed, where $\mathbf{w}_j$ is the jth column of $\mathbf{W}$ and $l > 1$ is an even integer then k-means with k clusters is run on the product $\mathbf{P} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^*$. Due to the fact that if $(\lambda_j, \mathbf{x}_j)$ is an eigenpair of $\mathbf{K}$ so is $(-\lambda_j, \bar{\mathbf{x}}_j)$, the matrix $\mathbf{P} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^* \in \mathbb{R}^{n\times n}$ is real valued. This means a standard k-means algorithm that takes real valued input can be run on $\mathbf{P}$.

However, the formation of $\mathbf{P}$ may cause computational issues, despite having the desirable property of being real-valued. First, the matrix $\mathbf{P}$ is of size $n \times n$ which is as large as the input graph and is likely dense. Therefore running k-means on and storing this $\mathbf{P}$ can be prohibitively expensive for large problems. Second, $\mathbf{P}$ may incur additional numerical issues due to the formation of the product, and in fact may not be real valued [16]. This can be overcome in a number of ways, for example by taking its real part and discarding the residual imaginary components. In the next section we propose a solution to these problems by observing some algebraic relationships.

B. Algebraic Properties

The real, skew-symmetric matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ has a real-valued Singular Value Decomposition (SVD) of the form $\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$ where $\{\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}\} \in \mathbb{R}^{n\times n}$. Working with the SVD of $\mathbf{K}$ is desirable due to the fact that computing it requires only real arithmetic (unlike the EVD) and reliable algorithms and software are readily available for its computation. Here we will follow this idea of using the SVD in place of the EVD.

The derivation of our algorithm relies on properties of the Real Schur Decomposition (RSD) of $\mathbf{K}$ [16]. The Schur Decomposition (SD), as opposed to the RSD, decomposes an arbitrary matrix $\mathbf{B} \in \mathbb{C}^{n\times n}$ into $\mathbf{B} = \hat{\mathbf{Q}}\mathbf{R}\hat{\mathbf{Q}}^*$ where $\hat{\mathbf{Q}} \in \mathbb{C}^{n\times n}$, $\hat{\mathbf{Q}}\hat{\mathbf{Q}}^* = \mathbf{I}$, and $\mathbf{R} \in \mathbb{C}^{n\times n}$ is upper triangular. We emphasize the EVD and SD are different in general. Even for a real matrix, $\mathbf{A} \in \mathbb{R}^{n\times n}$, its SD $\mathbf{A} = \hat{\mathbf{Q}}\mathbf{R}\hat{\mathbf{Q}}^*$ consists of a unitary matrix $\hat{\mathbf{Q}} \in \mathbb{C}^{n\times n}$ and an upper triangular matrix $\mathbf{R} \in \mathbb{C}^{n\times n}$ which is also complex in general. Alternatively, the RSD of a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^\mathsf{T}$ where $\mathbf{Q} \in \mathbb{R}^{n\times n}$, $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}$, and $\mathbf{T} \in \mathbb{R}^{n\times n}$, where $\mathbf{T}$ is block upper triangular with either 2×2 or 1×1 blocks on the diagonal, instead of being upper triangular.

Returning now to $\mathbf{K}$, since this matrix is real skew-symmetric, its RSD

$$\mathbf{K} = \mathbf{Q}\mathbf{T}\mathbf{Q}^\mathsf{T} \tag{4}$$

has a special form [17], where $\mathbf{Q} \in \mathbb{R}^{n\times n}$, $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}$, and $\mathbf{T} = \mathbf{T}_1 \oplus \cdots \oplus \mathbf{T}_b \in \mathbb{R}^{n\times n}$ is block diagonal with b diagonal blocks of size 1×1 or 2×2. Since all nonzero eigenvalues of $\mathbf{K}$ are purely imaginary, and appear in $\pm$ pairs, one may assume that $\mathbf{K}$ has $2s$ non-zero eigenvalues and each $\pm$ pair of the $2s$ eigenvalues appears in a block $\mathbf{T}_j \in \mathbb{R}^{2\times 2}$ for $j = 1, \ldots, s$, $\mathbf{T}_t = 0 \in \mathbb{R}^{1\times 1}$ for $t = (2s+1), \ldots, b$. We may further assume that each block $\mathbf{T}_j$ has the form $\mathbf{T}_j = \begin{bmatrix} 0 & \alpha_j; -\alpha_j & 0 \end{bmatrix}$ whose eigenvalues are $\alpha_j i$ and $-\alpha_j i$, and the blocks $\mathbf{T}_j$ are ordered in non-increasing order by $|\alpha_j|$ and $\alpha_j > 0$. These real Schur vectors can be easily used to construct eigenvectors. Observe that

$$\mathbf{K}\begin{bmatrix}\mathbf{q}_{2j-1} & \mathbf{q}_{2j}\end{bmatrix} = \begin{bmatrix}\mathbf{q}_{2j-1} & \mathbf{q}_{2j}\end{bmatrix}\begin{bmatrix} 0 & \alpha_j \\ -\alpha_j & 0 \end{bmatrix} = \begin{bmatrix}-\alpha\mathbf{q}_{2j} & \alpha\mathbf{q}_{2j-1}\end{bmatrix},$$

and $\mathbf{K}(\mathbf{q}_{2j-1} + i\mathbf{q}_{2j}) = -\alpha\mathbf{q}_{2j} + i\alpha\mathbf{q}_{2j-1} = i\alpha(\mathbf{q}_{2j-1} + i\mathbf{q}_{2j})$, so $(\mathbf{q}_{2j-1} + i\mathbf{q}_{2j})$ is an eigenvector of $\mathbf{K}$, and therefore also of $\mathbf{H}$.

Generalizing this observation define $\mathbf{J} = \mathbf{J}_1 \oplus \cdots \oplus \mathbf{J}_b \in \mathbb{C}^{n\times n}$, where $\mathbf{J}_j = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -i;1 & i\end{bmatrix} \in \mathbb{C}^{2\times 2}$ for $j = 1,\ldots,s$, and $\mathbf{J}_t = 1 \in \mathbb{R}^{1\times 1}$ for $t = (2s+1),\ldots,b$. Then $\mathbf{JTJ}^* = \bar{i}\mathbf{\Lambda}$, since $\mathbf{J}_j$, for $j = 1,\ldots,s$, unitarily diagonalizes $\mathbf{T}_j$, as $\mathbf{J}_j\mathbf{T}_j\mathbf{J}_j^* = \begin{bmatrix}-\alpha_j\bar{i} & 0;0 & \alpha_j\bar{i}\end{bmatrix}$. Therefore, we have

$$\mathbf{K} = \mathbf{QTQ}^{\mathsf{T}} = (\mathbf{QJ}^*)(\bar{i}\mathbf{\Lambda})(\mathbf{JQ}^{\mathsf{T}}) = \mathbf{W}(\bar{i}\mathbf{\Lambda})\mathbf{W}^*, \quad (5)$$

an EVD of $\mathbf{K}$ where $\mathbf{W} = (\mathbf{QJ}^*)$. This shows a relationship between the eigenvectors $\mathbf{W}$ in Eqn. 3 and the real Schur vectors $\mathbf{Q}$ in Eqn. 4 of $\mathbf{K}$.

We are now ready to discuss our first main proposition which enables our proposed algorithm.

Proposition 1. *Let $\tilde{\mathbf{Q}} = \mathbf{QE} \in \mathbb{R}^{n\times l}$, where $\mathbf{E} \in \mathbb{R}^{n\times l}$ is a truncation matrix whose l columns are the first l columns of the identity matrix of order n, l is a positive even integer $\leq 2s$. Assume that $\mathbf{K}$ has $2s$ non-zero eigenvalues. Then the embedding $\tilde{\mathbf{Q}}$ has the same Euclidean distances between all pairs of vertices as the embedding $\mathbf{P} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^* \in \mathbb{R}^{n\times n}$.*

Proof. Assume that a set of d points are collected as the rows of a matrix $\mathbf{Y} \in \mathbb{R}^{d\times f}$ as $\mathbf{y}_1^T, \cdots, \mathbf{y}_d^T$, where f is the number of features. Then the Euclidean distance matrix [18] of $\mathbf{Y}$, $\Delta(\mathbf{Y})$ whose (i,j)th element is the L_2-norm distance between row i and row j, $(\Delta(\mathbf{Y}))_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|_2^2$, is defined as

$$\Delta(\mathbf{Y}) = \mathbb{1}\ \mathrm{diag}(\mathbf{YY}^{\mathsf{T}})^{\mathsf{T}} - 2\mathbf{YY}^{\mathsf{T}} + \mathrm{diag}(\mathbf{YY}^{\mathsf{T}})\mathbb{1}^{\mathsf{T}}.$$

Since

$$\begin{aligned}(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*)^{\mathsf{T}})^{\mathsf{T}} &= (\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*\overline{\tilde{\mathbf{W}}}\tilde{\mathbf{W}}^{\mathsf{T}}) = (\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*\overline{\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*}) \\ &= (\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*) = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^* = \mathbf{QJ}^*\mathbf{E}(\mathbf{QJ}^*\mathbf{E})^* \\ &= \mathbf{QJ}^*\mathbf{EE}^{\mathsf{T}}\mathbf{JQ}^{\mathsf{T}} = \tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^{\mathsf{T}} \quad (6)\end{aligned}$$

we have $\Delta(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^*) = \Delta(\tilde{\mathbf{Q}})$. □

Proposition 1 implies an equivalent but simplified version of Algorithm 1. That is, compute the embedding $\tilde{\mathbf{Q}}$ from $\mathbf{K}$ and run k-means on it. Since both embeddings have the same Euclidean distances between vertices it can be expected that the algorithms produce the same result. A small issue that we now address is the use of the RSD.

Proposition 2. *The embedding $\tilde{\mathbf{Q}} = \mathbf{QE} \in \mathbb{R}^{n\times l}$ can be obtained from the Singular Value Decomposition of $\mathbf{K}$.*

Proof. Define the matrix $\mathbf{Z} = \mathbf{Z}_1 \oplus \cdots \oplus \mathbf{Z}_b$, which has the same block structure as $\mathbf{T}$, where $\mathbf{Z}_j = [0 \quad -1;1 \quad 0]$ for $j = 1,\cdots,s$ and $\mathbf{Z}_t = 1$ for $t = (s+1),\cdots,b$. Note that $\mathbf{Z}$ is orthogonal. Then

$$\mathbf{K} = \mathbf{QTQ}^{\mathsf{T}} = \mathbf{Q}(\mathbf{TZ})(\mathbf{Z}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}), \quad (7)$$

which is an SVD of $\mathbf{K}$ where $\mathbf{U} = \mathbf{Q}$, $\mathbf{\Sigma} = \mathbf{TZ}$, and $\mathbf{V} = \mathbf{QZ}$. The columns of $\tilde{\mathbf{Q}}$ can be obtained from $\mathbf{U}$ or $\mathbf{V}$. □

As a result of Proposition 2, the embedding $\tilde{\mathbf{Q}}$ can be easily obtained from computing the SVD of $\mathbf{K}$ using a readily available high quality implementation. Algorithm Skew-F presents the psuedo code of our main algorithm, which we call Skew-Symmetric Clustering.

a) SVD-Search Algorithm: Since the number of eigen/singular vectors to take is an open question. We propose searching for a large 'gap' in the eigen/singular values and using all the pairs above the gap. Empirically we find that this method significantly outperforms Hermitian Clustering and standard Skew-Symmetric Clustering on the graphs generated by certain inputs for the DSBM. Additionally, we experiment with manual inputs of the parameter l. Results in Section IV show that determining an appropriate l has a large impact on the embedding quality.

Algorithm Skew-F Skew-Symmetric Clustering (Skew-F)

input: A digraph adjacency matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ and desired number of clusters k.
Construct $\mathbf{K} = \mathbf{M} - \mathbf{M}^{\mathsf{T}}$.
Let $l = k$ if k is even and $l = k-1$ if k is odd
Compute a truncated SVD of $\mathbf{K} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^{\mathsf{T}}$ where $\{\tilde{\mathbf{U}}, \tilde{\mathbf{\Sigma}}, \tilde{\mathbf{V}}\} \in \mathbb{R}^{n\times l}$
Run k-means on $\tilde{\mathbf{U}}$ for k clusters
return k vertex clusters

C. Trade Flow and Skew-Symmetric Clustering

We now present a connection between our Skew-Symmetric Clustering algorithm and an intuitive cluster-quality metric called *Trade Flow*. Recall Eqn. 2 defines the Cut Imbalance (CI). Although [4] does not explicitly tie CI to Algorithm 1, the authors utilize this metric to evaluate the cluster quality of their methods on real world datasets where no ground truth was available. In place of the CI, we consider the Trade Flow (TF) metric for measuring imbalanced cuts, as proposed by Laenen [19]:

$$\mathrm{TF}(\mathcal{X}, \mathcal{Y}) = |w(\mathcal{X}, \mathcal{Y}) - w(\mathcal{Y}, \mathcal{X})|. \quad (8)$$

In the context of finding clusters with large imbalanced cuts the goal is to maximize $\mathrm{TF}(\mathcal{X}, \mathcal{Y})$ over the vertex partition $\mathcal{X}$ and $\mathcal{Y}$. Clearly, the TF is similar in spirit to the CI. A large value of TF, Eqn. 8, means that more edge weight is oriented from one cluster to the other than vice versa. A small value means that the cut is relatively balanced and thus by attempting to maximize Eqn. 8 one expects to obtain clusters with large imbalanced cuts between them. In fact a relationship can be seen between CI and TF as

$$|\mathrm{CI}(\mathcal{X}, \mathcal{Y}) - \frac{1}{2}| = \frac{1}{2} \cdot \left|\frac{w(\mathcal{X}, \mathcal{Y}) - w(\mathcal{Y}, \mathcal{X})}{w(\mathcal{X}, \mathcal{Y}) + w(\mathcal{Y}, \mathcal{X})}\right| = \frac{1}{2} \cdot \frac{\mathrm{TF}(\mathcal{X}, \mathcal{Y})}{|w(\mathcal{X}, \mathcal{Y}) + w(\mathcal{Y}, \mathcal{X})|}$$

Now we present a heuristic, relaxation argument for why Skew-Symmetric Clustering can be expected to recover large imbalanced cuts. Specifically, we will show that when $k = 2$ our method can be viewed as maximizing Eqn. 8 by relaxing the problem over the reals. We note the TF problem for $k = 2$ is solvable in linear time but for $k \geq 3$ is NP-hard [19]. This relaxation is not meant as an improved algorithm but to connect the above methods to a reasonable objective function.

Consider two indicator vectors for the partition $\mathcal{X}$ and $\mathcal{Y}$ denoted $\mathbf{e}_{\mathcal{X}}$ and $\mathbf{e}_{\mathcal{Y}}$, where $(\mathbf{e}_{\mathcal{X}})_u = 1$ if vertex $u \in \mathcal{X}$ and 0

otherwise. Given a digraph we can write the TF in terms of the adjacency matrix $\mathbf{M}$ as

$$\begin{aligned}\text{TF}(\mathcal{X},\mathcal{Y}) = |w(\mathcal{X},\mathcal{Y}) - w(\mathcal{Y},\mathcal{X})| &= |\mathbf{e}_{\mathcal{X}}^{\mathsf{T}}\mathbf{M}\mathbf{e}_{\mathcal{Y}} - \mathbf{e}_{\mathcal{Y}}^{\mathsf{T}}\mathbf{M}\mathbf{e}_{\mathcal{X}}| \\ &= |\mathbf{e}_{\mathcal{X}}^{\mathsf{T}}\mathbf{K}\mathbf{e}_{\mathcal{Y}}|\end{aligned}$$

Using the above observation we can then write the TF maximization problem as

$$\max_{\mathcal{X},\mathcal{Y}} \text{TF}(\mathcal{X},\mathcal{Y}) = \max |\mathbf{e}_{\mathcal{X}}^{\mathsf{T}}\mathbf{K}\mathbf{e}_{\mathcal{Y}}| \text{ s.t. } \{\mathbf{e}_{\mathcal{X}},\mathbf{e}_{\mathcal{Y}}\} \in \{0,1\}^n \text{ and } \mathbf{e}_{\mathcal{X}}^{\mathsf{T}}\mathbf{e}_{\mathcal{Y}} = 0 \quad (9)$$

Relaxing this problem by allowing $\mathbf{e}_{\mathcal{X}}$ and $\mathbf{e}_{\mathcal{Y}}$ to take on arbitrary real values we can instead consider the problem as

$$\max |\mathbf{a}_{\mathcal{X}}^{\mathsf{T}}\mathbf{K}\mathbf{b}_{\mathcal{Y}}| \text{ s.t. } \|\mathbf{a}_{\mathcal{X}}\| = \|\mathbf{b}_{\mathcal{Y}}\| = 1, \{\mathbf{a}_{\mathcal{X}},\mathbf{b}_{\mathcal{Y}}\} \in \mathbb{R}^n, \text{ and } \mathbf{a}_{\mathcal{X}}^{\mathsf{T}}\mathbf{b}_{\mathcal{Y}} = 0 \quad (10)$$

Where the norm constraint deals with scaling and the orthogonality constraint takes the place of $\mathcal{X} \cap \mathcal{Y} = \emptyset$.

Proposition 3. *Skew-Symmetric Clustering solves Equation 10. Therefore Skew-Symmetric Clustering can be viewed as solving a relaxation of Equation 9, the Trade Flow maximization problem.*

Proof. Consider the maximization problem in Eqn. 10, it is well known for an arbitrary matrix $\mathbf{B}$ the quantity $\mathbf{x}^{\mathsf{T}}\mathbf{B}\mathbf{y}$ is maximized by setting $\mathbf{x}$ to equal the first left singular vector of $\mathbf{B}$ and $\mathbf{y}$ to equal the first right singular vector of $\mathbf{B}$, assuming $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$. This is exactly what Skew-Symmetric Clustering does. Additionally since $\mathbf{K}$ is a real valued matrix its singular vectors are chosen to be real.

Next consider the orthogonality constraint, $\mathbf{a}_{\mathcal{X}}^{\mathsf{T}}\mathbf{b}_{\mathcal{Y}} = 0$. In general one does not expect the first left and right singular vectors of a matrix to be orthogonal. From Section III-B we know one can write $\mathbf{K} = \mathbf{Q}\mathbf{T}\mathbf{Q}^{\mathsf{T}} = \mathbf{Q}(\mathbf{T}\mathbf{Z})(\mathbf{Z}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}})$ using its RSD. As previously discussed this can be viewed as an SVD of $\mathbf{K}$ where $\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$ where $\mathbf{U} = \mathbf{Q}$, $\mathbf{V} = \mathbf{Q}\mathbf{Z}$ and $\mathbf{T}\mathbf{Z} = \mathbf{\Sigma}$. It then follows that $\mathbf{V}^{\mathsf{T}}\mathbf{U} = \mathbf{Z}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Q} = \mathbf{Z}^{\mathsf{T}}$ and $\mathbf{e}_1^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}\mathbf{V}\mathbf{e}_1 = \mathbf{u}_1^{\mathsf{T}}\mathbf{v}_1 = \mathbf{e}_1^{\mathsf{T}}\mathbf{Z}\mathbf{e}_1 = 0$. □

IV. Experiments

In this section we examine the empirical performance of our algorithms versus existing methods. First, we demonstrate our methods performance on synthetic data generated from the DSBM. In particular, we consider three different ways of generating the DSBM and include thorough experimental results for each. Second, we explore our methods effectiveness when applied to real world data. We consider the following algorithms:

1) Hermitian Clustering (Herm) see Algorithm 1.
2) Skew-Symmetric Clustering Full (Skew-F) see Algorithm Skew-F.
3) Skew-Symmetric Clustering Reduced (Skew-R) which is the same as Skew-F but takes in a user specified parameter l.
4) Skew-Symmetric Clustering Search (Skew-S) as in Algorithm Skew-F but modified as described in Section III-B0a. That is the gap in the singular values is used to determine l.
5) DD-Sym computes $\mathbf{S} = \mathbf{M}\mathbf{M}^{\mathsf{T}} + \mathbf{M}^{\mathsf{T}}\mathbf{M}$ and uses the top k-eigenvectors to cluster via k-means [20]. It was one of the top performing algorithms compared against 1 in [4].
6) Block Cyclic Clustering (BCS) uses elements of Perron Frobenius theory to compute a vertex embedding from the row normalized adjacency matrix [5].
7) SVD-M computes d left and right singular vectors of the adjacency matrix $\mathbf{M} \approx \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^{\mathsf{T}}$, forms them into an embedding as $[\hat{\mathbf{U}}\hat{\mathbf{\Sigma}}^{1/2}, \hat{\mathbf{V}}\hat{\mathbf{\Sigma}}^{1/2}]$, and applies K-means to extract clusters [21]. We set $d = k$.

Algorithm BCS Block Cyclic Spectral Clustering (BCS)

input: A directed, strongly connected graph adjacency matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ and desired number of clusters k.
Construct $\mathbf{P} = \mathbf{D}_{out}^{-1}\mathbf{M}$ where $\mathbf{D}_{out} = \text{diag}(M)\mathbb{1}$
Compute $l = \lfloor \frac{k}{2} \rfloor$
Compute the l largest eigenvalues $\lambda_1, \cdots, \lambda_l$ of $\mathbf{P}$ with largest absolute value that satisfy $\lambda \in \mathbb{C} : \text{Re}(\lambda) < 1, \text{Im}(\lambda) \geq 0$ and the associated right eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_l$
Collect the vectors into the matrix $\mathbf{\Gamma} = [\mathbf{u}_1, \cdots, \mathbf{u}_l] \in \mathbb{C}^{n\times l}$
Run k-means for k clusters on the matrix $[\text{Re}(\mathbf{\Gamma}), \text{Im}(\mathbf{\Gamma})]$
return k vertex clusters

We note there are also normalized variants of the above algorithms, which we utilize later in Section IV-C. For completeness, we also note Laenen and Sun [7] give an algorithm for the circulant case of the DSBM, see Section IV-A. All algorithms we consider are generally applicable and not restricted to the circulant case.

Algorithm DD-Sym Bibliometric Clustering (DD-Sym)

input: A directed graph adjacency matrix $\mathbf{M} \in \{0,1\}^{n\times n}$, desired number of clusters k, and $0 \leq \alpha \leq 1$
Compute $\mathbf{A} = \alpha\mathbf{M}\mathbf{M}^{\mathsf{T}} + (1-\alpha)\mathbf{M}^{\mathsf{T}}\mathbf{M}$
Compute the first k leading eigenvectors of $\mathbf{A}$ and collect them in the matrix $\mathbf{B}$
Run k-means for k clusters on the matrix $\mathbf{B}$
return k vertex clusters

a) Cut Metrics: To evaluate cluster quality, we utilize several multi-way cut objectives suitable for partitions with $k \geq 2$ clusters. First, we use an extension of CI metric (Eqn. 2), defined in [4] as:

$$\text{TopCI}^{vol}(\mathcal{A}_1, \cdots, \mathcal{A}_k) = \sum_{t=1}^{c} \text{CI}^{vol}(\mathcal{A}_{j_t}, \mathcal{A}_{h_t}) \quad (11)$$

where $\text{CI}^{vol}(\mathcal{A}_{j_t}, \mathcal{A}_{h_t})$ is the t-th largest CI^{vol} pair of clusters, $\text{CI}^{vol}(\mathcal{X},\mathcal{Y}) = |\text{CI}(\mathcal{X},\mathcal{Y}) - 0.5| * \min(vol(\mathcal{X}), vol(\mathcal{Y}))$, and

$vol(\mathcal{X})$ is the sum of all in and out degrees of vertices in $\mathcal{X}$. A related metric, TopCI^{sz} is defined by using the cardinality of a cluster in place of its volume in Eqn. 11. Secondly, for the TF metric (Eqn. 8) with $k \geq 2$, we have

$$\text{TopTF}(\mathcal{A}_1, \cdots, \mathcal{A}_k) = \sum_{t=1}^{c} \text{TF}(\mathcal{A}_{j_t}, \mathcal{A}_{h_t}) \text{ such that } j_t \geq h_t, \tag{12}$$

where, similar to the above, $\text{TF}(\mathcal{A}_{j_t}, \mathcal{A}_{h_t})$ is the t-th largest TF pair of clusters. In both equations c is the number of cuts considered.

A. Directed Stochastic Block Model Experiments

In our first set of experiments, we utilize the Directed Stochastic Block Model (DSBM) proposed in [4]. The DSBM is based on the inputs $(k, p, q, \mathbf{c}, \mathbf{F})$, where k denotes the number of clusters, p the probability that two vertices in the same cluster have an edge between them, q the probability that two vertices in different clusters have an edge between them, $\mathbf{c}$ a vector of length k whose entries are the number of vertices in each cluster, and the matrix $\mathbf{F} \in \mathbb{R}^{k \times k}$ which gives cluster level orientation probabilities. That is if u is in cluster a and v is in cluster b then $u \to v$ exists with probability $\mathbf{F}_{a,b}$. All diagonal entries of $\mathbf{F}$ are equal to $\frac{1}{2}$. The graph corresponding to the entries of $\mathbf{F}$ is called the *meta-graph* of a graph generated from this DSBM.

We utilize the following parameter settings: following [4] we set $p = q$ so that only the number of edges between clusters is expected to contain meaningful statistical information about the cluster memberships. We vary $p = 0.0045, 0.008$, set $n = 5000$, $k = 5$, and assign each cluster 1000 vertices. Further, we vary a noise parameter $0 \leq \mu < 0.5$ controlling the difficulty of recovery: if cluster i is oriented to cluster j, then $\mathbf{F}_{ij} = 1 - \mu$ and $\mathbf{F}_{ji} = \mu$. In this way, as μ approaches 0.5 the number of edges between clusters becomes random in expectation, making cluster recovery increasingly difficult; we consider 11 different μ values ranging from 0 to 0.3. We evaluate cluster quality using both the Adjusted Rand Index (ARI) [22] and TopTF (Eqn. 12), with an appropriate value of c. Under this setup, we consider three versions of DSBM, differing with regard to the structure of $\mathbf{F}$.

- *Circulant DSBM.* Here, the matrix $\mathbf{F}$ is circulant. This specific meta-graph model has received attention in recent work [4], [5], [7] because it is a natural pattern of interest, and because it affords tools from the spectral theory of (block) circulant matrices, as well as Perron-Frobenius theory. Figure 1 present the results. The best performing algorithms are Hermitian Clustering and the two Skew-Symmetric Clustering algorithms. We note for the Skew-R algorithm we set $l = 1$, and for TopTF computation we set $c = k$, as this is the number of meaningful cuts expected to be found.
- *Directed Acyclic DSBM.* Here, the meta-graph resembles a Directed Acyclic Graph (DAG). This case is motivated by the fact that matrices $\mathbf{F}$ constructed from DAGs have nonzero θ-distinguishing images – a requirement necessary for graphs generated by DSBM to statistically recoverable; see [4] for more. For our experiments we choose the DAG where the matrix $\mathbf{F}_{uv} = \mu$ if $v = u + 1$ or $v = u + 2$, $\mathbf{F}_{uv} = 1 - \mu$ if $v = u - 1$ or $v = u - 2$, and $\mathbf{F}_{uv} = 1/2$ otherwise. That is, the meta-graph is characterized by the first two lower and upper diagonals of $\mathbf{F}$. Figure 2 present the results. When computing TopTF we set $c = 2(k - 1)$, and again set $l = 1$ for the Skew-R algorithm. Results for this model are quite good for the SVD-R algorithm. As was the case for Circulant DSBM, the choice of l significantly impacts the results.
- *Complete Meta-Graph DSBM.* In the CMG model [4], $\mathbf{F}$ is generated by randomly orienting the flows between clusters, and setting all entries of $\mathbf{F}$, except for the diagonal, to either μ or $1 - \mu$. Figure 3 present the results. Skew-S significantly outperforms Herm and Skew-F in ARI, and slightly in terms of TopTF scores. This demonstrates a static choice of the number of eigen or singular vectors, l, is not the most effective technique. Moreover, this suggests optimal choice of l depends on the meta-graph pattern considered, rather than simply a function of the number of clusters. Lastly we note that the variance for these experiments is quite high. This is likely due to the fact that at each μ value ARI or TopTF scores from graphs with different $\mathbf{F}$'s are being averaged.

B. Computational Costs

The computational complexities of the various algorithms are not readily apparent from the given pseudo-code. In particular the complexity of the Herm algorithm was not thoroughly discussed in [4] and in fact some erroneous claims about its efficiency in comparison to other methods were made. Here we discuss and make clear the computational costs of Skew-Symmetric Clustering, Hermitian Clustering, and DD-Sym. Each of these algorithms relies on computing the top $O(k)$ eigenpairs of a sparse matrix. The efficient way to do this is to use an iterative Lanczos-type method [16]. The Lanczos method is iterative and does not require direct access to the matrix, say $\mathbf{A} \in \mathbb{C}^{n \times n}$, but only access to a linear function $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ that represents the action of $\mathbf{A}$ on a vector $\mathbf{x} \in \mathbb{C}^n$. A full discussion of the complexity of the Lanczos method is beyond the scope of this work. For our purposes we consider the complexity as $O(nnz(\mathbf{A})q_{max})$, where $\text{nnz}(\mathbf{A})$ denotes the number of nonzero entries of $\mathbf{A}$ and q_{max} is the number of $\mathbf{A}\mathbf{x}$ evaluations needed. This functionality is readily available in MATLAB and the python package Scipy via the eigs() function.

a) Complexity Analysis: In Skew-Symmetric Clustering, k-means is run on $\tilde{\mathbf{Q}} \in \mathbb{R}^{n \times l}$ yielding a computational complexity of $O(t_{max}k^2n)$, where t_{max} is the maximum number of k-means iterations. Additionally computing $\tilde{\mathbf{Q}}$ via Lanczos costs $O(nnz(\mathbf{Z})q_{max})$. Skew-Symmetric Clustering's storage complexity is $O(\text{nnz}(\mathbf{K}) + nl)$. Additionally, we note that only l singular vectors are needed in Skew-Symmetric Clustering even though typically it takes $2l$ real dimensions to capture l complex ones.

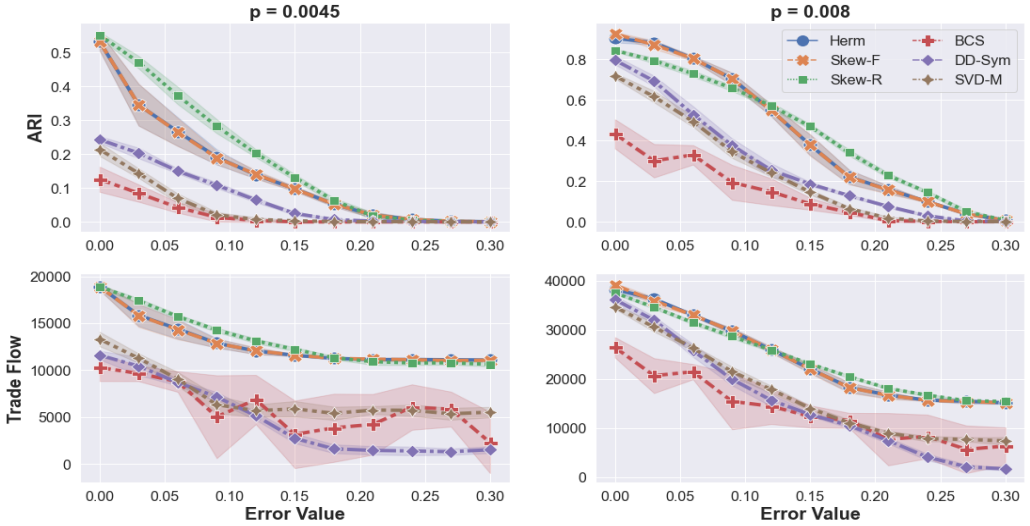


Fig. 1: ARI and TopTF results for the Circulant DSBM experiments.

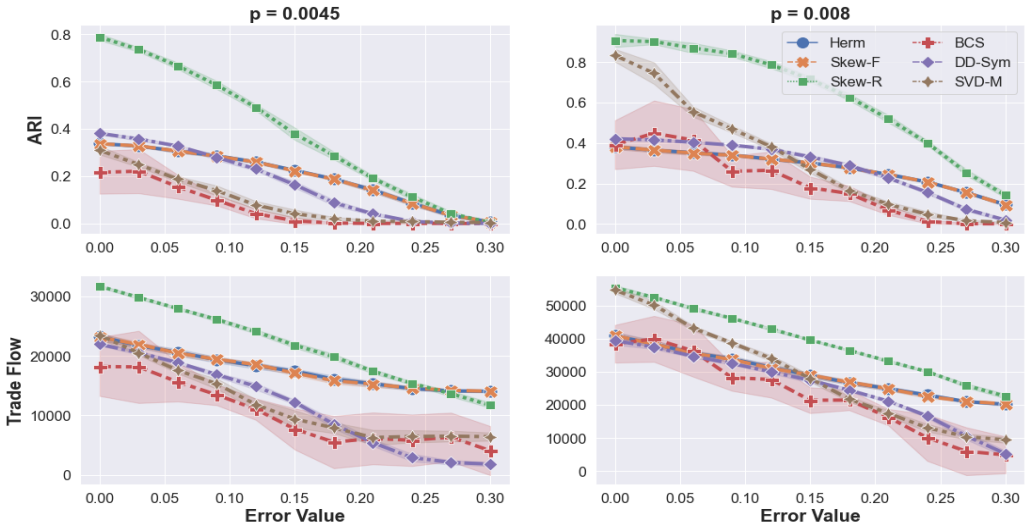


Fig. 2: ARI and TopTF results for the DAG DSBM experiments.

Herm, Algorithm 1, computes the embedding $\mathbf{P} = \mathbf{W}\mathbf{W}^*$ at a cost of $O(nnz(\mathbf{H})q_{max})$ and $O(n^2k)$ for the matrix multiply and the EVD. Then it computes $\mathbf{P} = \mathbf{G}\mathbf{G}^\mathsf{T}$, an EVD of $\mathbf{P}$, where $\mathbf{G} \in \mathbb{R}^{n\times l}$. Computing this explicitly, forming $\mathbf{P}$ directly and computing its eigenvalues, leads to a second costly EVD. Lastly k-means is run on $\mathbf{G}$ for cost $O(t_{max}k^2n)$. The cost of computing $\mathbf{G}$ in this way is $O(n^2q_{max})$. Therefore we see that Hermitian Clustering is more computationally more expensive than Skew-Symmetric Clustering. Additionally, forming $\mathbf{P}$ requires $O(n^2)$ memory, asymptotically more than Skew-Symmetric Clustering, which is infeasible for large problems. We note that many of these problems can be fixed by implementing Herm differently using a modified Lanczos method [16]. One can use the efficient matrix vector product $\mathbf{z} = (\mathbf{W}(\mathbf{W}^*\mathbf{x}))$ and project $\mathbf{z}$ to be real at each iteration instead of computing $\mathbf{G}$ from $\mathbf{P}$. This is not done in the previous work. We note without this projection step, we empirically observe that the Lanczos method will return a set of numerically-valid eigenvectors with large complex components thus failing the alleviate the original problem. For intuition as to why this occurs, note that all the columns of $\mathbf{W}$ are valid eigenvectors of $\mathbf{P}$.

While using this modified Lanczos procedure brings the asymptotic complexity of Herm inline with that of Skew-F, in practice Skew-F remains faster. This is due to constants, for example requiring only one SVD/EVD computation, a smaller embedding dimension, and using only real arithmetic. It is also claimed that the embedding $\mathbf{U}$ is 'analogous' to $\mathbf{W}$ but this is not rigorously justified [4]. In fact, via our analysis above one can straight forwardly prove that the singular vectors of $\mathbf{K}$ give a valid set of columns for $\mathbf{U}$.

Lastly, it is claimed, and supported with timing data that, in

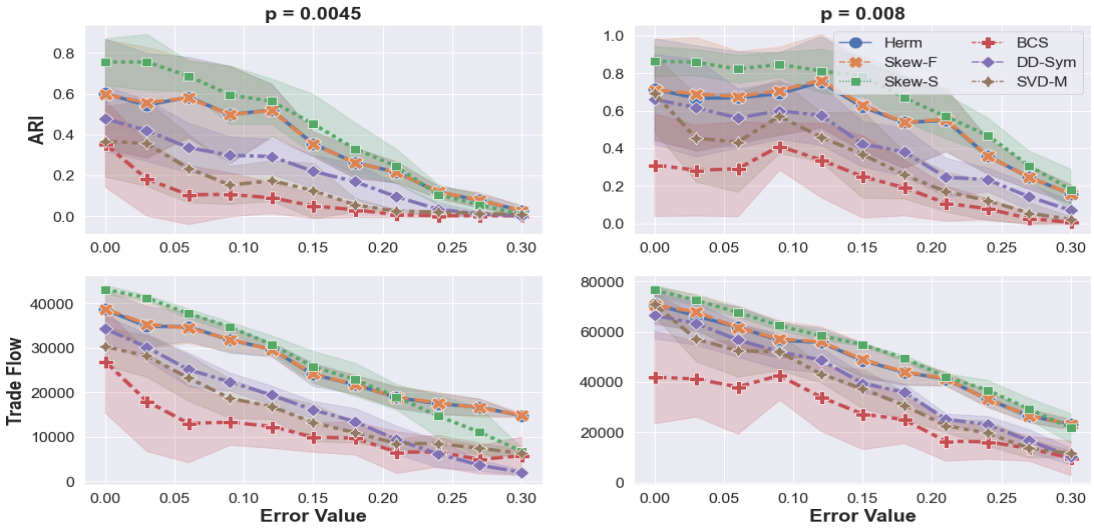


Fig. 3: ARI results for the CMG DSBM experiments.

Alg.	Embedding	Kmeans	SpeedUp
Herm	5.15	0.99	1
Skew	2.18	0.87	2
BCS	2.17	0.97	1.95
DD-Sym	1.04	1.08	2.9
SVD-M	1.26	1.39	2.3

TABLE II: Timings, in seconds, for Cyclic DSBM with $n = 10,000$, $k = 10$, and $p = 0.008$. Values are averages over 50 runs on 10 different graphs (5 runs per graph). Speed up is relative to the Herm algorithm.

[4] that the Herm Algorithm is significantly faster than DD-Sym due to the fact that DD-Sym requires the matrix products $\mathbf{M}\mathbf{M}^{\mathsf{T}} + \mathbf{M}^{\mathsf{T}}\mathbf{M}$. However, by using the Lanczos method with $f(\mathbf{x}) = \mathbf{M}(\mathbf{M}^{\mathsf{T}}\mathbf{x}) + \mathbf{M}^{\mathsf{T}}(\mathbf{M}\mathbf{x})$ this criticism no longer holds. In fact our timing experiments in Table II show DD-Sym is more efficient than Herm on a DSBM with $n = 10,000$.

b) Timing: We now consider timing results for a DSBM with parameters $k = 10$, $n = 10,000$, $p = 0.008$ and circulant $\mathbf{F}$. Table II has 3 columns : 1) Embedding (Emb.): the time spent selecting and computing the appropriate eigenvectors or singular vectors, 2) K-means: time spent running the k-means algorithm and 3) speed up relative to Hermitian Clustering.

All algorithms were run on a computer with a 2.3 GHz Quad-Core Intel Core i7 processor and 32GB memory, with matrices stored using MATLAB's sparse matrix format, and utilizing MATLAB's kmeans(), eigs() and svds() functions. MATLAB was given access to all 4 CPUs during the experiments.

We observe the Hermitian Clustering algorithm runs the slowest. This result is in contrast to previous timing results showing that Herm was roughly $2\times$ faster than DD-Sym [4].

Summarizing the DSBM experiments, Skew-Symmetric Clustering algorithms tend to perform the best, along with Hermitian Clustering, in terms of TopTF and ARI. This is congruent with Proposition 1 and results from [4]. Additionally, the Skew-Symmetric Clustering algorithm is cheaper in terms of memory consumption and computational cost than Hermitian Clustering. Skew-Symmetric Clustering's run time is comparable to the other methods and about $2\times$ faster than Herm. Lastly, we provided empirical evidence that the choice of l often has a significant impact on the performance of the algorithm.

C. Experiments on Real Data

Next, we test the performance of our algorithms on several food web datasets. We utilize normalizations which can often improve results on real-world data. Note that normalization is unnecessary for DSBM experiments, since vertices in the same clusters have the same expected value of edge degree. Random walk normalization is used in [4] and is written as $\mathbf{H}_{rw} = \mathbf{D}^{-1}\mathbf{H}$, where $\mathbf{D}_{uu} = \sum_v |\mathbf{H}_{uv}|$. One may analogously normalize $\mathbf{K}$ which we denote as $\mathbf{K}_{rw}$.

a) Florida Bay Food Web: The Florida Bay Food Web (FBFW)[1] is a data set containing information about carbon exchange between species in South Florida Ecosystems. In this digraph an edge $u \to v$ might mean species v eats species u. The graph contains 128 vertices and 2106 edges. We treat the graph as unweighted.

This data set has been analyzed quite extensively [5], [6], [23], and it is worth reviewing prior analyses to place ours in context. Benson et al. [23] use the FBFW to demonstrate the effectiveness of spectral motif clustering. Their clustering results separate out a number of interesting within-cluster dynamics. Li and Milenkovic [6] also used a spectral motif clustering approach in the context of their proposed inhomogenous hypergraph clustering problem. Their choice of motif resulted in finding 5 clusters where most of the edges are oriented between clusters, thus revealing the hierarchical structure present in the graph. The clusters in this hierarchical

[1] http://vlado.fmf.uni-lj.si/pub/networks/data/bio/foodweb/foodweb.htm

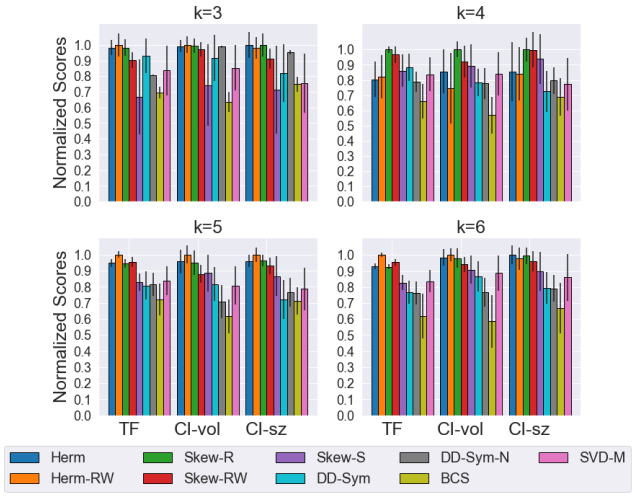


Fig. 4: Cut scores on the FBFW for $k = \{3, 4, 5, 6\}$. Bars corresponding to each metric are normalized by the highest achieved mean score so that all mean scores are between 0 and 1. Each bar's height is the mean over 100 runs and error bars give 1 standard deviation in each direction.

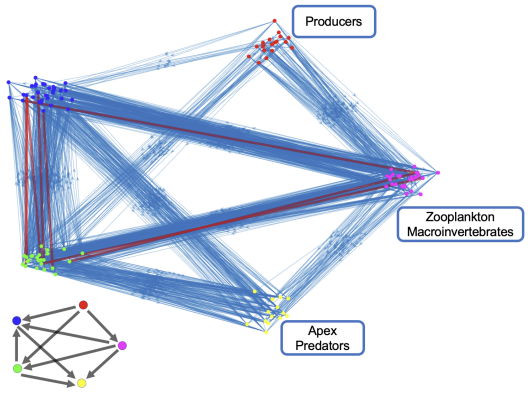


Fig. 5: Plot of vertices and edges for the Skew-RW clustering with $k = 5$ clusters on the 118 vertex subgraph from Li and Milekovic [6]. Edges oriented against the cluster hierarchy are in red and all other edges are in blue. The cluster hierarchy is red $\rightarrow$ magenta $\rightarrow$ green $\rightarrow$ blue $\rightarrow$ yellow. The matching bottom left graph, gives the cluster level orientations of edges. We label 3 clusters which exhibit consistent within cluster species labels. $l = 1$ for this run. The shown cluster is the best TopTF scorer over 100 runs.

structure are roughly interpretable as the trophic levels, or cluster level predator-prey relationships. Due to the nature of motif clustering Li and Milenkovic pruned the network of 10 vertices corresponding to 'singleton' clusters (manatee, kingfisher, hawksbill turtle, etc.) and detritus species. The resulting sub-network consists of 118 vertices and 1714 edges. Impressively, in their clustering only 5 edges are oriented from higher to lower clusters in the found hierarchy.

Our method uncovers a similar hierarchical structure. Skew-Symmetric Clustering with random walk normalization, Skew-RW, and $l = 1$ gives the best results out of all methods in terms of TopTF. All of the methods previously compared against via the DSBM are able to operate on the full 128 vertices of the FBFW graph. However, for direct comparison we generate Li and Milenkovic's subgraph and compare their reported clustering versus that returned by Skew-RW. Li and Milenkovic's clustering yields a TopTF score of 1536 meaning that $\approx 89.6\%$ of the 1714 edges are oriented between clusters according to the hierarchy. Skew-RW yields a maximum TopTF score of 1587, orienting about $\approx 92.6\%$ of edges between clusters according to the hierarchy. When computing TopTF we take all cluster-cluster relations into account (setting c in Eqn. 12).

The main difference between the Skew-RW clustering and that of Li and Milekovic is that our clustering has fewer within cluster edges, which of course do not contribute to the TopTF score. The most prominent example of this is that Skew-RW places the algae and seagrass species in the cluster lowest in the hierarchy while Li and Milekovic places them in the second lowest. In some sense, our clustering may be more intuitive. For example the species Drift Algae and Epiphytes, also an algae, have no incoming edges in the reduced digraph. While placing these species in the second lowest cluster does not introduce any edges oriented against the hierarchy, it does result in more within cluster edges, lowering the overall TopTF score. The clusters at the top of both hierarchies are identical. This top cluster contains species such as sharks and dolphins. We also note that our clustering orients 8 edges against the hierarchy. Our clustering is visualized in Figure 5.

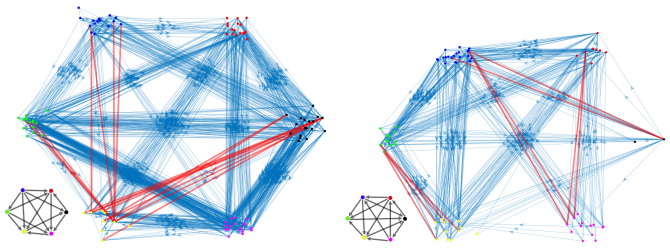
Figure 4 presents cut score results for $k = \{3, 4, 5, 6\}$. The methods which utilize the matrices $\mathbf{H}$ and $\mathbf{K}$ are generally most successful across all 3 computed cut metrics. Observe that Skew-R ($l = 1$) and Skew-RW ($l = 1$) are consistently two of the best performing algorithms in terms of cut scores. Running these algorithms with $k > 6$ did not increase, and often decreased, TopTF scores. DD-Sym-N refers to a normalized version of DD-Sym, see Satuluri and Parthasarathy [20] for details.

b) Other Food Webs: We briefly present results for two other food webs: Mangrove Wet Season and Cypress Dry Season, which are originally Pajek datasets[2]. Visualizations of output clusterings that achieve the highest TopTF score for the Hermitian Clustering algorithm with random walk normalization (Herm-RW) can be seen in Figure 6a and Figure 6b. The Herm-RW algorithm gives on average the highest TopTF scores for both of these graphs but the Skew algorithm is also able to recover the clustering which yields the highest found TopTF score. Again we observe that the method is able to uncover a clustering structure, with $k = 6$, that yields a high TopTF score and appears to reveal a cluster level hierarchical structure.

V. Conclusion

We have explored the role of complex-valued adjacency matrices for finding imbalanced cuts in directed graphs. Via

[2]http://vlado.fmf.uni-lj.si/pub/networks/data/bio/foodweb/foodweb.htm



(a) Visualization of the clustering output by Hermitian Clustering with random walk normalization on the Mangrove (wet season) data set. There are 16 edges, indicated in bold red, which actively detract from the TF score of 1104.

(b) Visualization of the clustering output by Hermitan Clustering with random walk normalization on the Cypress Dry Season graph. There are 10 edges oriented against the majority flows. Approximately 89% of the edges contribute to the TF score of 472.

Fig. 6: Additional food web visualizations

a careful analysis of algebraic relationships we show that real valued representation and algorithms which use real arithmetic are not only possible but advantageous. Our algorithm, Skew-Symmetric Clustering, is faster and requires asymptotically less memory than the existing state of the art method. It has a natural connection to a simple metric which captures the spirit of imbalanced cuts. We demonstrate the algorithm's ability to find meaningful patterns in real world data and outperform related methods on graphs generated from the DSBM.

In a broader sense we hope that this work will encourage careful consideration of the role of complex-valued representations for graphs. While our work primarily focuses on algorithmic drawbacks of using complex-valued representations, there are advantages for considering such matrices. For example Cucuringu et al. [4] use the Davis Kahan Theorem [24] for Hermitian matrices in their analysis and Laenen and Sun [7] use the fact that Hermitian matrices are subject to the min-max theorem. Some limitations of this work include focusing on a single, specific complex-valued digraph matrix limited to oriented graphs, the non-generality of the relaxation argument with respect to k, and a further results on larger, real-world graphs.

Acknowledgment

Koby Hayashi acknowledges support from the United States De- partment of Energy through the Computational Sciences Graduate Fellowship (DOE CSGF) under grant number: DE-SC0020347.

References

[1] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11222-007-9033-z

[2] D. Gleich, "Hierarchical directed spectral graph partitioning," *Information Networks*, 2006.

[3] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013, clustering and Community Detection in Directed Networks: A Survey. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157313002822

[4] M. Cucuringu, H. Li, H. Sun, and L. Zanetti, "Hermitian matrices for clustering directed graphs: insights and applications," *arXiv:1908.02096*, 2019.

[5] H. Van Lierde, T. W. S. Chow, and J.-C. Delvenne, "Spectral clustering algorithms for the detection of clusters in block-cyclic and block-acyclic graphs," *Journal of Complex Networks*, vol. 7, no. 1, pp. 1–53, 05 2018. [Online]. Available: https://doi.org/10.1093/comnet/cny011

[6] P. Li and O. Milenkovic, "Inhomogeneous hypergraph clustering with applications," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/a50abba8132a77191791390c3eb19fe7-Paper.pdf

[7] S. Laenen and H. Sun, "Higher-Order Spectral Clustering of Directed Graphs," *arXiv e-prints*, p. arXiv:2011.05080, Nov. 2020.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01. Cambridge, MA, USA: MIT Press, 2001, pp. 849–856.

[9] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *J. ACM*, vol. 51, no. 3, p. 497–515, may 2004. [Online]. Available: https://doi.org/10.1145/990308.990313

[10] R. Peng, H. Sun, and L. Zanetti, "Partitioning well-clustered graphs: Spectral clustering works!" in *Proceedings of The 28th Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, P. Grünwald, E. Hazan, and S. Kale, Eds., vol. 40. Paris, France: PMLR, 03–06 Jul 2015, pp. 1423–1455. [Online]. Available: https://proceedings.mlr.press/v40/Peng15.html

[11] B. Mohar, "A new kind of hermitian matrices for digraphs," *Linear Algebra and its Applications*, vol. 584, pp. 343–352, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0024379519304136

[12] S. Furutani, T. Shibahara, M. Akiyama, K. Hato, and M. Aida, "Graph signal processing for directed graphs based on the hermitian laplacian." in *ECML/PKDD (1)*, 2019, pp. 447–463.

[13] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. Hirn, "Magnet: A neural network for directed graphs," *arXiv preprint arXiv:2102.11391*, 2021.

[14] K. Guo and B. Mohar, "Hermitian adjacency matrix of digraphs and mixed graphs," *Journal of Graph Theory*, vol. 85, no. 1, pp. 217–248, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.22057

[15] J. Liu and X. Li, "Hermitian-adjacency matrices and hermitian energies of mixed graphs," *Linear Algebra and its Applications*, vol. 466, pp. 182–207, 2015.

[16] G. H. Golub and C. F. van Loan, *Matrix Computations*, 4th ed. JHU Press, 2013. [Online]. Available: http://www.cs.cornell.edu/cv/GVL4/golubandvanloan.htm

[17] W. Greub, *Linear algebra*, 3rd ed. Berlin,New York: Springer, 1967.

[18] I. Dokmanic, R. Parhizkar, J. Ranieri, and M. Vetterli, "Euclidean distance matrices: Essential theory, algorithms and applications," *Signal Processing Magazine, IEEE*, vol. 32, pp. 12–30, 11 2015.

[19] S. Laenen, "Directed graph clustering using hermitian laplacians," Master's thesis, 2019.

[20] V. Satuluri and S. Parthasarathy, "Symmetrizations for clustering directed graphs," in *Proceedings of the 14th International Conference on Extending Database Technology*, ser. EDBT/ICDT '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 343–354. [Online]. Available: https://doi.org/10.1145/1951365.1951407

[21] D. Sussman, M. Tang, D. Fishkind, and C. Priebe, "A consistent adjacency spectral embedding for stochastic blockmodel graphs," *Journal of the American Statistical Association*, vol. 107, 08 2011.

[22] A. Gates and Y.-Y. Ahn, "The impact of random models on clustering similarity," *Journal of Machine Learning Research*, vol. 18, 01 2017.

[23] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *CoRR*, vol. abs/1612.08447, 2016. [Online]. Available: http://arxiv.org/abs/1612.08447

[24] C. Davis and W. M. Kahan, "The rotation of eigenvectors by a perturbation. iii," *SIAM Journal on Numerical Analysis*, vol. 7, no. 1, pp. 1–46, 1970. [Online]. Available: https://doi.org/10.1137/0707001