Journal of Complex Networks (2022) **3**, Advance Access Publication on 15 June 2022
https://doi.org/10.1093/comnet/cnac013

Advances in scaling community discovery methods for signed graph networks

MARIA TOMASSO
Department of Computer Science, Texas State University, 601 University Ave, San Marcos, TX 78666, USA

LUCAS J. RUSNAK
Department of Mathematics, Texas State University, 601 University Ave, San Marcos, TX 78666, USA

AND

JELENA TEŠIĆ †
Department of Computer Science, Texas State University, 601 University Ave, San Marcos, TX 78666, USA
†Corresponding author. Email: jtesic@txstate.edu

Edited by: Tiago P. Peixoto

[Received on 19 October 2021; editorial decision on 1 April 2022; accepted on 11 April 2022]

Community detection is a common task in social network analysis with applications in a variety of fields including medicine, criminology and business. Despite the popularity of community detection, there is no clear consensus on the most effective methodology for signed networks. In this article, we summarize the development of community detection in signed networks and evaluate current state-of-the-art techniques on several real-world datasets. First, we give a comprehensive background of community detection in signed graphs. Next, we compare various adaptations of the Laplacian matrix in recovering ground-truth community labels via spectral clustering in small signed graph datasets. Then, we evaluate the scalability of leading algorithms on small, large, dense and sparse real-world signed graph networks. We conclude with a discussion of our novel findings and recommendations for extensions and improvements in state-of-the-art techniques for signed graph community discovery in real-world signed graphs.

Keywords: sign graph clustering; community discovery; sparse networks.

1. Introduction

The rise of social media interactions has illuminated an increasing necessity for a robust understanding of social network analysis (SNA), and social network theory has provided explanations for a variety of social phenomena ranging from individual creativity to corporate profitability. Community discovery has proven valuable in many areas of application, including detection of bot activity and fraud in criminology, identification of customer segmentation in marketing, characterization of *astroturfing* in political science, detection of cancers via diagnostic imaging and quantification of environmental hazards in public health [1]. In an era dominated by social media communication, the community detection tools developed specifically from social network theory can help researchers understand trends and propagation patterns within online communities [2].

Users are generally represented as vertices (nodes) in a graph, while edges are defined based on users' friendships and interactions with posts or re-posts; they are generally based on any social interaction on a media platform between users. A plethora of methods has been proposed to transform social media interactions in a simple graph network where edges exist along user interactions but do not exist if the connection is unknown. With the increased richness of social media interactions and additional information in the types of interactions that can exist in the social networks today (e.g. reviews, comments, shares, friends, blocked users), researchers have turned to richer interpretations of the edges in graph networks (signs, weights), recently turning their attention to the use of signed graph networks [3]. A community within a network is defined as a partitioning of nodes such that nodes within the same cluster are similar and usually strongly connected, while nodes in different clusters are dissimilar and usually weakly connected; in essence, similar nodes should be grouped together. In real datasets, community structure is almost always present to some degree [4]. Community detection in unsigned networks traditionally relies on the absence of connections between vertices (e.g. users) to determine if they belong in different communities. The presence of negative links provides affirmative evidence of their dissimilarity, allowing the use of richer signed network analysis for community detection. When negative edges are included in a network, we can study social dynamics and stability in respect to friendship and enmity in more depth [5, 6], or expand to new application domains such as the behaviour of the brain [7].

In this article, we present an overview of the work that has been done on community detection in signed networks to date. The methods are divided into two top level categories: methods adapted from unsigned methods and methods that work only for signed graphs, with additional subcategories. We then compare state-of-the-art methodologies on small signed networks with known ground-truth communities and compare their ability to recover the ground-truth labels based on edge signs alone. Finally, we evaluate the scalability in terms of effectiveness and efficiency of leading clustering methodologies on real signed networks [8] as they increase in complexity. Signed graph definitions are outlined in Section 1.1, while Section 2 describes unsigned clustering methods adapted for signed graphs, and Section 3 reviews novel methods that utilize signed graph characteristics such as balance or the random walk gap.

Prior surveys:

A comprehensive survey on mining techniques for signed graphs [9] includes several community detection methods. The survey had a much broader scope on signed graph analysis, from node ranking, classification and embedding, over link and sign prediction, to information diffusion and recommendations in signed graphs [9]. In the same year, a survey of spectral clustering methods for unsigned and signed graphs was published [10]. This survey provides a thorough overview of spectral methods and Laplacian variants for both unsigned and signed graphs. In this article, we focus on community detection in signed graphs and provide a comprehensive examination of spectral methods and non-spectral methods for community detection with respect to incremental development and suitability based on data characteristics.

Reference searching:

In the first stage of the literature review, both forward and backward reference searching were used to find significant contributions to the field. After forward and backward reference searching, a systemic literature review was conducted to find any publications on signed graph clustering that were previously missed. The following search terms were used: *'signed' AND 'graph' AND 'clustering'* and *'signed' AND 'graph' AND 'community' AND 'detection'*. All results were saved and manually reviewed for relevance.

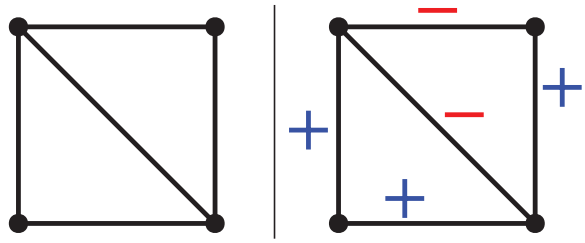

FIG. 1. Left: Unsigned graph with four vertices and five edges; Right: Signed graph with the same underlying unsigned graph: edges labelled as + or −.

1.1 *Signed graph definitions and methods*

A **graph** $\mathcal{G}$ consists of two disjoint sets: a set of *vertices* v , $v \in \mathcal{V}$ and a set of *edges* e, $e \in \mathcal{E}$. In this article, graph and network can be assumed to be synonymous. Graphs can be **directed** or **undirected**. In a directed graph, an edge may connect node i to node j without node j necessarily being connected to node i. In an undirected graph, if node i is connected to node j, then node j must be connected to node i. A graph can also be **weighted**, which means that each edge has a 'weight' attribute that can represent the strength of the connection. In a **signed** graph, edges are assigned +1 or −1 weights, as illustrated in Figure 1. In graph theory, a signed graph is **balanced** if the product of edge signs around every cycle is positive. In a **complete** graph, every pair of vertices is connected by an edge. A complete graph is **weakly balanced** if and only if it can be divided into multiple sets of mutual friends, with complete mutual antagonism between each pair of sets.

The *graph density* d of undirected graphs is the ratio of the number of edges e with respect to the maximum possible edges in a fully connected graph with v vertices; see Eq. 1.1.

$$d = \frac{2 \cdot e}{v \cdot (v-1)}. \tag{1.1}$$

A **dense** graph is a graph with a number of edges close to the maximum number of edges. With density scores of 0.483, 0.782 and 0.225, respectively, Highland Tribes [11], Sampson [12] and Correlates of War [13] are three examples of real-world signed and dense graphs. A **sparse** graph has very few edges relative to the number of nodes. Most social media networks have a high number of vertices (users) v and relatively small number of edges e as they are only connected to a small fraction of the overall community with density score 0.1 [8].

1.2 *Background*

A *community* in graph theory is a set (cluster) of nodes that are similar and generally densely connected to other nodes within the cluster and dissimilar and sparsely connected to those outside of the cluster relative to given graph or data metrics. However, in signed graph theory a *community* has the additional stipulation that the cluster is densely positive and sparsely negative within its connected component, and densely negative and sparely positive to the nodes outside the cluster. As a signed network graph innately represents expressed opinions between entities (vertices) through edge signs between them, the signed graph balancing model proved successful in social science in the 20th century. Social balance theory, described in Section 3.3, is a branch of signed graph theory proposed by [14] and developed

by [15] in the 1940s and 1950s. It allows certain well-behaved graphs to be 'perfectly' partitioned so that negative edges exist only between clusters, and positive edges exist only within clusters. In the real world, however, such well-behaved datasets are rare. Modularity-based metrics seek densely connected clusters using optimization techniques. Yang et al. introduced FEC, an algorithm for signed graphs that was designed for densely connected networks [16]. FEC treats the sign and density of edges as clustering attributes. Gomez *et al.* [17] refined modularity metrics and extended existing methods to signed graphs that were directed, weighted or contained loops. Both FEC and the Gomez method were designed for smaller signed graphs with dense connections: approaches were prohibitively slow even on our smallest networks, and fail to produce any results.

Prior to the explosion of SNA methods, all signed networks were assumed to be small and relatively densely connected; be it inter-personal relationships or warring counties such as the modelling of diplomatic relations in the Middle East [18], South Asia [19] and Allied and Axis powers during World War II [20]. Real world datasets and particularity social networks tend to follow power law degree distribution, as most nodes are only directly connected to a small percentage of the overall network and only few nodes are highly connected. While spectral clustering is a powerful technique for the detection of graph communities, the eigenvalues of signed graphs present a substantial obstruction in the development of a parallel spectral theory that is meaningful for the data. In [21], is it observed that these sparse graphs with a power-law degree distribution present two primary issues with unsigned spectral clustering: (1) the eigenvalues of a sparse network tend to spread, which can obscure the largest and smallest eigenvalues and makes the informative eigenvalues difficult to isolate; and (2) high heterogeneity in the degree distribution modifies the *i*th entry of the informational eigenvectors in proportion with the degree of node *i*, known as 'eigenvector pollution' by the authors [21].

In this article, we survey the signed graph community discovery methods of the 21st century. These techniques generally fall into two categories, and we explore them in the following sections. In Section 2, we review the signed graph adaptations from algorithms developed for unsigned graphs using discrete optimization techniques. We review novel methods that utilize signed graph characteristics such as balance or the random walk gap in Section 3. In Section 4, we provide a comprehensive study on 12 methods on real-world datasets of varying complexity and summarize the effectiveness each. Section 5 concludes with a discussion on timing and scalability to inform the creation of synthetic data to further test the algorithms.

2. Adaptations of unsigned spectral clustering methods to signed graph clustering

2.1 *Clustering for unsigned graphs*

Before examining methods for signed graphs, the algorithms developed for unsigned graphs must be understood. The simplest non-trivial case in undirected graph clustering is two-way partitioning. This involves separating the nodes of the graph into two groups such that nodes in the same group are strongly connected and nodes in opposite groups are weakly connected. To accomplish a two-way partitioning, two items are needed: (1) a criterion that defines a 'good' partition and (2) a method to efficiently optimize the criterion.

Criteria for two-way partitioning

Many criteria have been proposed for two-way graph partitioning. The first measure we will introduce for assessing two-way clustering of a graph is the graph cut. For an unsigned graph G with disjoint clusters X and Y, the *two-way graph cut* is defined as $cut(X, Y) = \sum_{i \in X j \in Y} A_{ij}$. The cut is essentially the number of

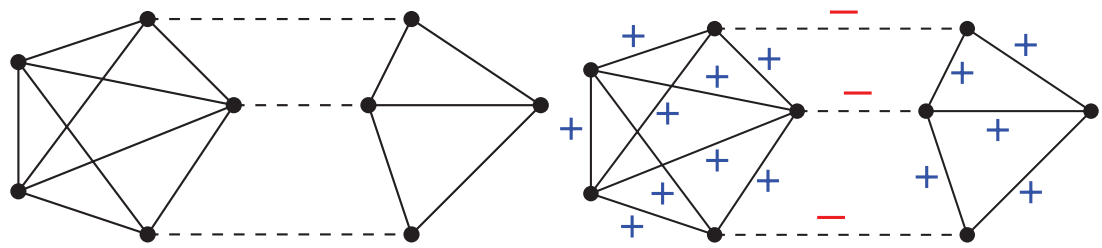

FIG. 2. (Left) Unsigned graph cut set, where cuts are dashed edges, and the criteria scores described in Section 2.1 are: $cut(X, Y) = 3$, $rcut(X, Y) = 1.35$ and $ncut(X, Y) = 0.933$; (right) An equivalent balanced signed graph cutset with edges labelled as $+$ or $-$. Dashed lines illustrate the ideal Harary cut.

edges or, for a weighted graph, the sum of weights between clusters. Since the typical goal of clustering is to group densely connected nodes together, choosing X and Y to minimize the cut is a good first step. Unfortunately, the cut does not account for the size of clusters, and the optimal solution can separate few or single vertices if applied as is. To remedy this, the *ratio cut* is introduced. For an unsigned graph G with disjoint clusters X and Y, the *two-way ratio cut* is defined as $rcut(X, Y) = cut(X, Y)(\frac{1}{|X|} + \frac{1}{|Y|})$, as illustrated in Fig. 2(left). The ratio cut takes the size of the clusters into consideration by minimizing the graph cut relative to the sizes of each cluster. Shi and Malik [22] refine the ratio cut to consider the strength of the connection of the nodes in X and Y to the rest of the graph with the *normalized cut*. For an unsigned graph G with disjoint clusters X and Y, the *two-way normalized cut* is defined as $ncut(X, Y) = cut(X, Y)(\frac{1}{vol(X)} + \frac{1}{vol(Y)})$, where $vol(P)$ represents all of the weights of all edges adjacent to nodes in the cluster P. By including volume in the normalized cut objective, the cut is minimized relative to both the size of the clusters and the connectivity of the graph.

Extending the criteria to k-way partitions

Real networks have more than two communities and a need for efficient k-way partitioning algorithms. The two-way partitioning algorithms provide a simple recursive technique to perform k-way partitioning [22]. First, partition the graph into two clusters. Then recursively run the two-way partitioning algorithm separately on the subgraph for each cluster. While this technique can be efficiently computed, it ignores the higher-order spectral information of the graph. As an alternative, k-way generalizations of the ratio cut and normalized cut have been introduced and are defined as follows: for an unsigned graph G with disjoint clusters X_1,...,X_k, the *k-way ratio cut* is defined as $rcut(X_1, ..., X_k) = \sum_{i=1}^{k} \frac{cut(X_j, \bar{X_j})}{|X_j|}$ and the *k-way normalized cut* is defined as $ncut(X_1, ..., X_k) = \sum_{i=1}^{k} \frac{cut(X_j, \bar{X_j})}{vol(X_j)}$. From [22], we know that two-way partitions can be solved efficiently for unsigned graphs. Unfortunately, the same is not true for k-way partitions, and finding a global optimum is NP-complete for most graphs. Thus, approximation methods are used to estimate solutions to the k-way criteria. Researchers initially tried to use greedy algorithms and gradient descent to find solutions to k-way clustering problems, but these approaches often failed to find a global optimum due to the high dimensionality of graph data and non-linearity of the criteria. As an alternative, Shi and Malik [22] developed a technique for approximating k-way normalized cut solutions by formulating them as generalized eigenvalue problems. This approach became known as spectral clustering; 20 years later, it is still considered to be a foundational algorithm in graph clustering.

2.1.1 *Spectral clustering* Spectral clustering begins by finding the Laplacian of the matrix representation of the network. Since several variants of the Laplacian exist, there are multiple versions of the

spectral clustering technique. After finding the Laplacian, the eigenvalues are computed. Note that the algorithm assumes that all eigenvalues of the Laplacian are non-negative (i.e. the Laplacian is positive semidefinite), and that the eigenvalues of the Laplacian can be efficiently computed. After the eigenvalues are found, they are plotted in increasing order, and the eigengap, the largest 'early' increase in sequential eigenvalues, is identified. The location of the eigengap provides options for the value of k, the number of communities in the graph. After identifying the number of clusters, k-means can be applied to cluster the communities [22]. The Laplacian matrix of an unsigned graph G is defined as $L = D - A$. D, the degree matrix, is a diagonal matrix such that the (i, i)th entry represents the degree of vertex v_i. A, the adjacency matrix, contains edge weight information such that entry (i, j) represents the weight of the edge between vertices v_i and v_j. If no such edge exists, the entry is 0. The spectral clustering algorithm is described in Alg. 1 and consists of four steps: (1) calculate the Laplacian L (or the normalized Laplacian); (2) calculate the first k eigenvectors (the eigenvectors corresponding to the k smallest eigenvalues of L); (3) consider the matrix U formed by the first k eigenvectors; the ith row defines the features of graph node i; (4) cluster the graph nodes based on u_i features using k-means clustering as outlined in Section 2.1.2 and in Algorithm 2. Since minimizing the normalized cut is NP-complete, the goal of the original and all subsequent spectral clustering algorithms is to find an approximate discrete solution efficiently. The two central problems of spectral clustering are the criterion that determines if a partition is 'good' and how partitions fitting the above criterion can be efficiently computed.

Laplacian variants The standard graph Laplacian matrix as defined in Section 2.1.1 is symmetric and positive semidefinite, meaning the eigenvalues are real and non-negative [23]. Additionally, two normalized variants of the Laplacian are commonly used in clustering, and they are defined as follows: the *symmetric normalized Laplacian* is $L_{sym} = D^{-1/2}LD^{-1/2}$, and the *random walk Laplacian* is $L_{rw} = D^{-1}L$. For undirected graphs, both L_{sym} and L_{rw} are positive semidefinite and have real, non-negative eigenvalues [23].

Eigenvalue computation is often expensive and prone to error for very large matrices, so if reasonable bounds on the problem are known (i.e. the maximum number of clusters), the problem can be reduced to finding the k smallest eigenvalues. The eigengap heuristic in spectral clustering indicates the number of clusters to use, but for noisy datasets the eigengap may be relatively small and difficult to detect. If the eigengap is large, however, the first k eigenvalues can be computed relatively efficiently through the use of Krylov subspaces or the power method [23].

Algorithm 1 Normalized spectral clustering [22]

Input: Adjacency matrix $A \in \mathbb{R}^{n\times n}$ of signed graph Σ; k number of clusters to construct:
Step 1: Compute the normalized Laplacian matrix $L_{n\times n}$ and diagonal matrix D of A.
Step 2: Compute the first k eigenvectors $l_1, ..., l_k$ corresponding to the k smallest eigenvalues of $L_{n\times n}$.
Step 3: Construct $U_{n\times k} \in \mathbb{R}^{n\times k}$ as the matrix containing the vectors $l_1, ..., l_k$ as columns.
Let u_i be the vector corresponding to the ith row of $U_{n\times k}$, $i = 1, ..., n$, $u_i \in \mathbb{R}^k$.
u_i defines the k-dimensional features of graph node i in $U_{n\times k}$, $i = 1, ..., n$.
Step 4: Cluster the graph nodes $i = 1, ..., n$ based on u_i features using k-means clustering described in Algorithm 2.
Output: Cluster labels $l = 1, ..k$ for all n nodes based on u_i vector closeness to final clusters centroids $C_1, ...C_k$.

2.1.2 *k-means clustering and weighted kernel k-means clustering* *The k-means algorithm* is used to partition a given set of observations into a predefined number (k) clusters. The algorithm starts with a

set of k centre-points and goes through multiple iterations to find optimal cluster centroids $C_1, ..., C_k$. Here, we present the k-means++ algorithms used in experimental comparisons in Sections 4 and 5. The k-means++ algorithm distributes the initial centroids over the given data to minimize the probability of bad outcomes[24] by a very simple randomized seeding technique, as illustrated in Algorithm 2.

Algorithm 2 k-means++ [24]

Step 1: Select centroids $C_1, C_2, ...C_k$ by taking uniformly a random data point from the data X and mark it as centroid C_1
for s **do**elect centroid $C_i, i \in [2, k]$
Choose C_i $C_i = x, max_{x \in X}(\frac{D(x)^2}{\sum_{x \in X} D(x)^2}), D(x) = min_{l \in [1, i-1]}(d(x, C_l))$
end for
Step 2: Iteratively compute new centroids for all data X
Iteration 0: $t = 0, t = limit$
while do$t \geq limit$
$t = 0$
for do$i, i \in [1, k]$
$S_i' = \left\{x_p : \left\|x_p - C_i\right\|^2 \leq \left\|x_p - C_j\right\|^2 \forall j, 1 \leq j \leq k\right\}$
Previous centroid value: $C_i' = C_i$
New centroid value: $C_i = \frac{1}{|S_i'|} \sum_{x_j \in S_i'} x_j$
$t = max(t, d(C_i, C_i'))$
end for
end while
Step 3: Assign $x, x \in X$ cluster label j so that $min_{i \in [1,...k]}(\left\|x - C_i\right\|^2) = \left\|x - C_j\right\|^2$

During each update step t in Algorithm 2, all observations x are assigned first to their nearest centre-point S_i^t. Next, the centre-points C_i^{t+1} are repositioned by calculating the mean of the assigned observations to the respective centre-points. As shown in Algorithm 2, this update process reoccurs until the centre-point update distance $d(C_i^{(t+1)}, C_i^{(t)})$ is smaller than the specified *limit*. There is only a finite number of possible assignments for the amount of centroids and observations available. As each iteration has to result in a better solution, the algorithm always ends in a local minimum. k-means++ approximately can be computed in $O(\log n)$ time [24].

Algorithm 3 Batch weighted kernel k means clustering [25]

Input: The kernel matrix K, the number of clusters k, and the weights for each input w, the initial clusters $\pi_1^{(0)}, \ldots, \pi_k^{(0)}$ (optional), and maximum number of iterations $t_{\max}$ (optional)
Step 1: Initialize the k clusters $\pi_1^{(0)}, \ldots, \pi_k^{(0)}$ arbitrarily if initial clusters were not provided.
Step 2: Set $t = 0$
Step 3: For each point a_i and every cluster c, compute the distance between a_i and the centroid of c as: $d(a_i, m_c) = K_{ii} - \frac{2\sum_{a_j \in \pi_c^{(t)}} w_j K_{ij}}{\sum_{a_j \in \pi_c^{(t)}} w_j} + \frac{\sum_{a_j, a_l \in \pi_c^{(t)}} w_j w_l K_{jl}}{(\sum_{a_j \in \pi_c^{(t)}} w_j)^2}$
Step 4: Find the updates index for each point a_i as $c^*(a_i) = \text{argmin}_c d(a_i, m_c)$ and update the clusters with $\pi_c^{(t+1)} = \{a : c^*(a_i) = c\}$
Step 5: If not converged and $t < t_{\max}$, increment t by 1 and go to Step 3.
Output: Cluster labels $\pi_1^{(t+1)}, \ldots, \pi_k^{(t+1)}$

The *weighted kernel k-means clustering* family of algorithms improves k-means clustering by introducing the weighted kernel approach, which maps the data to a higher-dimensional space and allows the separation of non-linear components [25]. Kernel function can be polynomial, Gaussian or Sigmoid, and the correct choice depends on target data characteristics. For weighted kernel k-means clustering, we need to choose the kernel matrix K first. If an input matrix is given, it is the weighted kernel matrix. If a standard graph partitioning objective is being used, we obtain the initial clusters using one of the following initialization methods: random, spectral, negative σ shift or METIS [26], a fast, multi-level graph partitioning algorithm that produces equally sized clusters. After we obtain initial clusters, we make kernel matrix K positive definite by adding to the diagonal. Finally, we oscillate between running batch weighted kernel k-means and incremental weighted kernel k-means (local search) until convergence. The sensitivity of the approach lies in the selection of the kernel matrix. The described spectral approach of finding eigenvectors, then performing clustering on features derived from eigenvectors, and its extensions and improvements proved to be highly effective on unsigned graphs. In the next section, we present the adaptations of unsigned graph clustering by applying spectral methods to signed graphs.

2.2 *Clustering for signed graphs*

Unsigned graph clustering methods described in Section 2.1 can be applied to signed graphs by either dropping all negative edges and hoping that the positive interactions produce the clusters, or all edges may be treated as positive, which returns the equivalent clustering of the underlying graph ignoring all sentiments; either way a lot of information is lost. In this section we present state-of-art modifications to signed graph clustering methods, and new methods that build upon unsigned graph clustering baseline.

2.2.1 *Modification of Laplacian matrix for signed spectral clustering* Spectral clustering assumes that all the eigenvalues of the Laplacian are *nonnegative* and *real*. The standard Laplacian matrix of a signed graph is indefinite [27] and will not yield real, non-negative eigenvalues. Moreover, accurate and efficient eigenvalue computation for large and sparse matrices is an open problem without signed graph extension. Thus, any method seeking to adapt spectral clustering to signed graphs must ensure that (1) eigenvalues are real and non-negative and (2) the new procedure is scalable to large networks. How do we compute a Laplacian for a signed graph and ensure that it is positive semidefinite? The Laplacian matrix introduced in [27] is indefinite, and modifications were made using the signed degree matrix, with the *signed Laplacian matrix* of a graph G as $\bar{L} = \bar{D} - A$, where $\bar{D}$ is the signed degree matrix given by $\bar{D}_{ii} = \sum_{j\sim i} |A_{ij}|$. Kunegis *et al.* [27] demonstrated that this signed Laplacian is positive semidefinite and, in some cases, positive definite, thus guaranteeing this Laplacian is a suitable basis for spectral clustering. Moreover, spectral clustering using the signed Laplacian is shown to be equivalent to the k-way signed ratio cut problem, which counts positive edges between clusters and negative edges within clusters [27]. A more natural signed graph Laplacian that possesses the expected relationship to its underlying incidence matrix as well as the signed-path property on the adjacencies was first presented in [28].

Symmetric normalized Laplacians tend to yield better results than unnormalized Laplacians for graphs with skewed degree distributions [27]. Kunegis *et al.* propose two ways of normalizing signed Laplacians. First, they define the random walk normalized Laplacian for signed graphs as $\bar{L}_{rw} = I - \bar{D}^{-1}A$ and show that the $\bar{L}_{rw}$ matrix is positive semidefinite [27]. Second, they define the symmetric normalized Laplacian for signed graphs as $\bar{L}_{sym} = \bar{D}^{-1/2}\bar{L}\bar{D}^{-1/2} = I - \bar{D}^{-1/2}A\bar{D}^{-1/2}$, where I is the identity matrix. The signed Laplacian matrix of a graph is positive-definite if and only if the graph is unbalanced [27]. For a signed graph G, the signed graph cut is given by $scut(G) = 2 \cdot cut^{+}(X, Y) + cut^{-}(X, X) + cut^{-}(Y, Y)$. The signed ratio cut is given by $SignedRatioCut(X, Y) = scut(X, Y)(\frac{1}{|X|} + \frac{1}{|Y|})$. The signed normalized cut is

given by $SignedNormalizedCut(X, Y) = scut(X, Y)(\frac{1}{vol(X)} + \frac{1}{vol(Y)})$, where $vol(X)$ and $vol(Y)$ represent the sum of the degrees of the nodes in X and Y, respectively.

The *balanced normalized signed Laplacian* is proposed by Zheng *et al.* [29] as an extension of the normalized signed Laplacian defined in [27], with an embedding map rather than an index of partitions. This yields additional information on the similarity between nodes rather than simply assigning cluster labels. Additionally, the authors argue that an embedding map is more likely to yield an approximate global solution rather than local optima. Zheng *et al.* take a two-step approach: (1) the Rayleigh quotient of the random walk normalized Laplacian is used as an objective function to achieve the embedding and (2) an objective function derived from the normalized signed cuts in [27] is used to complete clustering.

Geometric Laplacian means are proposed as a way to address shortcomings in [29] and its inability to recover ground-truth labels in real datasets. The arithmetic mean of the positive-edge and negative-edge Laplacians introduces noise to the embedding of the data points, and with the arithmetic mean the smallest eigenvectors of the Laplacian do not necessarily correspond to the smallest eigenvalues. The authors propose the use of the geometric mean of the positive-edge and negative-edge Laplacians to remedy these issues, although they concede that the geometric mean is more computationally expensive than the arithmetic mean and not well-suited to large, sparse networks [30]. The latest work modifies the Laplacian by combining the positive and negative Laplacians using the *matrix power means* [31]. This approach further improved the results in Section 4.

2.2.2 *Modification of k-way signed ratio cut criteria for signed clustering* Chiang *et al.* introduce the *balance normalized cut*, a criterion for k-way clustering problems that is analogous to the normalized cut [32]. The balance normalized cut is motivated by the failure of the signed k-way ratio cut (Eq. 2.1) to be minimized by any representation of partitions $\{x_1, \ldots, x_k\}$. Additionally, the k-way ratio cut (Eq. 2.1) inherently has less available information about each node than the 2-way signed ratio cut when $k > 2$. If there are only two clusters, c_1 and c_2, and we know that node i and node j both do not belong to c_1, then they both belong to c_2 and are therefore in the same cluster. However, if $k > 2$, we cannot infer that if two nodes are both excluded from one cluster, they must share another cluster. Without modification, minimizing the k-way signed ratio cut will not yield an optimal solution as proved by the authors [32].

$$\sum_{c=1}^{k} \frac{x_c^T \bar{L} x_c}{{}^T_c x_c} \tag{2.1}$$

Chiang *et al.* propose a series of new objectives that extended well to k-way partitioning [32]. In the following definitions, x_c represents the set of points assigned to cluster c; A, A^+ and A^- represent the full adjacency matrix, the positive edge-only adjacency matrix and the negative edge-only adjacency matrix, respectively; D, D^+ and D^- represent the diagonal matrix, the positive edge-only diagonal matrix and the negative edge-only diagonal matrix; and $L = D - A$, $L^+ = D^+ - A^+$ and $L^- = D^- - A^-$. The *positive ratio association* maximizes the number of positive edges within each cluster relative to the cluster's size, equal to the following:

$$\max_{\{x_1,\ldots,x_k\} \in I} \sum_{c=1}^{k} \frac{x_c^T A^+ x_c}{x_c^T x_c}. \tag{2.2}$$

The *negative ratio association* minimizes the number of negative edges within each cluster relative to the cluster's size:

$$\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T A^- x_c}{x_c^T x_c}. \tag{2.3}$$

The *positive ratio cut* minimizes the number of positive edges between clusters:

$$\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T L^+ x_c}{x_c^T x_c}. \tag{2.4}$$

The *negative ratio cut* maximizes the number of negative edges between clusters:

$$\max_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T L^- x_c}{x_c^T x_c}. \tag{2.5}$$

The *balance ratio cut* combines the positive ratio cut with the negative ratio association and simultaneously minimizes the number of positive edges between clusters while minimizing the number of negative edges within clusters:

$$\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^+ - A) x_c}{x_c^T x_c}. \tag{2.6}$$

The *balance ratio association* combines the negative ratio cut with the positive ratio association and simultaneously maximizes the number of positive edges within clusters while maximizing the number of negative edges between clusters:

$$\max_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^- + A) x_c}{x_c^T x_c} \tag{2.7}$$

The *balance normalized cut* is very similar to the balance ratio cut, except it normalizes the clusters by volume instead of number of nodes:

$$\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^+ - A) x_c}{x_c^T \bar{D} x_c}. \tag{2.8}$$

Similarly, the *balance normalized association* can be derived from the balance ratio association:

$$\max_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^- + A) x_c}{x_c^T \bar{D} x_c}. \tag{2.9}$$

Minimizing the balance normalized cut is equivalent to maximizing the balance normalized association. Thus, the choice between balance normalized cut and association is inconsequential [33]. Chiang *et al.*

proposed a multilevel framework that refines results by first dividing vertices into levels, and then applying their modified version of spectral clustering to each level. Here, the hierarchical approach to graph clustering increases algorithm scalability, and an 100 million edge graph is partitioned in under 4000 s [32].

2.2.3 *Modified generalized eigenvalue method for signed spectral clustering* Spectral methods in Section 2.2.2 use eigenvalues from one matrix. In this section, we describe the generalized eigenproblem methods that uses eigenvalues from two matrices, SPONGE [34]. The SPONGE algorithm is a method for k-way clustering in signed graphs that scales well to large graphs [34]. The objective is to decompose the network into disjoint groups, such that individuals within the same group are connected by as many positive edges as possible, while individuals from different groups are connected by as many negative edges as possible. The algorithm relies on a generalized eigenproblem formulation to find the k smallest eigenvectors before k-means clustering. The approach was inspired by constrained clustering [35], and the authors provide theoretical guarantees in the setting of a signed stochastic blockmodel [34]. For a given signed graph G, the objective function for SPONGE is derived from the normalized cut of the positive-edges subgraph $ncut(G^+)$ and the inverse normalized cut of the negative-edges subgraph $(ncut(G^-))^{-1}$. The trade-off and regularization parameters τ^+ and τ^- are introduced, and the previous metrics are merged into the new objective function in Eq. 2.10.

$$\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \frac{cut_{G^+}(C_i, \bar{C}_i) + \tau^- vol_{G^-}(C_i)}{cut_{G^-}(C_i, \bar{C}_i) + \tau^+ vol_{G^+}(C_i)}. \tag{2.10}$$

$C_1, \ldots, C_k$ represents a partitioning of G. The authors demonstrate that the prior objective function is equivalent to the discrete optimization problem in Eq. 2.11.

$$\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \frac{x_{C_i}^T (L^+ + \tau^- D^-) x_{C_i}}{x_{C_i}^T (L^- + \tau^+ D^+) x_{C_i}}. \tag{2.11}$$

The discrete optimization problem in Eq. 2.11 is NP-hard, so the authors relax the discreteness constraint on the x_{c_i}'s and allow solutions that are in $\mathbb{R}^n$. New vectors $z_1, \ldots, z_k \in \mathbb{R}$ are introduced such that $z_i^T (L^- + \tau^+ D^+) z_i = 1)$ and $z_i^T (L^- + \tau^+ D^+) z_i = 0 for i \neq j$, i.e., they are orthonormal with respect to $L^- + \tau + D^+$. Finally, the objective function can be rewritten as in Eq. 2.12.

$$\min_{z_i^T (L^- + D^+) z_j = \delta_{ij}} \sum_{i=1}^{k} \frac{z_i^T (L^+ + \tau^- D^-) z_i}{z_i^T (L^- + \tau^+ D^+) z_i}. \tag{2.12}$$

Objective function in Eq. 2.12 can be formulated as the generalized eigenproblem whose solution is given by the eigenvectors of $(L^- + \tau^+ D^+)^{-1/2}(L^+ + \tau^- D^-)(L^- + \tau^+ D^+)^{-1/2}$ [34]. The authors use LOBPCC [36] to solve for the eigenvectors corresponding to the k smallest eigenvectors and cluster the resulting node embedding using k means++ 2. The output of the k-means++ step is the final cluster labelling for the graph from the *SPONGE* procedure. Alternately, $SPONGE_sym$ uses the symmetric signed Laplacian, defined as $L_{sym}^{(+/-)} = (D^{(+/-)})^{-1/2} L^{(+/-)} (D^{(+/-)})^{-1/2}$ in the prior equations. *SPONGE* and $SPONGE_{sym}$ compared favourably against other leading signed spectral clustering algorithms in experiments performed by the authors, and we evaluate it further in Section 4.

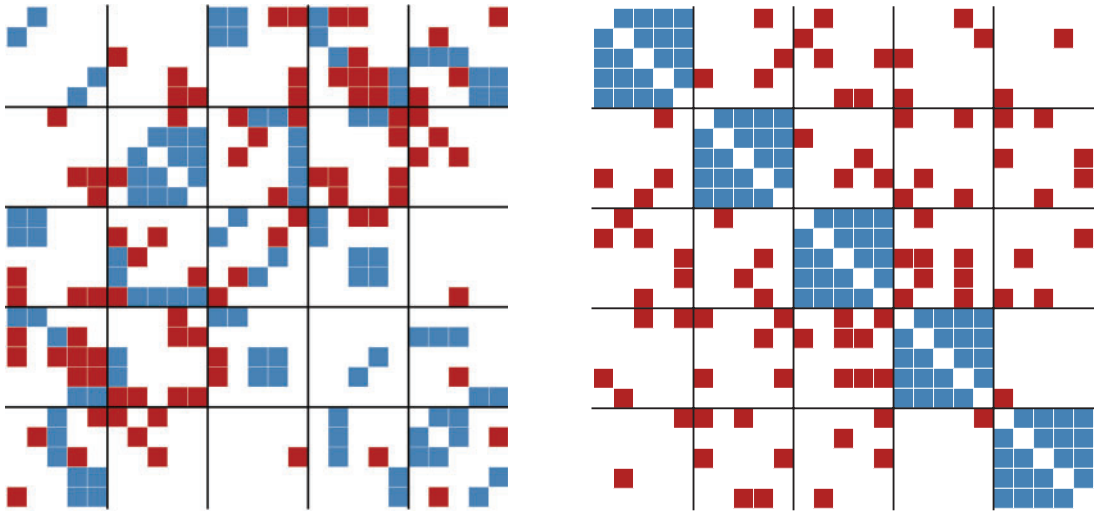

FIG. 3. Blockmodel illustration for two graphs with five community labels. Positive edges are blue (lighter shade) and negative edges are red (darker shade). Both blockmodel illustrations are sorted by assigned community labels: (a) random community assignment visualized with blockmodel illustration shows no clean community separation; (b) if the distinct communities are present in community assignment, it is visible in blockmodel as large positive connected component (blue (lighter shade) blocks).

3. Novel clustering methods for signed graphs

In this section, we review all clustering algorithms developed specifically for signed graphs. Early work in this field was heavily constrained by computational technology and focused on smaller, denser networks. While effective at the time, we note that some of these methodologies do not necessarily translate well to the large, sparse networks that are the focus of most modern research.

3.1 *Blockmodels*

Let S be a set and let $\{R_i\}_{i=1}^m$ be a set of binary relations on S. Individuals $a, b \in S$ are said to be **structurally equivalent** if and only if for any $c \in S$ and any $R_i \in \{R_i\}_{i=1}^m$, $aR_ic \iff bR_ic$ and $cR_ia \iff cR_ib$. In graph theory terms, the structural equivalence of two nodes means that both nodes are adjacent to exactly the same set of nodes with the same edge weights. Since this occurrence is very rare in real datasets, the authors used the concept of a blockmodel to relax the definition of structural equivalence. In a *blockmodel*, if the same permutation is applied to both the rows and the columns of the adjacency matrix, the underlying network structure is not changed. Blockmodels seek to *permute the data* in such a way that submatrices of all 0s exist within the adjacency matrix, as illustrated in Fig. 3. The adjacency matrix is then divided into blocks, with a block being assigned a value of 0 if all entries within it are 0 and 1 otherwise.

Nodes in the same block are assumed to be structurally equivalent if the value of the block is equivalent, thus relaxing the prior definition of structural equivalence to fit real-world datasets.

Lean fit

Breiger *et al.* [37] build on the concept of blockmodels to develop their system for clustering signed data. A blockmodel is said to be a *lean fit* to a matrix M if and only if there exists a permutation of M, yielding a permuted matrix M^* that can be blocked in such a way that (1) zeroblocks in M^* correspond to 0s in the

blockmodel, and (2) blocks in M^* containing at least one non-zero value have a corresponding value of 1 in the blockmodel. While a *lean fit* falls short of the algebraic definition of structural equivalence, the authors rationalize this decision by arguing that maintaining a social tie requires effort, while no work is required in the absence of a tie [37]. Thus, it is appropriate to assign any block with nonzero values an overall value of 1 in the blockmodel. The authors further emphasize that nonzero blocks do not need to be true cliques or fully connected subgraphs.

Limitations

For a graph with n vertices, there are $n!$ possible permutations (and thus blockmodels) of the vertices. The first limitation is that exhaustively checking all possible blockmodels quickly becomes impractical, even on relatively small graphs. The second limitation is that once a blockmodel is chosen, the blockings must still be enumerated and checked for a lean fit. The third limitation is the upper limit on number of blockings and the resulting clustering interpretation. The CONCOR (CONvergence of iterated CORrelations) algorithm repeatedly applies bipartitions to the raw data until a hierarchical clustering at the appropriate level of granularity is established [37]. Ordinary product moment correlation coefficients between the columns of the input matrix are computed at each iteration and stored in a correlation matrix. The process is repeated on the correlation matrix until convergence is reached, that is, there is no change in the matrix between iterations. Once convergence is reached, a clear bipartition emerges. The process is then repeated on each partition to identify sub-clusters. CONCOR does not optimize a specific metric; it exhaustively checks all possible blockmodels and checks blockings for a lean fit. This makes the algorithm prohibitively slow when applied to large and sparse signed graphs [37].

3.2 *Random walk models*

A *random walk* on a graph is a process that begins at a node and moves to one of the nodes to which it is connected. When the graph is unweighted, the node to which the walk moves is chosen uniformly at random among the neighbours of the present node. Harel *et al.* [38] introduced the random walk clustering algorithm for positively weighted edges in the graph. The method requires only $O(nlogn)$ time, and one of its variants needs only constant space [38]. The Fast Clustering for Signed Graphs (FCSG) algorithm employs a random walk gap approach to extract cluster structure information within the graph for positive edges only and for the entire graph [39]. A random walk gap is defined as the difference in cumulative transition probabilities between nodes in the positive-only subgraph versus the unsigned graph. The FCSG Algorithm 5 calls the RWG Algorithm 4 as a subroutine and uses the RWG matrix to reweigh the edges. Then, an iterative procedure is used to merge nodes connected by a positive edge until no positive edges remain. Nodes that have been merged together are assigned to the same cluster. The FCSG algorithm gives better results than existing algorithms based on the performance criteria of imbalance and modularity [39].

For Algorithm 4, the transition probabilities for the all-positive subgraph are normalized using the unsigned version of the input graph. If nodes i and j are not connected by a path with a length less than or equal to k, the k-step transition probability is zero. The matrix is D is defined as $D = ((H_{ij}^{G''} - H_{ij}^{G'})/H_{ij}^{G'})_{n \cdot n}$ where $H_{ij}^{G''}$ and $H_{ij}^{G'}$ represent the k-step transition probabilities between i and j on the all-positive subgraph and unsigned version of the input graph respectively. If $H_{ij}^{G'} = 0$, the normalized transition probability is undefined.

Random walk gap (RWG) algorithm is outlined in Algorithm 4, and its underlying assumption is that *the positive-only subgraph of the network must be a single connected component*. This places a

Algorithm 4 RWG algorithm [39]

Input: The adjacency matrix W of a signed graph
Step 1: Compute one step transition probabilities θ' and θ''
Step 2: Compute k-step transition probabilities $H'^{(k)}$ and $H''^{(k)}$
Step 3: Compute the sum of transition probabilities $H^{G'}$ and $H^{G''}$
Step 4: Compute normalized transition probabilities G
Step 5: Adjust the normalized transition probabilities D to generate D^*
Step 6: Generate the RWG matrix H
Output: RWG matrix H

large constraint on running the algorithm on a real dataset. If it is not a single connected component, clustering will only be performed on the greatest connected component, and all nodes outside of the greatest connected component will not be placed into a cluster. This condition is usually not met, and it results in many vertices being left out in experiments described in Sections 4 and 5.

Algorithm 5 FCSG algorithm [39]

Input: A RWG matrix H and graph G
Create a new weighted signed graph, G^* with weights W^*, by updating the weights of G using the following formula: $W^* = (w^*_{ij})$ where $w^*_{ij} = w_{ij} x h_{ij}$ if $(i,j) \in E^+$
while there are positive edges in G^* **do**
Select the edge (i,j) with the greatest weight
Let $i' = \min(i,j)$
Fuse i and j into a single node i'
Merge the edges that linked to both i and j and shared a common node. The weight of the new edge is the sum of the weights.
end while
All points that have been merged are labelled as a cluster
Output: Cluster labels $C_1, ...C_k$

The first significant limitation of the proposed Algorithm 5 for signed graphs is the underlying assumption that the spanning tree can be constructed using *only positive edges* in the signed graph. The algorithm starts from the premise that, for some values $\alpha \in [0,1]$ and $d > 0$, if node i and node j belong to the same cluster and $dist(i,j) \leq d$, then the probability that a random walk originating at i will reach j before leaving the cluster is at least α. While this principle is sound for unsigned graphs, it cannot be easily generalized to signed graphs, as its underlying assumption is that the spanning tree can be constructed using *only positive edges* only in the signed graph. To satisfy this requirement, we have to take the greatest connected component of the all-positive subgraph of the input signed graph. Figure 5(centre) illustrates two all-positive subgraphs for Highland Tribes, and it is clear there is no all-positive spanning tree. As a result, four vertices in a community (in blue) are left out of the analysis. In summary, the first limitation leads to some vertices being unlabelled in the final analysis, as shown in Section 5.

The second significant limitation is the assumption of the *small world hypothesis*, a theory that most users are linked by no more than five degrees of separation in a social network. This becomes critical in Step 4 of Algorithm 4. The authors assume that $H^{G'}_{ij} > 0$ for $k \geq 5$ due to the small-world hypothesis,

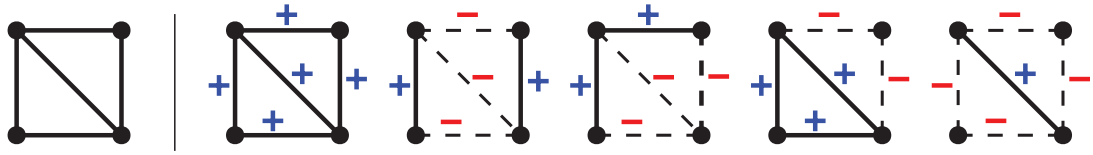

FIG. 4. For an underlying unsigned graph with four vertices and five edges in (a), there are five different balanced signed graphs in (b). Algorithm 6 can be applied to these five signed graphs. We remove the negative (dashed) edges from the fully balanced graph, and the result is a bi-cut set of clusters with positive (solid) edges only.

but, for graphs with a diameter exceeding 5, this does not hold. For this reason, the parameter used in the random walk gap matrix calculation L must be greater than or equal to the diameter of the all-positive subgraph of the input graph. The authors recommend that L be set to 5 and warn that the algorithm begins to degrade in quality if L is greater than or equal to 10.

3.3 *Heider balance theory based methods*

3.3.1 *Clustering for balanced signed graphs* Social balance theory was introduced by Heider in 1946 [14] and mathematically formalized by Cartwright and Harary in 1956 [15]. A signed graph is **balanced** if and only if (1) all of its edges are positive or (2) the nodes can be partitioned into two distinct clusters so that all edges within a cluster are positive and all edges between clusters are negative. Such a partition is known as a **Harary cut**, as illustrated in Fig. 4 for a balanced signed graph with four nodes and five edges. The clustering approach for the *balanced signed graphs* reduces to a two-step process as illustrated in Algorithm 6.

Algorithm 6 Clustering a balanced signed graph

Input: A balanced signed graph Σ
Step 1: Apply a Harary-cut and partition the signed graphs into two sets.
Step 2: Apply unsigned spectral clustering to each of the subsets.
Output: distinct signed graph clusters

While clustering adaptation for balanced signed graphs is simple, balanced graphs are rare in real-world data. A Harary cut provides a natural cut to examine clusters with similar sentiments. A new robust signed graphic generalization of normalized cuts using nearest Harary cuts recently appeared in [40].

3.3.2 *Clustering for weakly balanced signed complete graphs* A *fully connected* network (complete graph) is **weakly balanced** if and only if (1) all of its edges are positive or (2) the nodes can be partitioned into k distinct clusters so that all edges within a cluster are positive, and all edges between clusters are negative [41]. If a complete graph is *balanced* or *weakly balanced*, it is said that an underlying community structure exists in the graph. The signed clustering extension for *weakly balanced graphs* is outlined in Algorithm 7.

Note that weakly balanced partitioning reduces to Algorithm 6 for $k = 2$, as illustrated in Fig. 2(right). Weakly balanced graphs are rare within social networks, and identifying the network beyond a special case is computationally prohibitive. We require clustering techniques that can be applied to signed graphs in any state of balance.

Algorithm 7 Clustering a weakly balanced signed graph [41]

Input: Weakly balanced signed graph Σ
Step 1: Group vertices so that positive edges are within clusters and negative edges are between k clusters.
Step 2: Apply unsigned spectral clustering to each of the subsets.
Output: distinct signed graph clusters

3.3.3 *Augmentation-induced cluster balance for clustering* Hseih *et al.* add new edges between unconnected nodes to achieve balance. After creating a fully connected, maximally balanced graph based on the initial data, the eigenvectors corresponding to the k greatest eigenvalues of the completed adjacency matrix are computed. Finally, k-means clustering is run on the eigenvectors to assign nodes to clusters [42].

Algorithm 8 Clustering via matrix completion [42]

Input: A signed matrix G and number of clusters k Σ
Step 1: Impute the sign of missing edges in a way that minimizes frustration of the complete graph $\hat{G}$.
Step 2: Find the eigenvectors $l_1, ..., l_k$ corresponding to the k greatest eigenvalues of the adjacency matrix of $\hat{G}$.
Step 3: Construct $U_{n\times k} \in \mathbb{R}^{n\times k}$ as the matrix containing the vectors $l_1, ..., l_k$ as columns.
Step 4: Cluster the graph nodes $i = 1, ..., n$ based on u_i features using k-means clustering described in Algorithm 2.
Output: Cluster labels for all n nodes.

3.3.4 *Semi-supervised signed network clustering* Semi-supervised signed network clustering approach SSSnet [43] uses modified SNA and triangle balancing heuristics ('friend of my friend is my friend') [6] to address the issue for cluster discovery based on a modified version of Heider balance theory [14]. The authors of SSSnet transform the paradigm that 'the enemy of my enemy is my friend' and assert that the relationship should be neutral. First, a signed mixed-path aggregation scheme is used to create the node embedding. Next, the node embedding is used to generate cluster assignment probabilities, and clustering is achieved by training with a weighted sum of a supervised and unsupervised loss function and the unsupervised loss function is a probabilistic balanced normalized cut. SSSnet produces more robust results when the labelled data and node input features are available in the training step. If node input features are not available, SSSnet constructs the node features from the graph structure [43]. In this paper, we focus on comparing community discovery methods that are retrieve community information *solely* on the signed graph structure. In that light, we do not consider external vertex features for SSSnet implementation in Sections 4 and 5. We also do not use SSSnet data-driven training step to optimize the supervised loss function to a specific dataset in Sections 4 and 5. We compare the performance of SSSnet to the other unsupervised methods based solely on the information SSSnet can retrieve from signed graph structure. Our analysis of SSSnet extends to the community and ground truth recovery SSSnet can retrieve in an unsupervised manner from the previously unseen signed graphs that have no external vertex features.

3.3.5 *Graph clustering in balance feature space* Sharma *et al.* [44] study the network design problem of maximizing balance of a target community given a fixed number of edge-deletions. They demonstrate the NP-hardness of this problem while also exhibiting that it is also non-monotone and non-submodular. These computational issues were overcome using the spectral relation of balance with the Laplacian spectrum of the network. Since the spectral approach lacks approximation guarantees, a greedy approach was also implemented with bounds on the approximation quality. The bounds are achieved using pseudo-submodularity, and the effectiveness was established on sample dataset. Optimized nearest balanced states were characterized by Rusnak and Tešić in [40] with the introduction of the *frustration cloud* which relaxes the NP-hardness of determining the frustration index to provide additional context on the likelihood a consensus balanced state could be reached from a given signed graph. Exact values for edge deletions are computed, but instead of being deleted, they are trained to change sentiment to report back a balanced state. These balanced states are then aggregated over statistically significant samples to quantify the change a vertex or an edge would contribute to a consensus decision. Graph balancing was recently explored as an alternative to spectral clustering [40, 45]. The concept of the frustration cloud builds on the graph balance theory, relaxes the notion of a single frustration index to a family of minimally balanced graphs of a given signed graph and derives the numeric features of the vertex, status, and influence in a signed graph based on the balance theory [40].

The *status* of a vertex is the likelihood a vertex will appear in the majority over all sampled balancings, while the *influence* of a vertex is the likelihood the edges incident to the vertex will appear in the majority. Thus, influence is always less than or equal to status and, when plotted against each other they appear in between the status axis and the line $y = x$. It was shown in [40] that these two metrics are very different where status can detect 'promotability' while influence can detect those making the decisions regarding promotion. The authors have also demonstrated that status and influence attributes capture the spectrum of signed spectral clustering (Fig. 5(right)) and indicate a possible direction for moving away from eigenvector computation. This is accomplished by sampling spanning trees to detect a minimal set

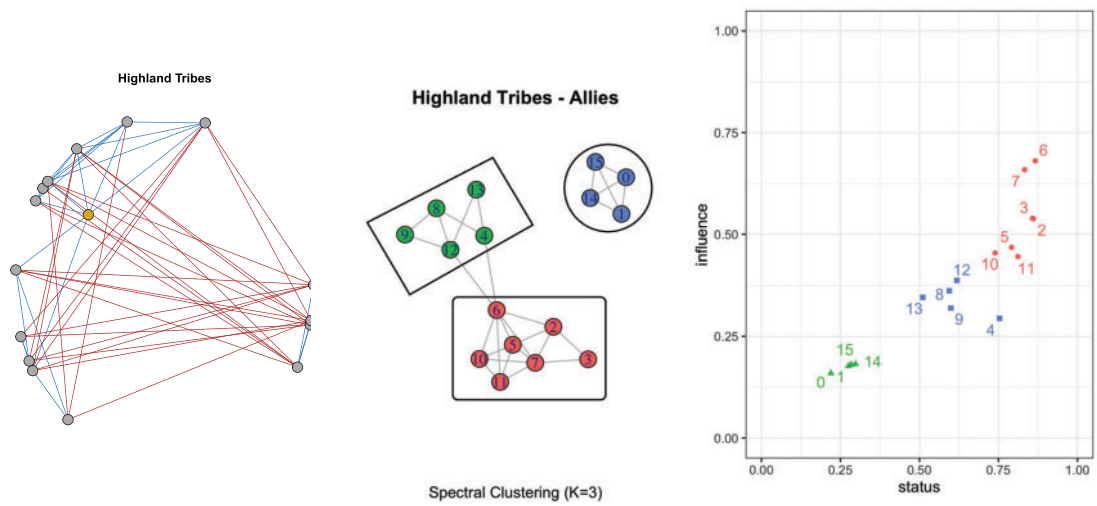

FIG. 5. Highland Tribes (**highland**) network has 16 vertices, 29 positive edges, 29 negative edges and 3 communities as shown in the entire network (left); positive only subnetwork with ground truth groupings (centre); graph vertex correspondence of Laplacian symmetric (lap_sym) method [27] to graphB clustering in status/influence space [40].

TABLE 1 *Legend: indexing 12 methods in Tables 2–9 and Fig. 11*

Table label		Figure label	Method description
ground	truth		Ground truth community labelling
Laplacian	_none	lap_none	Spectral clustering using the signed graph Laplacian [46]
	_sym	lap_sym	Spectral clustering using the symmetric Laplacian [46]
	_sep	lap_sep	Spectral clustering using the symmetric separated Laplacian [46]
Balanced Cuts	_none	BNC_none	Balanced normalized cuts [32]
	_sym	BNC_sym	Symmetric balanced normalized cuts [32]
SPONGE	_none	SPONGE_none	Baseline SPONGE implementation [34]
	_sym	SPONGE_sym	Symmetric SPONGE [34]
Power Means	GM	GM	Geometric means [30]
	SPM	SPM	Matrix power means [31]
FCSG		FCSG	Fast clustering for signed graphs [39]
SSSnet		SSSnet	SSSnet: semi-supervised signed network clustering [43]
graphB_km		graphB_km	k-means clustering in graph Balancing space [40]

of signs that obstruct balance, thus leveraging the difference between underlying bases of balance and unbalanced signed graphs. The study on more degenerate, adversarial networks is necessary as the next step to determine if consensus-based attributes [40, 45] can provide insight into networks where spectral clustering fails.

4. Signed graph clustering for known communities: a comparison

In this section, we compare the effectiveness, strengths and weaknesses of *twelve* state-of-the-art signed graph clustering approaches when ground-truth is available for five different datasets. Each of the 12 approaches listed in Table 1 and the high-level attributes of each of the datasets are listed in Table 8. Detailed findings on the signed clustering performance and the description of each of the dataset used for comparison are presented in Sections 4.1, 4.2, 4.3 and 4.4. We conclude the section with a discussion on method performances, taking labelling and signed graph attributes in consideration over all methods and all datasets in Section 4.5.

Approach

We have applied *twelve* different signed graph clustering methods on four datasets. Tables 2–9 and Fig. 11 use the indices for the methods outlined in Table 1. The Python package *signet* [47] was used to run Laplacian, Balanced Cuts and SPONGE methods; see Table 1 for the list of methods. Power Means [48, 49], SSSnet [43, 50] and graphB [51] implementations were provided by the authors as an open source.

Fast clustering for signed network implementation and its limitations are described in Section 3.2. The authors did not provide an implementation, and the paper did not discuss efficiency strategies [39]. Our in-house implementation follows the paper guidelines when possible, as outlined in Algorithms 4 and 5. RWG algorithm has underlying assumption that *the positive-only subgraph of the network must be a single connected component*. This places a large constraint on running the algorithm on a real dataset

TABLE 2 *Detailed analysis of ARI for all 12 methods for* **highland** *dataset. 75% of the methods entirely recover the ground truth*

highland		ground truth	Laplacians _none	Laplacians _sym	Laplacians _sep	Balanced cuts _none	Balanced cuts _sym	SPONGE _none	SPONGE _sym	Power Means GM	Power Means SPM	FCSG	SSS net	graphB _km
ground	truth	1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
	_none	1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
Laplacian	_sym	1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
	_sep	0.26	0.26	0.26	1	0.26	0.26	0.26	0.26	−0.06	0.13	0.26	0.26	0.26
Balanced	_none	1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
Cuts	_sym	1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
SPONGE	_none	1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
	_sym	1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
Power	GM	0.4	0.4	0.4	−0.06	0.4	0.4	0.4	0.4	1	0.42	0.4	0.4	0.4
Means	SPM	0.78	0.78	0.78	0.13	0.78	0.78	0.78	0.78	0.42	1	0.78	0.78	0.78
FCSG		1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
SSSnet		1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1
graphB_km		1	1	1	0.26	1	1	1	1	0.4	0.78	1	1	1

TABLE 3 *Detailed analysis of ARI for all 12 methods for* **sampson** *dataset. We have marked* **best** *ARI and second best ARI for recovering ground truth (first row and first column). We emphasize* **mutual** *ARI scores higher than ground truth recovery*

sampson		ground truth	Laplacians _none	Laplacians _sym	Laplacians _sep	Balanced Cuts _none	Balanced Cuts _sym	SPONGE _none	SPONGE _sym	Power Means GM	Power Means SPM	FCSG	SSS net	graphB _km
ground	truth	1	**0.61**	0.33	0.25	0.51	0.51	0.52	0.6	0.32	0.37	0.31	0.13	0.41
	_none	**0.61**	1	0.31	0.4	**0.69**	0.39	**0.74**	**0.7**	0.3	0.35	0.18	0.38	0.4
Laplacian	_sym	0.33	0.31	1	0.19	0.5	0.4	0.27	0.51	0.52	**0.89**	0.57	0.24	0.27
	_sep	0.25	0.4	0.19	1	0.26	0.26	0.52	0.25	0.25	0.19	0.18	0.4	0.41
Balanced	_none	0.51	0.7	0.5	0.26	1	0.48	0.54	**0.72**	0.27	0.53	0.6	0.14	0.18
Cuts	_sym	0.51	0.39	0.4	0.26	0.48	1	0.31	0.41	0.49	0.4	0.45	0.11	0.43
SPONGE	_none	0.52	**0.74**	0.27	0.52	0.54	0.31	1	**0.62**	0.22	0.31	0.33	0.21	0.53
	_sym	0.6	**0.7**	0.51	0.25	**0.72**	0.41	**0.62**	1	0.18	0.55	0.36	0.23	0.3
Power	GM	0.32	0.3	0.52	0.25	0.27	0.49	0.22	0.18	1	0.41	0.31	0.16	0.33
Means	SPM	0.37	0.35	**0.89**	0.19	0.53	0.4	0.31	0.55	0.41	1	0.54	0.28	0.24
FCSG		0.31	0.4	0.57	0.18	0.6	0.45	0.33	0.36	0.31	0.54	1	0.19	0.23
SSSnet		0.13	0.18	0.24	0.4	0.14	0.11	0.21	0.23	0.16	0.28	0.19	1	0.24
graphB_km		0.41	0.38	0.27	0.41	0.18	0.43	0.53	0.3	0.33	0.24	0.23	0.24	1

and reduces FCSG scores. We implemented FCSG in a Python + NetworkX package and have released a python wrapper we developed to efficiently compare methods and produce ARI matrices for all methods [52]. Note that the reproducibility of the reported results differs from the original paper.

Datasets

We evaluate the performance of the 12 signed clustering methods on the following five datasets: (1) **highland** [11] models agreeable and antagonistic relationships between tribes in the Eastern Central

TABLE 4 *Detailed analysis of ARI for all 12 methods for the* ***cow*** *dataset. We have marked* **best** *ARI and second best ARI for recovering ground truth (first row and first column). Note that most of* **mutual** *ARI scores are better than ground truth recovery, and we emphasize top* **mutual** *ARI scores higher than ground truth recovery for each method*

cow		ground truth	Laplacians _none	Laplacians _sym	Laplacians _sep	Balanced Cuts _none	Balanced Cuts _sym	SPONGE _none	SPONGE _sym	Power Means GM	Power Means SPM	FCSG	SSSnet	graphB_km
ground	truth	1	0.06	0.12	-0.02	0.02	0.12	0.06	0	0.06	0.06	−0.01	**0.22**	0.1
	_none	0.06	1	0.72	0.36	0.84	0.04	**0.91**	0.8	1	0.65	0.84	−0.01	0.65
Laplacian	_sym	0.12	0.72	1	0.34	**0.84**	0.17	0.68	0.74	0.72	0.71	0.77	0.13	0.71
	_sep	−0.02	0.36	0.34	1	0.38	0.38	0.36	**0.44**	0.36	0.38	0.33	0.03	0.4
Balanced	_none	0.02	0.84	0.84	0.38	1	0.03	0.77	0.87	0.84	0.69	**0.92**	0.02	0.66
Cuts	_sym	0.12	0.04	0.17	**0.38**	0.03	1	0.06	0.1	0.04	0.14	0	0.16	0.25
SPONGE	_none	0.06	**0.91**	0.68	0.36	0.77	0.06	1	0.74	**0.91**	0.67	0.77	0.01	0.65
	_sym	0	0.8	0.74	0.44	0.87	0.1	0.74	1	0.8	0.65	**0.88**	0.03	0.65
Power	GM	0.06	1	0.72	0.36	0.84	0.04	**0.91**	0.8	1	0.65	0.84	−0.01	0.65
Means	SPM	0.06	0.65	**0.71**	0.38	0.69	0.14	0.67	0.65	0.65	1	0.64	0.06	0.62
FCSG		−0.01	0.84	0.77	0.33	**0.92**	0	0.77	0.88	0.84	0.64	1	0	0.62
SSSnet		**0.22**	−0.01	0.13	0.03	0.02	0.16	0.01	0.03	−0.01	0.06	0	1	0.1
graphB_km		0.1	0.65	**0.71**	0.4	0.66	0.25	0.65	0.65	0.65	0.62	0.62	0.1	1

TABLE 5 *Percentage of positive edges in the detected communities (pow_in) and negative edges outside the detected communities (neg_out) per signed clustering method for* **cow** *dataset*

Method	ground truth	Laplacians _none	Laplacians _sym	Laplacians _sep	Balanced Cuts _none	Balanced Cuts _sym	SPONGE _none	SPONGE _sym	Power Means GM	Power Means SPM	FCSG	SSSnet	graphB_km
pos_in	0.52	0.98	**0.99**	0.94	1.0	0.99	0.97	1.0	0.98	0.97	0.92	0.51	0.98
neg_out	0.88	0.2	**0.88**	0.73	0.55	0.78	0.2	0.3	0.2	0.88	0.20	0.88	0.58

Highlands of New Guinea in Section 4.1(2) **sampson** [12], which models sentiment over time between novice monks in a New England monastery captured by Sampson [12]; (3) CoW [13] captures Second World War Allies, among 50 nations from Correlates of War data in Section 4.3; (4) **football** [53] tracks Twitter interactions between players belonging to the English Premier League clubs in Section 4.4; and (5) **olympics** models Twitter interactions between athletes competing in 2012 London Olympics [53]. Five datasets are selected as the ground labels are known, and they vary in number of community labels, number of vertices, edges and percentage of negative edges. Table 8 summarizes all five graph characteristics: number of vertices, positive, negative edges and vertex degree statistics. Table 8 also summarizes graph attributes: density score d (Eq. 1.1) and the number of balanced triangles over the total number of triangles in the graph bal_3. Finally, we have included information about the ground truth labelling in Table 8: the number of communities; pos_in—the percentage of positive edges in the ground truth communities; and neg_out—the percentage of negative edges between ground truth communities.

TABLE 6 *Detailed analysis of ARI for all 12 methods for the Football dataset with emphasis on* **best** *ARI and second best ARI for recovering ground truth (first row and first column). One* **mutual** *ARI method score is better than ground truth recovery, and we observe a great variations in ARI scores per methods*

Football		Ground truth	Laplacians _none	Laplacians _sym	Laplacians _sep	Balanced Cuts _none	Balanced Cuts _sym	SPONGE _none	SPONGE _sym	Power Means GM	Power Means SPM	FCSG	SSSnet	graphB _km
Ground	_truth	1	0.03	**0.76**	0.09	0.68	0.49	0.74	0.71	0.08	0.27	0	0.7	0.01
	_none	0.03	1	0.03	0.01	0.02	0.09	0.04	0.04	0.05	0.07	0.18	0.04	0.01
Laplacian	_sym	**0.76**	0.03	1	0.09	0.68	0.48	**0.84**	0.7	0.08	0.28	0	0.66	0.01
	_sep	0.09	0.01	0.09	1	0.09	0.09	0.08	0.12	0.01	0.04	0	0.1	0
Balanced	_none	0.68	0.02	0.68	0.09	1	0.46	0.65	0.66	0.07	0.28	0	0.63	0.03
Cuts	_sym	0.49	0.09	0.48	0.09	0.46	1	0.48	0.51	0.07	0.23	0.03	0.46	0.01
SPONGE	_none	0.74	0.04	**0.84**	0.08	0.65	0.48	1	0.67	0.07	0.27	0.01	0.66	0.02
	_sym	0.71	0.04	0.7	0.12	0.66	0.51	0.67	1	0.08	0.27	0.01	0.64	0.01
Power	GM	0.08	0.05	0.08	0.01	0.07	0.07	0.07	0.08	1	0.15	-0.01	0.08	0
Means	SPM	0.27	0.07	0.28	0.04	0.28	0.23	0.27	0.27	0.15	1	-0.01	0.25	0
FCSG		0	0.18	0	0	0	0.03	0.01	0.01	-0.01	-0.01	1	0.01	0
SSSnet		0.7	0.04	0.66	0.1	0.63	0.46	0.66	0.64	0.08	0.25	0.01	1	0.01
graphB_km		0.01	0.01	0.01	0	0.03	0.01	0.02	0.01	0	0	0	0.01	1

TABLE 7 *Detailed analysis of ARI for all* 12 *methods for* **olympics** *dataset with emphasis on* **best** *ARI and second best ARI for recovering ground truth (first row and first column). Olympics data show great variations in ARI scores per method*

olympics		Ground truth	Laplacians _none	Laplacians _sym	Laplacians _sep	Balanced Cuts _none	Balanced Cuts _sym	SPONGE _none	SPONGE _sym	Power Means GM	Power Means SPM	FCSG	SSSnet	graphB _km
Ground	truth	1	0.02	0.72	0.21	0.3	0.11	0.64	**0.85**	0.38	0.46	0.03	0.8	0.05
Laplacian	_none	0.02	1	0.02	0.02	-0.05	0.22	0.03	0.03	0.06	0.06	0.14	0.03	0
	_sym	0.72	0.02	1	0.18	0.25	0.08	0.68	0.7	0.31	0.38	0.02	0.66	0.05
	_sep	0.21	0.02	0.18	1	0.08	0.06	0.17	0.2	0.15	0.16	0.01	0.2	0.02
Balanced	_none	0.3	−0.05	0.25	0.08	1	0.02	0.21	0.28	0.19	0.23	−0.06	0.28	0.04
Cuts	_sym	0.11	0.22	0.08	0.06	0.02	1	0.09	0.11	0.14	0.13	0.3	0.12	0.01
SPONGE	_none	0.64	0.03	0.68	0.17	0.21	0.09	1	0.64	0.29	0.35	0.03	0.63	0.04
	_sym	**0.85**	0.03	0.7	0.2	0.28	0.11	0.64	1	0.35	0.43	0.03	0.75	0.05
Power	GM	0.38	0.06	0.31	0.15	0.19	0.14	0.29	0.35	1	0.5	0.03	0.37	0.03
Means	SPM	0.46	0.06	0.38	0.16	0.23	0.13	0.35	0.43	0.5	1	0.01	0.43	0.02
FCSG		0.03	0.14	0.02	0.01	−0.06	0.3	0.03	0.03	0.03	0.01	1	0.03	0.01
SSSnet		0.8	0.03	0.66	0.2	0.28	0.12	0.63	0.75	0.37	0.43	0.03	1	0.05
graphB_km		0.05	0	0.05	0.02	0.04	0.01	0.04	0.05	0.03	0.02	0.01	0.05	1

4.1 *Signed graph clustering for clearly defined communities in a dense graph: Highland's Tribe*

The Highland Tribes dataset describes agreeable and antagonistic relations between 16 tribes of the Eastern Central Highlands of New Guinea. Each tribe is a node, with agreeable tribes connected by a positive edge and antagonistic tribes connected by a negative edge. The Highland Tribes dataset [11]

TABLE 8 *Dataset attributes: v is a number of vertices; e is a number of edges in a graph;* % positive *is the number of positive edges divided by e; vertex degree statistics is computed in terms of average, mean, and max node degree; graph density d is calculated as in 1.1, and bal_3 is the percent of triangles in the graph that are balanced; l is a number of communities, pos_in is the percentage of positive edges in the ground truth communities, and neg_out is the percentage of negative edges between ground truth communities*

Labelled	Vertices	Edges		Vertex degrees			Attributes		Communities		
dataset	*v*	*e*	% positive	Average	Median	Max	Density *d*	bal_3	*l*	pos_in	neg_out
highland [11]	16	58	50	7.25	7.5	10	0.483	0.868	3	0.93	1
sampson [12]	18	112	54.4	12.44	12.50	16	0.732	0.6	4	0.48	0.98
cow [13]	50	276	85.51	11.04	9	25	0.225	0.987	3	0.52	0.88
football [53]	248	3174	83.3	25.6	22.5	121	0.104	0.878	20	0.41	1
olympics [53]	464	9345	83.3	40.28	33.00	207	0.087	0.920	28	0.45	1

TABLE 9 *ARI illustrated in Fig. 11 for 12 methods over five datasets considered with emphasis on* **best** *ARI and* second best *ARI for recovering ground truth*

Method/	Laplacians			Balanced Cuts		SPONGE		Power Means		FCSG	SSS	graphB
dataset	_none	_sym	_sep	_none	_sym	_none	_sym	GM	SPM		net	_km
highland	**1**	**1**	0.26	**1**	**1**	**1**	**1**	0.4	0.78	**1**	**1**	**1**
sampson	**0.61**	0.33	0.25	0.51	0.51	0.52	0.6	0.3	0.37	0.41	0.31	0.13
cow	0.06	0.12	-0.02	0.02	0.12	0.06	0	0.06	0.06	-0.01	**0.22**	0.1
football	0.03	**0.76**	0.09	0.68	0.5	0.74	0.71	0.08	0.27	0	0.7	0.01
olympics	0.02	0.72	0.21	0.3	0.11	0.64	**0.85**	0.38	0.46	0.03	0.8	0.07

captures the alliances between sixteen tribes of the Eastern Central Highlands of New Guinea as depicted in Fig. 5 (left). There are three communities in the Highland Tribes data, shown in Fig. 5 (left). The golden node in Fig. 5 (left) has no adjacent negative edges and belongs to two communities.

The graph constructed from Highland Tribes data, **highland**, has a high number of balanced triangles (86.8% from Table 8), is of small size (3 communities, 16 vertices) and exhibits high clusterability. Ground truth labels line up with high % of the positive edges within labelled communities (93%) and negative edges between them (100%), as illustrated in Fig. 5 (left). The signed graph has a density score 0.48, and 87% of triangles in the graph are balanced. Graph's narrow spread of vertex degree (mean is 7; median is 7.5) and high pos_in and neg_out scores for ground communities indicate that the graph coherently represents the communities. Results in Table 2 show that 75% of the methods evaluated achieve a perfect score 1.0 on this dataset. Detailed analysis of ARI for all 12 methods for Highland dataset is presented in Table 2. In Fig. 5(right), we illustrate the vertex correspondence of Laplacian symmetric (lap_sym) method [27] to graphB k-means clustering in status/influence space [40]. These methods retrieve identical results, and community separation is clear in status-influence space. Spectral clustering with symmetric separated Laplacian and both power means methods failed to recover the ground truth. Notably, the ARI

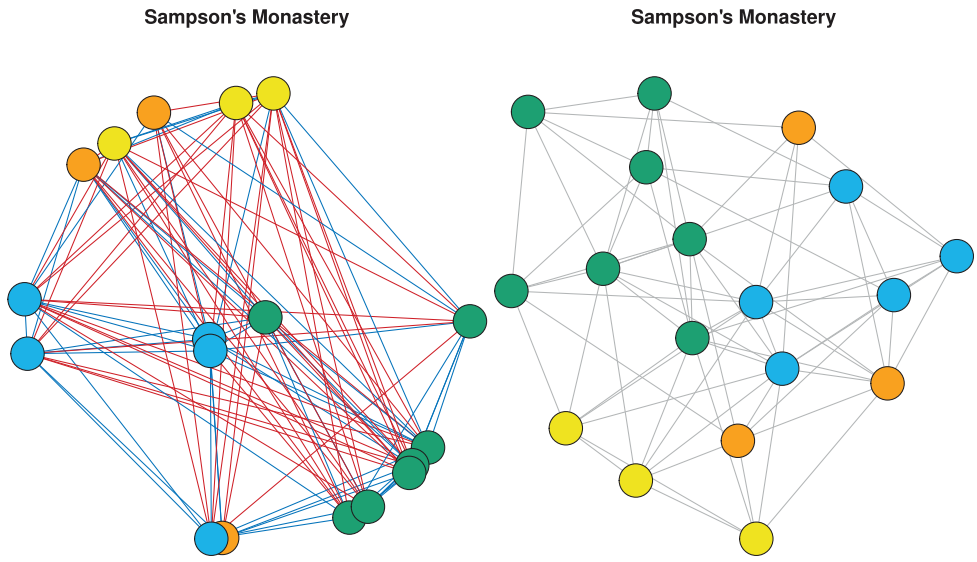

FIG. 6. Network structure for **sampson** dataset where shades depict the community label: 18 vertices, 63 positive edges and 49 negative edges [12] (left); and the ground truth grouping is not distinguishable even when we look at the same network with only the positive edges between nodes (right).

values comparing the symmetric separated Laplacian labels with the geometric means and matrix power means labels are −0.06 and 0.13, respectively see Table 2, indicating very little similarity in the outcomes between these methods. This experiment shows that when small signed graphs reflect ground community labelling, most of the methods perform well.

4.2 *Signed graph clustering for communities not reflected in a signed dense graph: Sampson's Monk survey*

The Sampson's Monastery dataset has 18 nodes that represent eighteen monks as illustrated in Fig. 6(left). Four non-overlapping communities are labelled as the Young Turks, the Loyal Opposition, the Waverers, and the Outcasts [12]. Data were collected during the implementation of the Vatican II, an influential change in the Catholic Church that was controversial among the monks. The Sampson dataset captures this dissent among the monks. The Young Turks began training before the Vatican II and were resistant to change, while the Loyal Opposition were more open to new ideas and began after the Vatican II. The Waverers did not take a strong stance for or against the Vatican II, while the Outcasts were rejected by both the Young Turks and the Loyal Opposition. Note that the Outcasts and the Waverers are not defined by their positive ties to a group or ideology, but by their ambivalence or their rejection from the mainstream opinions.

The sampson dataset is also complex in combining multiple sentiments into a single edge weight. In the original sampson dataset, surveys were administered in which the monks were asked to rank their top three and bottom three peer choices on four qualities. To create the signed graph, we look at sentiments as measured by the surveys in each monk–monk pair. If each quality was ranked positively, we assign a +1 edge. If all qualities were ranked negatively, we assign a −1 edge. In the case of mixed sentiments, we use a weighted average for the scores to determine the edge sign. If the weighted average

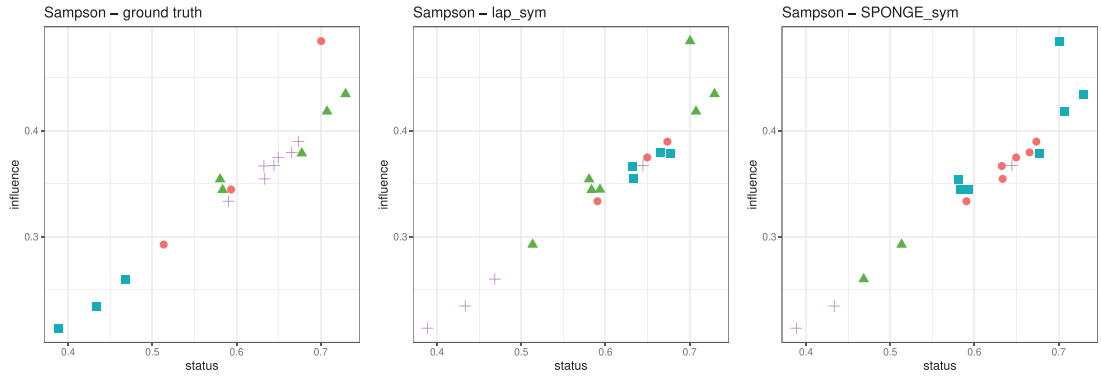

FIG. 7. Vertex correspondence of ground truth (left), Laplacian symmetric (lap_sym) method [27] (centre), and SPONGE (_sym) to graphB clustering in status/influence space [40] for **sampson** dataset. Methods derive very different results for $k = 4$.

is positive, we assign a positive edge; if it is negative, we assign a negative edge; and if the average is 0, we consider the relation ambivalent and assign no edge between the monks. The signed graph resulting from Sampson's monk survey, **sampson**, has 18 vertices, 61 positive and 51 negative edges as illustrated in Fig. 6(right). The density of the graph is 0.732, and it has 60% of triangles that are balanced. The Sampson Monks group dynamic is more complex than Highland Tribes, and the mapping of the ground truth to signed graph communities is not as clean cut, as illustrated in Fig. 6(right). The data have four ground community labels, clear community separation (98% of negative edges are between communities) and *poor* clusterability as 52% of the positive edges among vertices are *not* captured by ground truth community labels.

The performance of the 12 approaches on the Sampson data greatly varies with the highest ARI score of 0.61 for ground truth, and great variation in the mutual ARI scores (Table 3). The greatest agreement is between basic SPONGE and spectral clustering using signed graph Laplacians 0.75. This was expected as they are both spectral methods. Large variations in performance scores show that the constructed signed network *and* ground truth community labelling does not capture the complexity in the relations sufficiently to decipher assigned labels (see Fig. 6(left) for the ground truth). Figure 7 shows the labelling of ground truth and two methods in status-influence space. In Fig. 7(right), vertex 17 is far from its community in status-influence space, and no approach can recover that. Figure 7 also shows that signed spectral clustering using symmetric Laplacian and SPONGE symmetric struggle with coherently forming agreeable groups with minimal sentiment disruption that recover ground truth labels. FCSG underlying assumption that the positive-only subgraph of the network must be a single connected component recovers two out of four communities for sampson and results in low ARI. graphB methodology performs at the level of existing methods but provides additional data resolution and features for analysis. Spectral methods show the best results for **sampson** dataset clustering.

4.3 *Signed graph clustering for communities not reflected in a signed sparse graph: correlates of war*

The Correlates of War dataset contains records of alliances, wars and militarized interstate disputes between 50 countries from 1816 to 2014 [13]. In this study, we chose to focus on the year 1944, assuming that it was clear which side the participants were on during the Second World War. Resulting signed graph from the records for 1944, **cow**, has countries were represented by vertices $v = 50$, and edges

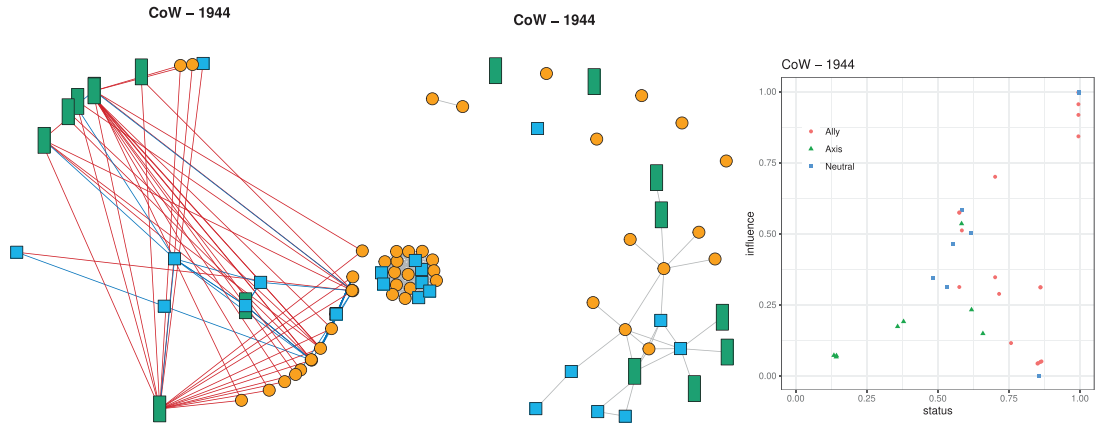

FIG. 8. Correlates of War (**cow**) ground truth labels are illustrated as rectangles (axis), squares (neutral) and circles (allies). Signed graph with positive and negative edges and ground-truth communities labeled (left); the positive-edge-only subgraph of **cow** with allies and neutral forming one tightly connected cluster (centre); and ground-truth labelling in graphB status-influence space (right).

represented relationships between countries $e = 276$. If two countries were allies, we assigned a positive edge between them. If two countries were at war or experienced a militarized interstate dispute, we assigned a negative edge between them. If there was no record of two countries interacting that year, there was no edge in the signed graph. Since we picked 1944, we assigned one of the three labels to a country: Allied Powers, Axis Powers or remaining neutral.

The Correlates of War **cow** signed graph has 50 vertices, 236 positive edges and 40 negative edges as illustrated in Fig. 8. Twenty-five vertices have less than nine edges, while one vertex interacts with half of the vertices (25). The reason we selected the **cow** dataset is that it exhibits similar traits as social network signed graphs at a smaller scale, with skewed vertex density distribution, over 80% positive edges and a low density (0.225). 98.7% of all the triangles in the graph are balanced. Correlates of War provide three ground community labels for the graph; see Fig. 6(right). The ground truth on the signed graph shows solid community separation (88% of negative edges are between communities) and *poor* clusterability as 48% of the positive edges among vertices are not captured by ground truth community labels.

The **cow** clustering products are illustrated in Table 4. ARI scores for ground truth are low for *all* methods. We conclude that ground truth labelling does not capture communities in the signed graph. This makes sense as the WWII countries listed are from multiple continents, and the dynamic between the countries is more complex than their WWII siding. Next, we focus on the method performance and how they compare in recovering data to one another. The Laplacian_none and Sponge_none ARI score is 0.91, the balanced cuts non-symmetric method and FCSG ARI score is 0.92, and the other mutual ARI scores for this dataset are in Table 4.

Next, we visualize the performance of three methods in terms of positive (blue) and negative (red) edges in adjacency matrix in Fig. 9. Adjacency matrices for **cow** are sorted by clustering labels for (a) symmetric Laplacian: API 0.12 pos_in 0.99 neg_out 0.88 (b) balance normalized cut: API 0.12, pos_in 1 neg_out 0.55 and (c) symmetric SPONGE: API 0.06, pos_in 1 neg_out 0.3. We can see that all three methods successfully identified the *Allied Powers* label, and it is consistently the second largest cluster out of three. Their pos_in scores are almost perfect. We conclude that the methods work well in recovering ground truth labels, as the API score is not a great representation of the recovery measure. Where the methods differ is the ability to keep negative edges outside of the clusters. We expand our experiment

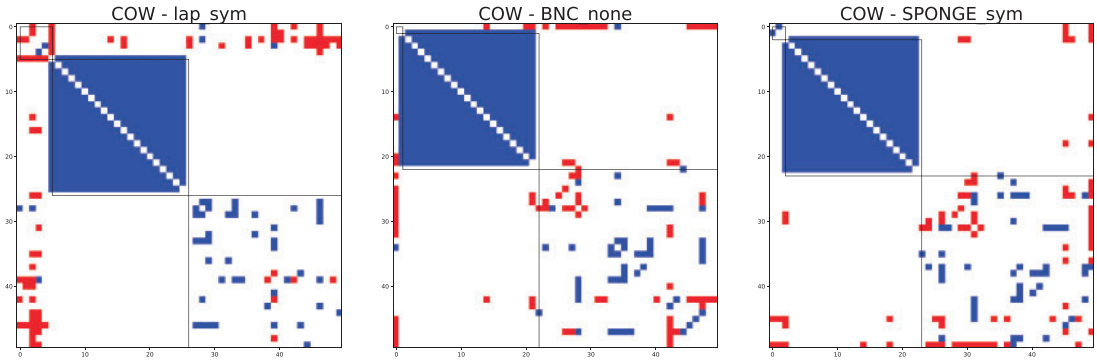

FIG. 9. Adjacency matrices sorted by clustering labels for (left) symmetric Laplacian (centre) balance normalized cut and (right) symmetric SPONGE for **cow** dataset.

analysis to all 12 methods in Table 5. All methods but SSSnet achieve near-perfect results in recovering positive edges within assigned communities better than ground truth. The performance on the negative edges separates methods into two groups: the ones that separate negative edges well for a fixed k and ones that do not. FCSG underlying assumption that *the positive-only subgraph of the network must be a single connected component* recovers only one community for cow (out of three) resulting in low neg_out score.

4.4 *Signed graph clustering for sparse social network data: sports communities*

For this experiment, we have used the network graph datasets constructed in [53]. They have used cosine similarity to produce edges, and we are using the unweighted directed follower graph the authors produced for **football** and **olympics** dataset forms positive only graph basis construction. Since the sports graphs do not have negative edges, we augment them. In **football** data, the graph captures club fan communities, so it is appropriate to use randomized block sampling to insert negative edges among communities. In **olympics** data, the graph captures athletes competing in a sport. Since they were not likely to compete in other sports, we have used randomized block sampling to insert negative edges among labelled groups. We enforce the separation among the communities in the following fashion: first, we randomly select communities to insert a negative edge and then randomly select nodes within those communities for that negative edge to be added between selected nodes. All generated random edges were checked against previously generated negative edges and existing positive edges to avoid duplication. The number of negative edges to add was determined as a percent relative to the number of positive edges. In this case, the number of negative edges was set to be 20 percent of the number of positive edges. Negative edges are only added between communities.

4.4.1 *Football* The Football dataset represents football players and clubs from the English Premier League. The data contains 248 Twitter users grouped into one of 20 clubs in the league. Positive edges in the graph are constructed based on the computed co-occurrence of two players from a larger Twitter community of 7814 followers [53]. Figure 10 (left) illustrates the network with for the network structure. The graph has 2644 positive edges, and we have added 530 negative edges among 20 communities. Vertex degree distribution is converging to power law, as one vertex is connected to almost half of the dataset,

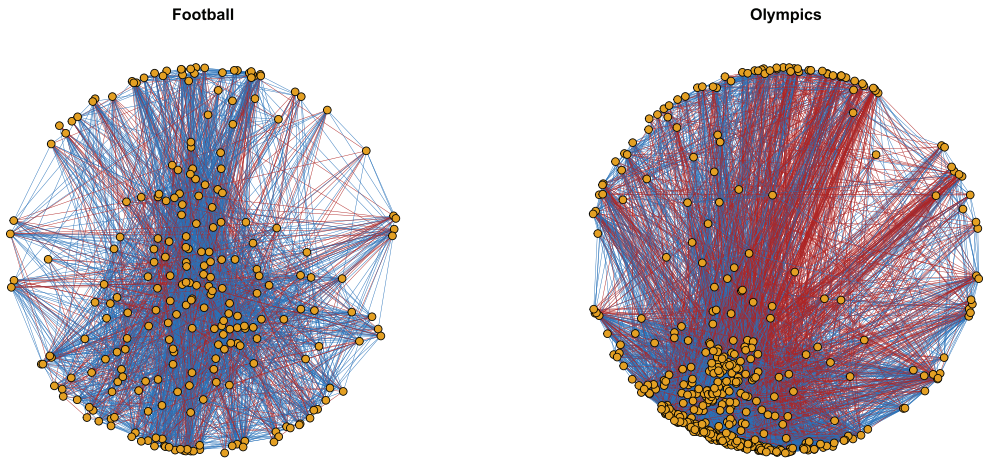

FIG. 10. Illustration of the network structure with blue (lighter shade) positive edges and red (darker shade) negative edges. The original network have no negative edges [53], and the 20% of negative edges was augmented based on the community label separation for **football** (left) and **olympics** (right) datasets [53].

while half of vertices are connected to less than 10% of the vertex set. This is a very sparse graph (0.104) with 87.8% of the triangles balanced and all the characteristics of a real social networks.

The dataset has 20 labels, and since we augmented negative edges only between the communities, we get the perfect neg_out score. Only 41% of positive edges are captured within clusters by ground truth. The high number of positive edges among players in different communities indicates that they all know each other. The labelled community here is more defined by augmented negative edges than by true positive ones. Mutual ARI between methods greatly varies in Table 6. Laplacian_sym, Balance Cuts, SPONGE and SSSnet seem to recover some ground truth, while other methods fail. FGSC yielded isolated vertices at the end of runtime and they were placed in an 'outcast' cluster together to ensure communities sizes of at least two, similar to the sampson dataset. As graphB builds positive clusters, applying fixed k in k-means only distorts its outcome, and hierarchical clustering is better suited. We conclude that experiment needs Twitter edge sentiment data to construct a more truthful signed network graph.

4.4.2 *Olympics* The Olympics dataset features athletes and organizations that were part of the London 2012 Summer Olympics. The data contain 464 Twitter users grouped into 28 different sports based on an analysis of the profile content of 4942 users, whom they follow and their 725662 tweets [53]. Each vertex (athlete) belongs to only one community (sport); see Fig. 10 (right) for the network structure of 7784 positive edges (in blue) and augmented 1561 negative edges in red [53]. Vertex degree distribution converges to power law as half of the vertices are connected with less than 7% of the vertex set, and one super user is connected to 45% of all other athletes. 92% of the triangles are balanced, and this graph is sparse with a density of < 0.09. There are 28 communities, and ground truth captures 45% of the positive labels within the groups. When we augment negative edges only between the communities, we get the perfect neg_out score. The high number of positive edges among athletes in different communities indicates they are likely connected by country or adjacent sport. This dataset is more suitable for overlapping labelling, and the labelled community here is more defined by augmented negative edges than by true positive ones. The method ARI scores are very similar to the football dataset.

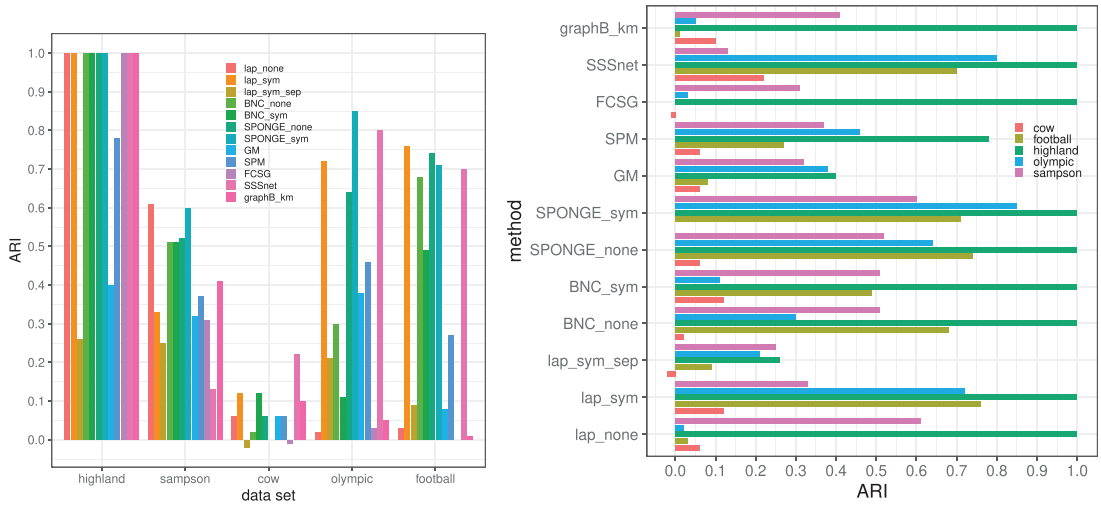

FIG. 11. The ARI [54] based comparison of the accuracy of the methods: by dataset (left) and by method (right). A value near 0 represents a random pairing and 1 represents perfect recovery of ground truth labels.

4.5 *Experiment analysis and conclusion*

Network attributes of the five graphs we have applied 12 signed clustering methods to are outlined side-by-side in Table 8. Figure 11 captures ARI score comparisons per dataset (left) and per method (right) on five datasets. The bolded scores in Table 9 show the top performer per dataset for the ground truth recovery.

We have analysed the clustering success with respect to the size, sparseness, balance, and percent of the positive edges of these datasets, and the results are summarized in Table 9. This experiment does not show clear trends in the effectiveness of a clustering algorithm with respect to the characteristics of the dataset as listed in Table 8; dataset size, ratio of positive edges and density do not influence the top performing algorithm. Clusterable datasets (Table 8 higher pos_in and neg_out scores for ground truth) tend to give better results across most methods (see Table 9), especially for symmetric Laplacian, symmetric SPONGE, and Balanced Cuts. The SPM (matrix power means) [31] approach was an update to the originally proposed geometric means (GM) [30], and SPM has shown in our experiments to do a better job across the board than the GM, as illustrated in Table 9.

The Highland dataset is generally the easiest small dataset to recover ground truth, and all methods handled this well except for Laplacian_sep and Power Means. Sampson serves as a contrasting small dataset with poor ground truth recovery with singed Laplacian and symmetric SPONGE producing the best (but only serviceable) results, and SSSnet falling furthest behind. For Correlates of War, only SSSnet was able to recover some ground truth. The Football and Olympics dataset evaluation shows symmetric Laplacian and symmetric SPONGE as the best performers.

We have made several interesting observations in this section. First, the success of SSSnet seems related to the percentage of balanced triangles, but not the density of edges. This is interesting as the entire balance for the signed graph is reclaimable via triangles in complete graphs. Second, balanced normalized cuts and graphB (which uses a generalization of balanced cuts via the frustration cloud) provide competitive results when they are more suited for hierarchical applications.

TABLE 10 *Five methods used in Section 5 experiments*

Criteria	Method name	Table and figure label
	Symmetric Laplacian [46]	lap_sym
Effectiveness	Balanced normalized cuts [32]	BNC_none
	SPONGE [34]	SPONGE_sym
	Fast clustering for signed graphs [39]	FCSG
Scalability	graph balancing [40]	graphB_km

TABLE 11 *Dataset attributes: v is a number of vertices; e is a number of edges in a graph; % positive is the number of positive edges divided by e; vertex degree statistics is computed in terms of average, mean, and max node degree; graph density d is calculated as in 1.1, and bal_3 is the percent of triangles in the graph that are balanced; l is a number of communities, pos_in is the percentage of positive edges in the ground truth communities, and neg_out is the percentage of negative edges between ground truth communities*

Labelled	Vertices	Edges		Vertex degrees			Attributes	
dataset	v	e	% positive	Average	Median	Max	Density d	bal_3
cow	50	276	85.51	11.04	9	25	0.225	0.987
wiki	7,468	105,160	73.33	28.16	5	1,007	0.004	0.798
slashdot	82,140	500,481	77.03	12.19	2	2,548	<0.001	0.856
epinions	131,828	711,210	83.23	11.82	2	3,558	<0.001	0.890

5. Scaling considerations for signed graph community discovery in real networks

In this section, we select five out of the 12 methods discussed and evaluate their performance on real signed social networks [8]. We also evaluate how these methods fare against large sparse networks. Symmetric Laplacian, Symmetric SPONGE and balanced normalized cuts were selected as they demonstrated the most robust performance in Section 4 experiments. We have selected FCSG [39] and graph Balancing (graphB) [40] methods as they both claim to have the potential to deal with the large sparse graphs that are social networks. Table 10 summarizes five methods evaluated in this section. We wanted to include SSSnet also, but we were not able to successfully run provided code on large graphs due to underlying OpenBlas library error for large dataset. First, we access the effectiveness and the efficiency of the spectral methods in the proposed framework in Section 5.1. Second, we analyse implementation and scalability of FCSG and graphB methods, and discuss avenues for the improvement in Section 5.2.

We are using four signed networks: Correlates of War (cow) [13], Wikipedia Elections (wiki) [8], Slashdot [8] and Epinions [8]. Their attributes are listed in Table 11. The max vertex degree node reflects the existence of influencers in the network. From Table 11, we see that the median vertex degree for Wikipedia Elections is 5, and for Slashdot and Epinions is 2. Median degree in sparse networks is usually much lower than the average degree, unlike Table 8 data.

Wikipedia elections

The Wikipedia Elections dataset was created from data curated by the Stanford Network Analysis Project (SNAP) Wikipedia vote network [8]. It consists of 7066 Wikipedia users running for and voting in

adminship elections prior to 3 January 2008. There were a total of 103663 votes cast in 2794 elections. Users are represented by nodes, and votes are represented by edges, with a positive edge between i and j indicating that either i voted for j or vice versa. Negative edges indicate a negative vote between users. Duplicate and self-referencing edges were removed, and users who were both voters and candidates were represented by separate voter nodes and candidate nodes, with the voter node adjacent only to the edges representing votes they cast and the candidate node adjacent only to the edges representing votes they received. Before clustering, the greatest connected component (GCC) of the graph was taken to eliminate the trivial clustering problem of isolated vertices and/or disconnected components, which will always be placed in their own cluster. Pre-processing resulted in the signed graph w 7468 nodes, 77117 positive and 28043 negative edges. The dataset has low density (0.04) and 79.8% of the triangles in the network are balanced.

Slashdot

Slashdot is a technology website that allows users to tag each other as 'friends' or 'foes'. The dataset was curated by SNAP and contains 82,140 vertices and 549,202 edges [8]. Duplicate and self-referencing edges were removed, and all vertices are connected in this dataset. After pre-processing, the graph has the same number of vertices, 82,140, and now has 385, 515 positive, and 114, 966 negative edges. Half of the nodes have two or less edges, and the graph density is 0.1 %. The graph qualifies as a large and sparse graph with a power-law degree distribution [21].

Epinions

Epinions.com is a website that hosts consumer reviews and employs a 'trust/distrust' metric among users. Users may rate each other as trusted or not, and the resulting 'Web of Trust' is used to weigh product reviews based on the reputations of the authors. The SNAP signed graph has 131828 vertices and 841372 edges [8]. Duplicate and self-referencing edges were removed from the dataset. After pre-processing, the Epinions dataset had 131828 vertices, 592236 positive and 118974 negative edges, and low density (under 0.1%). The graph qualifies as a large and sparse graph with a power-law degree distribution [21].

5.1 *How do best performing methods fare when scale and sparsity of the graphs increase?*

First, we evaluate the effectiveness of three most effective methods from Section 4, namely symmetric Laplacian, symmetric SPONGE and compare it to normalized Balanced Cuts. Community labels are not available so we are using pos_in and neg_out measures to access method's performance. We choose $k = 30$ based on prior work [31]. The performance of the methods is measured by computing the percent of positive edges placed within a cluster and the percent of negative edges placed between clusters for the newly discovered communities. The ratio of positive edges within clusters and negative edges between clusters that each algorithm produces is outlined in Table 12, and their runtime is outlined in Fig. 12.

In a ground-truth clustering of a perfectly modular graph, both metrics would be 1 (100%). Symmetric Laplacian and SPONGE show degradation in performance in Table 12 on the Wikipedia Elections data as the sparsity of the graph increases and the vertex degree distribution begins converging to power law. The Slashdot and Epinions clustering results in Table 12 producing similar results. The low success metrics for negative edges between Epinions and Slashdot, combined with the low community size, suggest a breakdown in effectiveness across the three methods. While they classify almost all positive edges within clusters, close to zero edges remain outside of the clusters. So for large datasets, the community cut is the

TABLE 12 *Ratio of positive edges within clusters and negative edges between clusters for Laplacian symmetric [46]), Balanced Normalized Cuts[32]) and symmetric SPONGE [34] on the CoW [13], Wiki, Slashdot and Epinions [8]. We present results for recommended k [31] and lower k for graphB as fixed k is not applicable for the method with emphasis on* **best** *and second best scores*

		Laplacian		Balanced Cuts		SPONGE		FCSG		graphB		graphB		
Dataset	k	pos_in	neg_out	pos_in	neg_out	pos_in	neg_out	pos_in	neg_out	pos_in	neg_out	k	pos_in	neg_out
cow	3	**0.99**	**0.89**	1.0	0.55	1.0	0.3	0.84	0.25	0.98	0.58	3	0.98	0.58
wiki	30	**0.63**	**0.67**	0.9	0.17	0.59	0.71	0.49	0.52	0.05	0.96	4	0.29	0.73
slashdot	100	1.0	0.0	0.96	0.19	1.0	0.0	N/A	N/A	0.02	0.98	10	**0.22**	**0.78**
Epinions	100	1.0	0.0	**0.96**	**0.19**	1.0	0.0	N/A	N/A	0.03	0.97	10	0.13	0.88

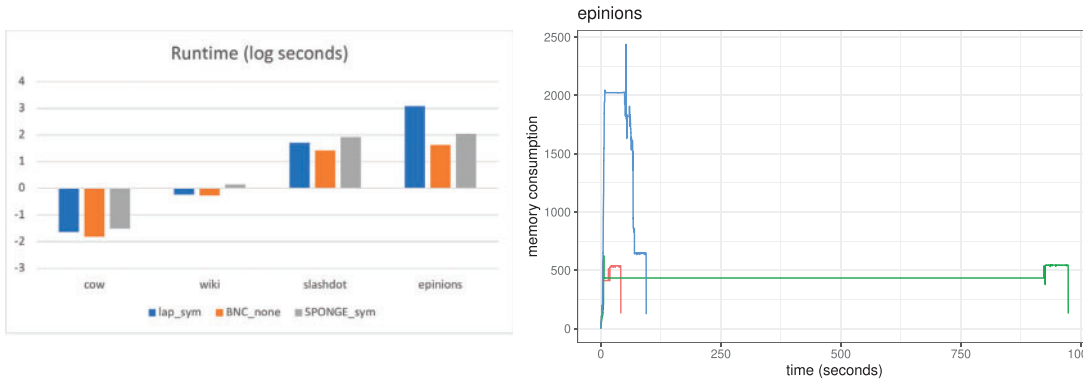

FIG. 12. Log run-times (left) for Laplacian symmetric (lap_sym ([46]), Balanced Normalized Cuts (BNC_none [32]) and SPONGE symmetric (SPONGE_sym [34]) on the CoW (Correlates of War ([13]), Wiki, Slashdot and Epinions ([8]) datasets; Memory consumption for all four methods on Epinion dataset (right).

random negative edge. All the methods show relative fast run times in Fig. 12, with normalized balanced cuts being the most efficient method runtime and memory wise.

Cucuringu *et al.* [34] note the difficulty in recovering community structure in sparse networks with spectral methods, even when normalization is introduced. Dalla'mico [21] observes the following for large sparse network spectral analysis: (1) the eigenvalues of a sparse network tend to spread, which can obscure the largest and smallest eigenvalues and makes the informative eigenvalues difficult to isolate; and (2) high heterogeneity in the degree distribution modifies the *i*th entry of the informational eigenvectors in proportion with the degree of node *i*, known as 'eigenvector pollution' by the authors [21]. Symmetric Laplacian *lap_sym* and symmetric SPONGE *SPONGE_sym* method implementations rely on the eigenvalue approximation. Eigenvalue approximation is notoriously *unstable* for large matrices [36], and the results we have obtained suggest the calculations in these three methods were so prone to error on the large datasets that meaningful community labels were not found. From this experiment, we conclude that the effectiveness of the spectral methods significantly degrades for large sparse networks, likely due to a combined effect of 'eigenvector pollution' and cumulative error in approximate eigenvector computations for large sparse datasets. The Balanced Cuts method *BNC_none*, does not fully break, but its capability to separate clusters by negative edges degrades for networks with a few thousand nodes.

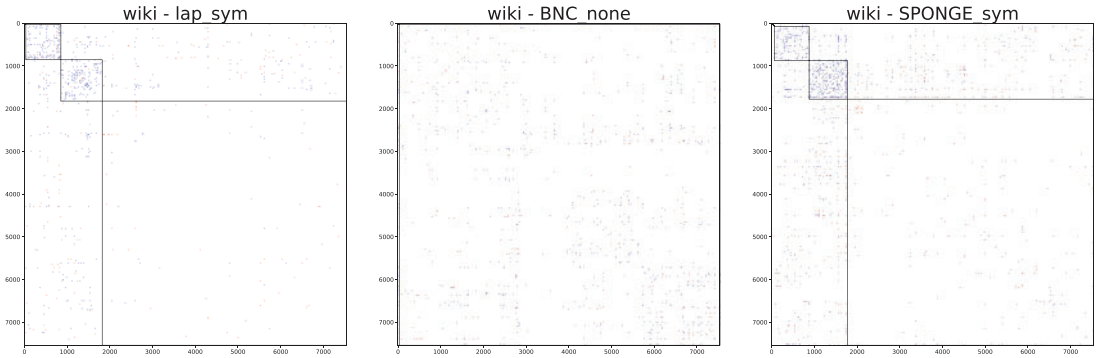

FIG. 13. Blockmodels for Wiki [8] community discovery methods for (left) Laplacian symmetric [46], (centre) Balanced Normalized Cuts [32] and (right) symmetric SPONGE symmetric [34].

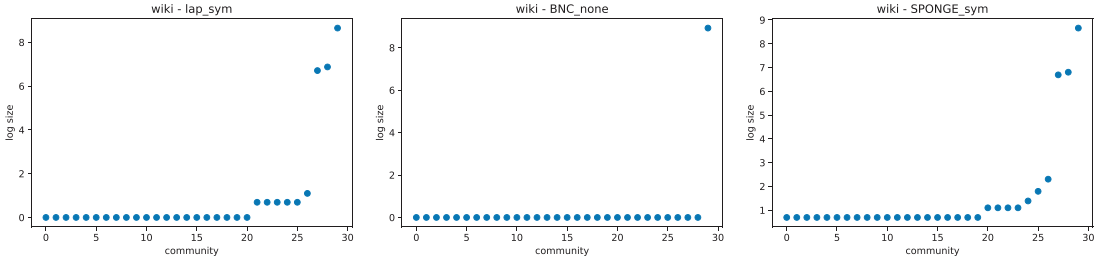

FIG. 14. Community size on log scale in ascending order for Wiki community discovery for (left) Laplacian symmetric [46], (centre) Balanced Normalized Cuts [32] and (right) symmetric SPONGE symmetric [34].

While it preserves the high pos_in score for larger datasets, neg_out score severely degrades for networks with few thousand nodes.

Next, we analyse the degradation of community assignments these three methods produced for Wiki dataset. The Wikipedia Elections dataset clustering using the three methods is hard to visualize using blockmodelling due to the sparseness of the network, as illustrated in Fig. 13. The log scale of discovered community size for all three methods is illustrated in Fig. 14. Figure 14 reveals that all three methods struggled with small cluster sizes for large sparse graphs. Figure 13(centre) contains 29 tiny communities and 1 large community, so the community lines are not visible as for Fig. 13(left) and (right). The introduction of the signed normalized cut, as an alternative to the signed ratio cut, was intended to prevent clustering algorithms from producing trivial or near-trivial optimization solutions with either single nodes or pairs of nodes isolated into a cluster, but we can see that this remedy fails on highly sparse networks such as the Wikipedia Elections dataset. While spectral clustering is a powerful technique for the detection of graph communities, the eigenvalues of signed graphs present a substantial obstruction in the development of a parallel spectral theory that is meaningful for the real social network data [21]. Normalized balanced cuts approach did not scale to sparse networks and trivial communities are prioritized in the clustering. We conclude that none of the approaches that works well on the small dense graphs scales to real large sparse networks.

5.2 *Evaluation of scalable methods for community detection in large sparse graphs*

FCSG

FCSG implementation and its limitations are described in Section 3.2. The authors did not provide an implementation, and the article did not discuss efficiency strategies [39]. Our in-house implementation follows the paper guidelines when possible, as outlined in Algorithms 4 and 5. The approach assumes that the positive-only subgraph of the network must be a single connected component. This places a large constraint on running the algorithm on a real dataset, and reduced FCSG ARI for all the data but highland (ARI = 1.0) in Section 4. FCSG results are outlined in Table 11. For the **cow** dataset, the number of vertices in largest positive connected component (see Algorithm 4) is 21, and 29 vertices are all assigned the same community label. This explains the high pos_in score (0.84) and low neg_out score (0.25) as the algorithm does not consider most of the vertices.

Sparse graphs do not conform to small-world hypothesis, a theory that most users are linked by no more than 5 degrees of separation in a social network. The diameter of largest positive connected component for the **wiki** dataset is 8, for the **slashdot** dataset is 11 and for the **Epinion** is 14 [8]. The authors recommend that L be set to 5 and warn that the algorithm begins to degrade in quality for $L > 5$ and is theoretically unsound for L greater than or equal to 10. The recommended value of L cannot be used for Slashdot and Epinion for step 4 of Algorithm 4 as the parameter used in the random walk gap matrix calculation L must be greater than or equal to the diameter of the all-positive subgraph of the input graph. For the **wiki** dataset, FCSG considers 6530 out of total of 7468 wiki graph vertices for community clustering. The diameter of this positive-subcommunity is 8. Because the diameter value we are using is higher than 5, we see that algorithm begins to degrade for the *wiki* data as pos_in score is 0.49 and neg_out score is 0.52.

FCSG algorithm runs under a minute for small graph processing on a local workstation. The implementation did not scale to the thousands of vertices in the Wikipedia Election dataset [8], and experiments were run for *over 2 weeks* on the Texas State University LEAP system [55]. The Dell PowerEdge C6320 cluster node consists of two (14-core) 2.4 GHz E5-2680v4 processors with 128 GB of memory each, and two 1.5TB memory vertices with four (18-core) 2.4 GHz E7-8867v4 Intel Xeon processors [55]. The implementation ran out of memory when tried on **slashdot** and **Epinion**. The algorithm is irregular, so we used a serial implementation for proof-of-concept as parallelization was non-trivial. We had to implement a custom merge function for this algorithm that did not scale, resulting in N/A entries in Table 11 for slashdot and Epinion data.

graphB

Graph balancing approach uses Balanced Cuts to bipartition the network into agreeable (all positive clusters) [40]; a posteriori application of k-means will necessarily reduce positive edges. Table 11 shows that graphB performance on **cow** is comparable to Balanced Cuts, pos_in is 0.98, neg_out is 0.58. For the wiki dataset, for chosen $k = 100$ graphB is forcefully separating established clusters resulting in pos_in is 0.05, neg_out is 0.96. As implementation of hierarchical clustering is beyond the scope of this paper, we chose $k = 4$ and graphB results improve to pos_in 0.29, neg_out 0.73. We observed the similar trend for slashdot and Epinion dataset for $k = 100$ and $k = 10$. graphB clustering performance improves when more optimal k is chosen, see Table 11 for more details.

Graph Balancing community discovery takes under a minute on the highland, wiki, and sampson datasets on a local workstation. The implementation (underlying NetworkX calls) did not scale to the thousands of vertices in the Wikipedia Election dataset [8]. Since we typically process 1000 trees, one

of the bottlenecks was reading and writing the h5 files [51]. We mitigated this bottleneck using Apache Spark to parallelize the file operations and achieved a speedup of 12.5 over serial processing. Wiki election experiments were completed in 96 minutes on the Texas State University LEAP system [55] using this released timing reproducibility benchmark of the code [51]. Processing the slashdot and Epinions datasets on LEAP took approximately 100 hours on LEAP for these datasets and exposed excessive memory consumption of the implementation. Since the number of fundamental cycles tends to grow with the size of the graph, this quickly became excessive. The authors have proposed leaner more scalable solutions to be used for networks with millions of users and billions of edges [56].

FCSG and graphB scores are complementary on the wiki data, and further studies are needed to determine the right synergy of the methods, and how to overcome small world assumption for FCSG that does not hold for real graphs. *graphB* offers dual metrics for vertices and edges [40], and that is guiding us in the direction of hierarchical clustering for community discovery for large sparse graphs.

6. Conclusion and future work

Scalable, effective and reproducible signed clustering algorithms for community detection on the networks have been a prominent focus in SNA research in the past decade. In this article, we have selected eight real world graph datasets and characterized signed graphs in terms of percent of positive edges, vertex degrees, percentage of balanced triangles, vertex degrees and density. These real-world networks and their summative and comparative statistics provide new benchmarks to build synthetic datasets to examine the strengths and weaknesses of each method. We have compared *twelve* signed graph clustering methods for community discovery. We have evaluated them in terms of effectiveness: ARI scores when ground truth is available, and percentages of positive edges in (pos_in) and negative edges among (neg_out) the discovered clusters when ground truth is not available.

First, we have compared the efficiency of the algorithms on five real social media graphs with ground truth community labels and found the top three performers over a wide range of data with ground-truth labels in Section 4. Second, we showed in Section 5.1 that the state-of-the-art methods of approximate eigenvector calculation do not scale as they retrieve trivial communities. The cumulative errors in the algorithm prevent the meaningful interpretation of output results, as shown in Section 5.2. We used author-provided code when available, but many of the techniques discussed in this paper did not have the original code used to perform the published experiments. This was especially notable for the large real signed networks Epinions and Slashdot, which performed poorly across the board in our experiments but have more promising results published in the literature. The computationally expensive family of techniques that did not rely on local minima or maxima showed potential, but the provided implementation details were too sparse to reproduce the results, as shown in Section 5.2. Another central flaw is that algorithms make assumptions that limit the applicability of some techniques to real-world data, such as the small world assumption in Section 5.2.

We can group the conclusions from the study in three items, as outlined in Table 13. This real-world network study provided us many new avenues to explore. We hope to expand and inform the generation of synthetic signed networks to highlight the differences between each of these methods, implementing a scalable process similar such as [57]. The next step is to *quantify the network attributes that will predict if spectral methods will fail for community detection* We are looking into the improvement of non-spectral, hierarchically robust, methods to synergize with the spectral methods we have studied.

TABLE 13 *This survey helped us identify three issues with state-of-art signed graph clustering and propose mitigation plan*

Scope	Issue	Mitigation approach
Evaluation	No clear preferred method across some datasets	Community-agreed large and sparse dataset benchmarks
Reproducibility	Cannot reproduce the claimed results	Artefacts for code availability and result reproducibility to accompany research products.
Scale	Methods do not scale to large and/or sparse networks	Methods advancement needed, good starting points [39, 43, 45].

REFERENCES

1. KARATAS, A. & SAHIN, S. (2018) Application areas of community detection: a review. In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pp. 65–70.
2. NOGUEIRA DE MOURA, L. (2020) Social network analysis at scale: graph-based analysis of Twitter trends and communities. *Master's Thesis*, Texas State University.
3. ESMAILIAN, P. & JALILI, M. (2015) Community detection in signed networks: the role of negative ties in different scales. *Sci. Rep.*, **5**, 14339.
4. GIRVAN, M. & NEWMAN, M. E. J. (2002) Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, **99**, 7821–7826.
5. ANTAL, T., KRAPIVSKY, P. L. & REDNER, S. (2006) Social balance on networks: the dynamics of friendship and enmity. *Physica D*, **224**, 130–136.
6. LESKOVEC, J., HUTTENLOCHER, D. & KLEINBERG, J. (2010) Predicting positive and negative links in online social networks. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 641–650.
7. SABERI, M., KHOSROWABADI, R., KHATIBI, A., MISIC, B. & JAFARI, G. (2021) Topological impact of negative links on the stability of resting-state brain network. *Sci. Rep.*, **11**, 2176.
8. LESKOVEC, J. & KREVL, A. (2014) SNAP datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.
9. TANG, J., CHANG, Y., AGGARWAL, C. & LIU, H. (2016) A survey of signed network mining in social media. *arXiv:1511.07569 [physics]*. arXiv: 1511.07569. Retrieved from: https://arxiv.org/abs/1511.07569.
10. GALLIER, J. (2016) Spectral theory of unsigned and signed graphs. Applications to graph clustering: a survey. arXiv:1601.04692.
11. READ, K. (1954) Cultures of the Central Highlands, New Guinea. *Southwest. J. Anthropol.*, **10**, 1–43.
12. SAMPSON, S. (1968) A novitiate in a period of change: an experimental and case study of relationships. *Ph.D. Thesis*, Cornell University.
13. SARKEES, M. R. & WAYMAN, F. (2010) Resort to war: 1816 - 2007. https://correlatesofwar.org/data-sets/COW-war/cow-wars.
14. HEIDER, F. (1946) Attitudes and cognitive organization. *J. Psychol.*, **21**, 107–112.
15. HARARY, F. (1953) On the notion of balance of a signed graph. *Michigan Math. J.*, **2**, 143–146.
16. YANG, B., CHEUNG, W. & LIU, J. (2007) Community mining from signed social networks. *IEEE Trans. Knowl. Data Eng.*, **19**, 1333–1348.
17. GÓMEZ, S., JENSEN, P. & ARENAS, A. (2009) Analysis of community structure in networks of correlated data. *Phys. Rev. E*, **80**, 016114.
18. MOORE, M. (1978) An international application of Heider's balance theory. *Eur. J. Soc. Psychol*, **8**, 401–405.
19. MOORE, M. (1979) Structural balance and international relations. *Eur. J. Soc. Psychol.*, **9**, 323–326.

20. Axelrod, R. & Bennett, D. S. (1993) A landscape theory of aggregation. *Br. J. Polit. Sci.*, **23**, 211–233.

21. Dall'Amico, L., Couillet, R. & Tremblay, N. (2021) A unified framework for spectral clustering in sparse graphs. *J Mach. Learn. Res.* **22**, pp. 1–56.

22. Shi, J. & Malik, J. (2000) Normalized cuts and image segmentation. *IEEE Trans. Patt. Anal. Mach. Intell.*, **22**, 18.

23. von Luxburg, U. (2007) A tutorial on spectral clustering. *arXiv:0711.0189 [cs]*. arXiv: 0711.0189 Retrieved from: https://arxiv.org/abs/0711.0189.

24. Arthur, D. & Vassilvitskii, S. (2006) K-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035.

25. Dhillon, I. S., Guan, Y. & Kulis, B. (2004) Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (*KDD '04*). Association for Computing Machinery, New York, NY, USA, pp. 551–556.

26. Karypis, G. & Kumar, V. (1999) A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Scientific Comput.*, **20**, 359–392.

27. Kunegis, J., Schmidt, S., Lommatzsch, A., Lerner, J., De Luca, E. W. & Albayrak, S. (2010) Spectral analysis of signed graphs for clustering, prediction and visualization. *Proceedings of the 2010 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, pp. 559–570. https://epubs.siam.org/doi/abs/10.1137/1.9781611972801.49.

28. Zaslavsky, T. (2010) Matrices in the theory of signed simple graphs. In *Advances in Discrete Mathematics and Its Applications (Mysore, 2008)*. Ramanujan Math. Soc. Lect. Notes Ser., Mysore, pp. 207–229.

29. Zheng, Q. & Skillicorn, D. (2015) Spectral embedding of signed networks. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, pp. 55–63.

30. Mercado, P., Tudisco, F. & Hein, M. (2016) Clustering signed networks with the geometric mean of Laplacians. *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. Luxburg, U. V., I. Guyon & R. Garnett eds). Curran Associates, Inc., pp. 4421–4429. Retrieved from: https://proceedings.neurips.cc/paper/2016/hash/7bc1ec1d9c3426357e69acd5bf320061-Abstract.html.

31. Mercado, P., Tudisco, F. & Hein, M. (2019) Spectral clustering of signed graphs via matrix power means. *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri & R. Salakhutdinov eds), vol. 97. Long Beach, CA, USA: PMLR, pp. 4526–4536.

32. Chiang, K.-Y., Whang, J. J. & Dhillon, I. S. (2012) Scalable clustering of signed networks using balance normalized cut. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. Association for Computing Machinery, New York, NY, USA, pp. 615–624.

33. Chiang, K.-Y., Hsieh, C.-J., Natarajan, N., Dhillon, I. S. & Tewari, A. (2014) Prediction and clustering in signed networks: a local to global perspective. *J Mach. Learn. Res.* **15**, pp. 1177–1213.

34. Cucuringu, M., Davies, P., Glielmo, A. & Tyagi, H. (2019) SPONGE: a generalized eigenproblem for clustering signed networks. *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, in *Proceedings of Machine Learning Research*, Vol. 89, pp. 1088–1098. Available from https://proceedings.mlr.press/v89/cucuringu19a.html.

35. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S. (2001) Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 577–584.

36. Knyazev, A. V. (2001) Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, **23**, 517–541.

37. White, H. C., Boorman, S. A. & Breiger, R. L. (1976) Social structure from multiple networks. I. Blockmodels of roles and positions. *Am. J. Sociol.*, **81**, 730–780.

38. Harel, D. & Koren, Y. (2001) Clustering spatial data using random walks. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*. Association for Computing Machinery, New York, NY, USA, pp. 281–286.

39. Hua, J., Yu, J. & Yang, M.-S. (2020) Fast clustering for signed graphs based on random walk gap. *Soc. Netw.*, **60**, 113–128.

40. Rusnak, L. & Tešić, J. (2021) Characterizing attitudinal network graphs through frustration cloud. *Data Mining Knowl. Discov.*, **6**.

41. Altafini, C. (2013) Consensus problems on networks with antagonistic interactions. *IEEE Trans. Automatic Control*, **58**, 935–946.

42. Hsieh, C.-J., Chiang, K.-Y. & Dhillon, I. S. (2012) Low rank modeling of signed networks. *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY: Association for Computing Machinery, pp. 507–515.

43. He, Y., Reinert, G., Wang S., & Cucuringu, M. (2022) SSSNET: semi-supervised signed network clustering. *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pp. 244–252.

44. Sharma, K., Gillani I.A., Medya, S., Ranu, S. & Bagchi, A. (2021) Balance maximization in signed networks via edge deletions. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (*WSDM '21*). Association for Computing Machinery, New York, NY, USA, pp. 752–760.

45. Tomasso, M., Rusnak L., & Tešić, J. (2022) Cluster boosting and data discovery in social networks. *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing Learning*.

46. Kunegis, J., Lommatzsch, A. & Bauckhage, C. (2009) The Slashdot zoo: mining a social network with negative edges. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. Association for Computing Machinery, New York, NY, USA, pp. 741–750.

47. Cucuringu, M., Davies, P., Glielmo, A. & Tyagi, H. (2019) SigNet. https://github.com/alan-turing-institute/SigNet.

48. Mercado, P., Tudisco, F. & Hein, M. (2016) Clustering Signed Networks with the Geometric Mean of Laplacians. *Proceedings of Advances in Neural Information Processing Systems 29 (NIPS 2016)*, Barcelona, Spain.

49. Mercado, P., Tudisco, F. & Hein, M. (2019) Spectral Clustering of Signed Graphs via Matrix Power Means. *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97, PMLR, pp. 4526–4536.

50. He, Y., Reinert, G., Wang, S., and Cucuringu, M. (2022) *SSSNET: semi-supervised signed network clustering*. https://github.com/SherylHYX/SSSNET_Signed_Clustering.

51. Tešić, J., Mitchell, J., Hull, E., Rusnak, L. (2020) *graphB: Python software package for graph analysis at scale*. https://github.com/DataLab12/graphB.

52. Tomasso M., Rusnak, L., Tešić, J. (2022) *graphC: Python software package for signed graph clustering comparison*. https://github.com/DataLab12/graphC.

53. Greene, D. & Cunningham, P. (2013) Producing a unified graph representation from multiple social network views. In *Proceedings of the 5th Annual ACM Web Science Conference (WebSci '13)*. Association for Computing Machinery, New York, NY, USA, 118–121.

54. Steinley, D. (2004) Properties of the Hubert-Arabie adjusted rand index. *Psychol. Methods*, **9**, 386–396.

55. Division of Information Technology, T. S. U. (2020) LEAP - high performance computing cluster. https://doit.txstate.edu/rc/leap.html.

56. Alabandi, G. and Tešić, J. and Rusnak, L. and Burtscher, M. (2021) Discovering and balancing fundamental cycles in large signed graphs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 68, pp. 1–17.

57. Jung, J. and Park, Ha-Myung and Kang, U. (2020) BalanSiNG: fast and scalable generation of realistic signed networks. In *EDBT*, pp. 193–204.

58. Foundation, P. S. (2020) Python Time Library, Python 3 documentation. https://docs.python.org/3/library/time.html.

59. Guha, R.,Kumar R., Raghavan, P. & Tomkins, A. (2004) Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web (WWW '04)*. Association for Computing Machinery, New York, NY, USA, pp. 403–412.