Advances in Scaling Community Discovery Methods for Large Signed Graph Networks

Maria Tomasso Lucas Rusnak Jelena Tešić

Oct 15 2021

Abstract

Community detection is a common task in social network analysis (SNA) 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 paper, we summarize the development of community detection in signed networks and evaluate current state-of-the-art techniques on several real-world data sets. 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 data sets. 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 large, sparse, real-world signed graphs.

1 Introduction

The rise of social media interactions has illuminated an increasing necessity for a robust understanding of social network analysis, 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 [24]. 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 [37].

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 [12]. A community within a network is defined as a partitioning of nodes such that nodes within the same cluster are strongly connected, while nodes in different clusters are weakly connected; in essence, similar nodes should be grouped together. In real data sets, community structure is almost always present to some degree [15]. 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 [2, 29], or expand to new application domains such as the behavior of the brain [40].

In this paper, 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 [30]. Signed graph definitions are outlined in Section 2, while Section 3 describes unsigned clustering methods adapted for signed graphs, and Section 4 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 [45] 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 [45]. In the same year, a survey of spectral clustering methods for unsigned and signed graphs was published [14]. This survey provides a thorough overview of spectral methods and Laplacian variants for both unsigned and signed graphs. In this paper, 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.

2 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 paper, 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 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.
In graph theory, a signed graph is **balanced** if the product of edge signs around every cycle is positive.

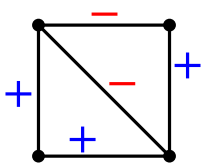

Figure 1: A signed graph Σ has four vertices and five edges. Edges are labeled as + or −.

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

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. 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 [38], Sampson [41], and Correlates of War [42] are three examples of real-world signed and dense graphs.
A **sparse** graph has very few edges relative to the number of nodes. A **planar** graph is a graph that can be embedded in the plane; it can be drawn in such a way that no edges cross each other. Planar graphs are sparse graphs, in that they have only $O(v)$ edges. Note that not all *sparse* graphs are planar graphs even if the inequality holds. Planar graphs are a subset of sparse graphs, and none of the clustering methods for community discovery analyzed in this paper are conditioned by the planarity of the graph. 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. The Wikipedia Election [30], Epinions [30], and Slashdot Zoo [30] data sets are large-scale, sparse signed networks that are not planar.

2.1 Signed Graph Community Discovery in the Past Century

A *community in graph theory* is defined as a cluster of nodes such that nodes within the cluster are densely connected to other nodes within the cluster and sparsely connected to

those outside of the cluster. A *community in signed graph theory* is a community (cluster) of vertices (nodes) that are connected with dominantly positive edges within the cluster and connected to vertices outside of the cluster with dominantly negative edges. 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, e.g., modeling diplomatic relations in the Middle East [35], South Asia [36], and Allied and Axis powers during World War II [4]. Social balance theory, described in Section 4.2, is a branch of signed graph theory proposed by [20] and developed by [18] 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 data sets are rare.

In this paper, 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: (1) in Section 3, we review the signed graph adaptations from algorithms developed for unsigned graphs using discrete optimization techniques; (2) in Section 4, we review novel methods that utilize signed graph characteristics such as balance or the random walk gap.

3 Adaptations of Unsigned Spectral Clustering Methods to Signed Graph Clustering

3.1 Clustering for Unsigned Graphs

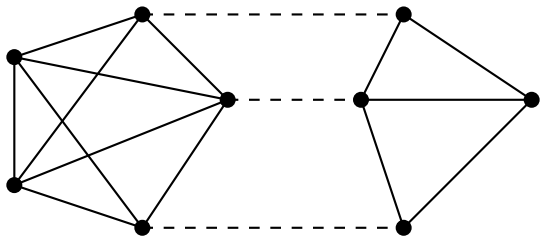

Figure 2: An example of the graph cut set, cuts are dashed edges. For this example, the criteria scores described in Section 3.1 are $cut(X,Y) = 3$, $rcut(X,Y) = 1.35$, and $ncut(X,Y) = .933$.

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 2-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 2-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 2-way Partitioning Many criteria have been proposed for 2-way graph partitioning. The first measure we will introduce for assessing 2-way clustering of a graph is the graph cut. For an unsigned graph G with disjoint clusters X and Y, the **2-way graph cut** is defined as $cut(X,Y) = \sum_{i \epsilon X} \sum_{j \epsilon Y} A_{ij}$. The cut is essentially the number of 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 **2-way ratio cut** is defined as $rcut(X,Y) = cut(X,Y)(\frac{1}{|X|} + \frac{1}{|Y|})$. 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 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** [43]. For an unsigned graph G with disjoint clusters X and Y, the **2-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 2-way partitioning algorithms provide a simple recursive technique to perform k-way partitioning [43]. First, partition the graph into two clusters, then recursively run the 2-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 [43], we know that 2-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 nonlinearity of the criteria. As an alternative, Shi and Malik developed a technique for approximating k-way normalized cut solutions by formulating them as generalized eigenvalue problems [43]. This approach became known as spectral clustering; twenty years later, it is still considered to be a foundational algorithm in graph clustering.

3.1.1 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 [43]. The Laplacian matrix of an unsigned graph G is defined as $\mathcal{L} = \mathcal{D} - \mathcal{A}$. $\mathcal{D}$, the degree matrix, is a diagonal matrix such that the $(i,i)^{th}$ entry represents the degree of vertex v_i. $\mathcal{A}$, 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 $\mathcal{L}$ (or the normalized Laplacian); (2) calculate the first k eigenvectors (the eigenvectors corresponding to the k smallest eigenvalues of $\mathcal{L}$); (3) consider the matrix U formed by the first k eigenvectors; the i^{th} 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 3.1.2 and in Alg. 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 3.1.1 is symmetric and positive semidefinite, meaning the eigenvalues are real and non-negative [46]. 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 [46].

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 data sets 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 [46].

Algorithm 1 Normalized Spectral Clustering [43]

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 $\mathcal{L}_{n\times n}$ and diagonal matrix D of A.
Step 2: Compute the first k eigenvectors $l_1, ..., l_k$ corresponding to the k smallest eigenvalues of $\mathcal{L}_{n\times n}$.
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 i-th 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 Alg. 2.
Output: Cluster labels $l = 1, ..k$ for all n nodes based on u_i vector closeness to final clusters centroids $C_1, ...C_k$.

3.1.2 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 amount of k clusters. The algorithm starts with a set of k center-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 Section 5. The k-means++ algorithm distributes the initial centroids over the given data to minimize the probability of bad outcomes[3] by a very simple randomized seeding technique, as illustrated in Algorithm 2. During each update step t in Alg. 2, all observations x are assigned to their nearest center-point S_i^t. Afterwards, the center-points C_i^{t+1} are repositioned by calculating the mean of the assigned observations to the respective center-points. The update process reoccurs until the center-point update distance $d(C_i^{(t+1)}, C_i^{(t)})$ is smaller than the specified *limit*, as show in Alg. 2. As there is only a finite number of possible assignments for the amount of centroids and observations available, and 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 [3].

Algorithm 2 k-means++ [3]

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 $min_{i \in [1, ...k]}(\left\|x - C_i\right\|^2) = \left\|x - C_j\right\|^2$

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 nonlinear components [10]. 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 $\mathcal{K}$ 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 [25], a fast, multi-level graph

partitioning algorithm that produces equally sized clusters. After we obtain initial clusters, we make kernel matrix $\mathcal{K}$ 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.

Algorithm 3 Batch Weighted Kernel K Means Clustering [10]

Input: The kernel matrix K, the number of clusters k, and the weights for each input w, the inital 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 $c^*(a_i) = argmin_c d(a_i, m_c)$ 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 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.

3.2 Spectral Methods for Signed Graphs

Spectral clustering cannot be immediately applied to signed graphs without prior modifications. The standard Laplacian matrix of a signed graph is indefinite [28] and will not yield real, non-negative eigenvalues. Spectral clustering assumes that all the eigenvalues of the Laplacian are *nonnegative* and *real*. 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 [28] is indefinite, and modifications were made using the signed degree matrix, with the **signed Laplacian matrix** of a graph G as $\bar{\mathcal{L}} = \bar{\mathcal{D}} - \mathcal{A}$, where $\bar{\mathcal{D}}$ is the signed degree matrix given by $\bar{\mathcal{D}}_{ii} = \sum_{j \sim i} |\mathcal{A}_{ij}|$.

Kunegis et al. [28] 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 [28]. 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 [50]

Symmetric normalized Laplacians tend to yield better results than unnormalized Laplacians for graphs with skewed degree distributions [28]. Kunegis et al. propose two ways of normalizing signed Laplacians. First, they define the random walk normalized Laplacian for signed graphs as $\bar{\mathcal{L}}_{rw} = I - \bar{\mathcal{D}}^{-1}\mathcal{A}$ and show that the $\bar{\mathcal{L}}_{rw}$ matrix is positive semidefinite [28]. Second, they define the symmetric normalized Laplacian for signed graphs as $\bar{\mathcal{L}}_{sym} = \bar{\mathcal{D}}^{-1/2}\bar{\mathcal{L}}\bar{\mathcal{D}}^{-1/2} = \mathcal{I} - \bar{\mathcal{D}}^{-1/2}A\bar{\mathcal{D}}^{-1/2}$, where $\mathcal{I}$ is the identity matrix. The signed Laplacian matrix of a graph is positive-definite if and only if the graph is unbalanced [28]. 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. [51] as an extension of the normalized signed Laplacian defined in [28], 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 [28] is used to complete clustering.

Geometric Laplacian means are proposed as a way to address shortcomings in [51] and its inability to recover ground-truth labels in real data sets. 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 [31]. The latest work modifies the Laplacian by combining the positive and negative Laplacians using the **matrix power means** [33]. This approach further improved the results, as we will demonstrate in Experiment 1.

3.3 Non-Spectral Optimization For Signed Graphs

Modified k-way Signed Ratio Cut Criteria Chiang et al. introduce the *balance normalized cut*, a criterion for k-way clustering problems that is analogous to the normalized cut [6]. The balance normalized cut is motivated by the failure of the signed k-way ratio cut, defined as $\sum_{c=1}^{k} \frac{x_c^T \bar{L} x_c}{x_c^T x_c}$, to be minimized by any representation of partitions $\{x_1, \ldots, x_k\}$ which is also proved in [6]. Additionally, the k-way signed ratio cut 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 [6]. Chiang et al. propose a series of new objectives that extended well to k-way partitioning [6]. 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. It is equal to $\max_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T A^+ x_c}{x_c^T x_c}$. The **negative ratio association** minimizes the number of negative edges within each cluster relative to the cluster's size. It is given by $\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T A^- x_c}{x_c^T x_c}$. The **positive ratio cut** minimizes the number of positive edges between clusters. It is given by $\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T L^+ x_c}{x_c^T x_c}$. The **negative ratio cut** maximizes the number of negative edges between clusters. It is given by $\max_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T L^- x_c}{x_c^T x_c}$. 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. It is given by $\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^+ - A) x_c}{x_c^T x_c}$. 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:

$$\max_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^- + A) x_c}{x_c^T x_c} \tag{2}$$

The **balance normalized cut** is very similar to the balance ratio cut, except it normalizes the clusters by volume instead of number of nodes. It is given by $\min_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^+ - A) x_c}{x_c^T \bar{D} x_c}$. Similarly, the **balance normalized association** can be derived from the balance ratio association and is given by $\max_{\{x_1,\ldots,x_k\}\in I} \sum_{c=1}^{k} \frac{x_c^T (D^- + A) x_c}{x_c^T \bar{D} x_c}$. Minimizing the balance normalized cut is equivalent to maximizing the balance normalized association. Thus, the choice between balance normalized cut and association is inconsequential [5]. 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 seconds [6].

SPONGE The SPONGE algorithm is a method for k-way clustering in signed graphs that scales well to large graphs [8]. 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 [47], and the authors provide theoretical guarantees for our approach in the setting of a signed stochastic blockmodel [8]. 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 3. $C_1, \ldots, C_k$ represents a partitioning of G.

$$\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \frac{cut_{G^+}(C_i, \bar{C}_i) + \tau^- vol_{G^-}(C_i)}{cut_{G^-}(C_i, \bar{C}_i) + \tau^+ vol_{G^+}(C_i)} \tag{3}$$

The authors demonstrate that the prior objective function is equivalent to the discrete optimization problem in Eq 4.

$$\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \frac{x_{C_i}^T (L^+ + \tau^- D^-) x_{C_i}}{x_{C_i}^T (L^- + \tau^+ D^+) x_{C_i}} \tag{4}$$

The discrete optimization problem in Eq 4 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. 5.

$$\min_{z_i^T (L^- + D^+) z_j = \delta_{ij}} \sum_{i=1}^{k} \frac{z_i^T (L^+ + \tau^- D^-) z_i}{z_i^T (L^- + \tau^+ D^+) z_i} \tag{5}$$

Objective function in Eq. 5 can be formulated as the generalized eigenproblem whose solution is given by the eigenvectors of $(L^- + \tau^+ D^+)^{-1/2} (L^+ + \tau^- D^-)(L^- + \tau^+ D^+)^{-1/2}$ [8]. The authors use LOBPCC [26] 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 labeling 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 favorably against other leading signed spectral clustering algorithms in experiments performed by the authors, and we evaluate it further in Section 5.

4 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.

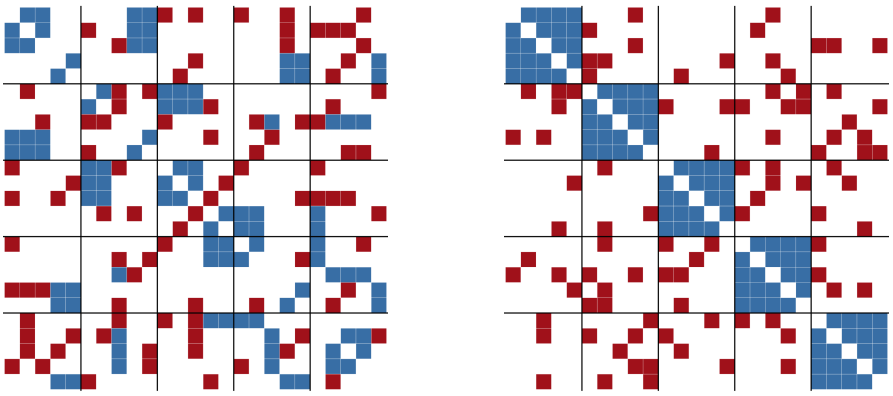

Figure 3: Blockmodel illustration for two graphs with 5 community labels. Positive edges are blue and negative edges are red. Both blockmodel illustrations are sorted by assigned community labels: (left) random community assignment visualized with blockmodel illustration shows no clean community separation; (right) if the distinct communities are present in community assignment, it is visible in blockmodel as large positive connected component (blue blocks).

4.1 Blockmodels for Small Dense Networks

Let $\mathcal{S}$ 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 $c \in \mathcal{S}$ 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 data sets, 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 Figure 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 data sets.

Lean Fit Breiger et al. build on the concept of blockmodels to develop their system for clustering signed data [48]. 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 nonzero 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 [48]. 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 [48]. 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, i.e., 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 no optimize a specific metric; it exhaustively checking all possible blockmodels and checking blockings for a lean fit. This makes the algorithm prohibitively slow when applied to large and sparse signed graphs [48].

4.2 Heider Balance Theory Based Methods

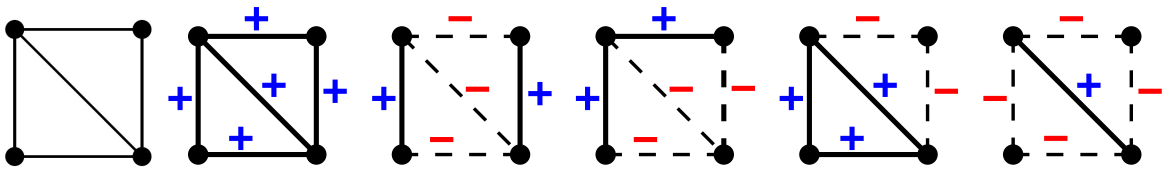

Figure 4: For an underlying unsigned graph with 4 vertices and 5 edges, there are 5 different balanced signed graphs, as illustrated above. Alg. 4 can be applied to these 5 signed graphs. We remove the negative (dashed) edges from the fully balanced graph, and the result is a bi-cut set of clusters with positive (solid) edges only.

Clustering for Balanced Signed Graphs Social balance theory was introduced by Fritz Heider in 1946 [20] and mathematically formalized by Cartwright and Harary in 1956 [18]. 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 Figure 4 for a balanced signed graph with 4 nodes and 5 edges. The clustering approach for the *balanced signed graphs* reduces to a 2-step process as illustrated in Algorithm 4. 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 [39].

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 [1]. If a complete graph is *balanced* or

Algorithm 4 Clustering a fully balanced signed graph

Input: fully balanced signed graph Σ:

Step 1: Apply a Harary cut and separate the signed graphs into 2 sets.

Step 2: Apply unsigned spectral clustering to each of the subsets.

Output: distinct signed graph clusters

Algorithm 5 Clustering a weakly balanced signed graph [1]

Input: Weakly balanced signed graph Σ

Step 1: Partition the nodes into k clusters with only positive edges within clusters and negative edges between clusters.

Step 2: Apply unsigned spectral clustering to each of the subsets.

Output: distinct signed graph clusters

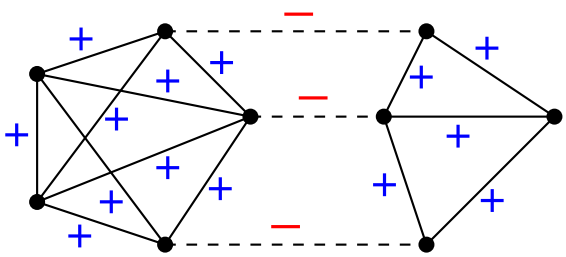

Figure 5: A more complex balanced signed graph (9 vertices) with negative edge set as a Harary-cut.

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 5. Note that weakly balanced partitioning reduces to Alg. 4 for $k = 2$. 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.

Augmentation Based Balancing Hseih et al. impute 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 [21].

Graph clustering in balance feature space was recently explored as an alternative to spectral clustering [39]. 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 vertex, status, and influence in a signed graph based on the balance theory [39]. Authors have demonstrated that status and influence attributes capture the spectrum of signed spectral clustering (Fig. 6) and indicate a possible direction for moving away from eigenvector computation. A study on more degenerate, adversarial networks is necessary to determine if consensus-based attributes [39] can provide insight into networks where spectral clustering fails.

Algorithm 6 Clustering via matrix completion [21]

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 Alg. 2.
Output: Cluster labels for all n nodes.

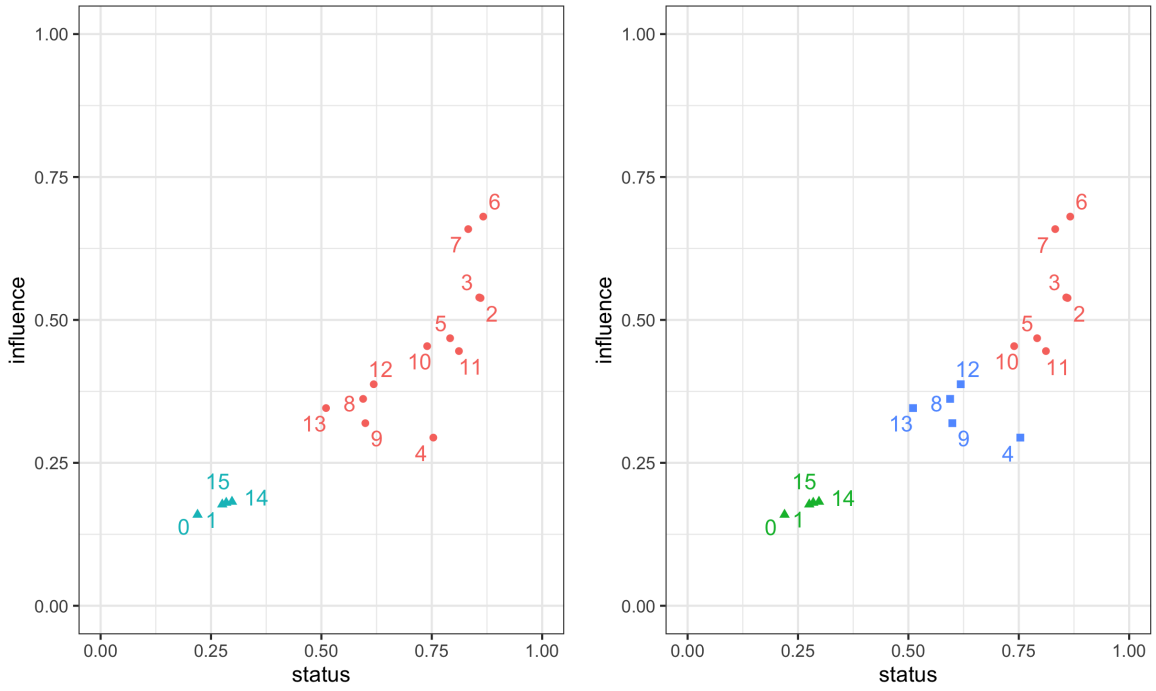

Figure 6: Example of the correspondence of radial distance in status/influence space [39] to signed spectral clustering [28] using symmetric Laplacian results for Highland Tribes data for $k = 2$ and $k = 3$. Each cluster is represented with a different shape/color.

4.3 Modularity-Based Methods

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 [49]. FEC treats the sign and density of edges as clustering attributes. In 2009, Gomez et al. refined modularity metrics and extended existing methods to signed graphs that were directed, weighted, or contained loops [16]. Unfortunately, both FEC and the Gomez method were designed for smaller signed graphs with dense connections. Since many modern clustering applications involve large and sparse networks, we have excluded these methods from the experiments in Section 5.

4.4 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 neighbors of the present node. Harel et al. introduced the random walk clustering algorithm for positively weighted edges in the graph [19]. The method requires only $O(nlogn)$ time, and one of its variants needs only constant space [19]. 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 [22]. 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 8 calls the RWG Algorithm 7 as a subroutine and uses the RWG matrix to reweight 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 [22].

Algorithm 7 RWG algorithm [23]

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

Algorithm 8 FCSG algorithm [23]

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 labeled as a cluster
Output: Cluster labels $C_1, ...C_k$

The first significant limitation of the proposed Algorithm 8 for signed graphs is the underlying assumption that the spanning tree can be constructed using *only positive edges* only 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 17(left) illustrates two all-positive subgraphs for Highland Tribes, and it is clear there is no all-positive spanning tree, and as a result four vertices in a community are left out of the analysis. Figure 17(right) illustrates Sampson's Monastery all-positive subgraph and an all-positive spanning tree, so each vertex receives a label at the end of the procedure. In summary, the first limitation leads to some vertices being unlabeled in the final analysis, as shown in Section 5.3.

The second significant limitation is the assumption of the *small world hypothesis*, a theory that most users are linked by no more than 5 degrees of separation in a social network. The second assumption becomes critical in step 4 of Algorithm 7, where 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. The authors assume that $H_{ij}^{G'} > 0$ for $k \geq 5$ due to the small-world hypothesis, 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 $\mathcal{L}$ must be greater than or equal to the diameter of the all-positive subgraph of the input graph. The authors recommend that $\mathcal{L}$ be set to 5 and warn that the algorithm begins to degrade in quality if $\mathcal{L}$ is greater than or equal to 10.

5 Experimental Comparison

In this section, we compare state-of-the-art signed graph clustering algorithms and implementations in terms of effectiveness when ground-truth is available and efficiency for large real-life data sets; we then discuss some of the algorithm implementations.

5.1 Modified Spectral Clustering Techniques: A Comparison

This experiment evaluates the strengths and weaknesses of nine modified spectral clustering approaches for signed graphs using four real-world graphs with known ground-truth community labels. Each of the nine approaches is evaluated on how well it aligns with true communities. Graphs are described in Section 5.1.1, and their characteristics are listed in Table 1.

	vertices		edges		vertex degrees			measures	
dataset	l	v	e	% positive	average	median	max	density d	balance
highland	3	16	58	50%	7.25	7.5	10	0.483	0.868
sampson	4	18	112	54.4%	12.44	12.50	16	0.732	0.603
football	20	248	3174	83.3 %	25.6	22.5	121	0.104	0.878
olympics	28	464	9345	83.3 %	40.28	33.00	207	0.087	0.920

Table 1: Characteristics of 4 data sets used for comparing modified spectral clustering techniques: l is a number of community labels; v is a number of edges; 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, and balance is the percent of triangles in the graph that are balanced.

5.1.1 Data Sets with Known Community Labels

In this experiment, we use four data sets: (1) Highland Tribes, which models agreeable and antagonistic relationships between tribes in the Eastern Central Highlands of New Guinea [38]; (2) Sampson's Monastery, which models sentiment over time between novice monks in a New England monastery [41]; (3) Football, which tracks Twitter interactions between 248 players belonging to 20 distinct clubs of the English Premier League [17]; and (4) Olympics, which models Twitter interactions between 464 athletes belonging to 28 distinct sports involved in the 2012 London Olympics [17]. Their characteristics are listed in Table 1.

Highland The Highland Tribes data set 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 data set [38] captures the alliances between sixteen tribes of the Eastern Central Highlands of New Guinea as depicted in Fig. 7 (left). There are three communities in the Highland Tribes data, shown in Fig. 7 (left). The golden node in Fig. 7 (left) has no adjacent negative edges and belongs to two communities. The Highland Tribes data set has a high number of balanced triangles (86.8%) and exhibits high clusterability due to the density of the positive edges within clusters and negative edges between clusters in comparison to Sampson network, as illustrated in Fig. 7.

Sampson The Sampson's Monastery data set has eighteen nodes that represent eighteen monks as illustrated in Fig. 7(right). Four non-overlapping communities are labeled as the Young Turks, the Loyal Opposition, the Waverers, and the Outcasts [41]. Data was collected during the implementation of the Vatican II, an influential change in the Catholic Church that was controversial among the monks. The Sampson data set 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.

Negative edges in Sampson's Monastery dataset are crucial to understanding the community structure as two groups; the Outcasts and the Waverers are defined not by their positive

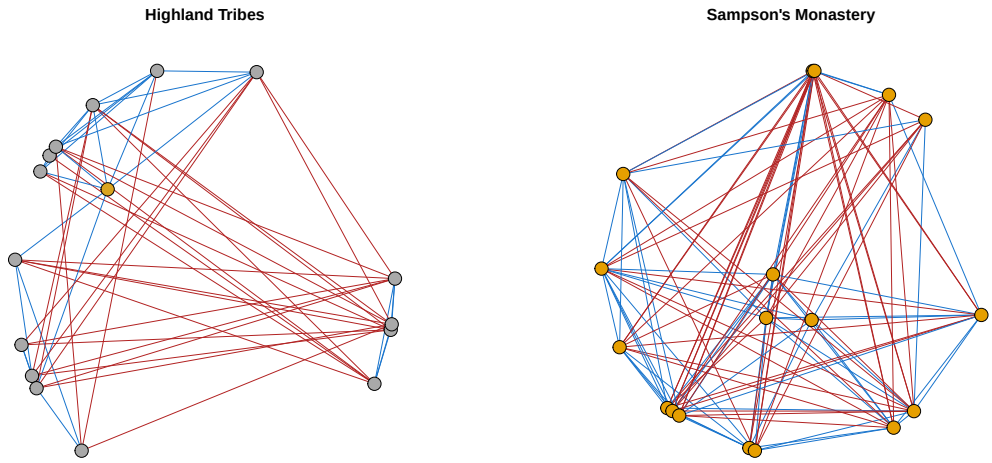

Figure 7: Real sign graph network illustrations, where positive edges are blue and negative edges are red: (left) network structure for Highland Tribes [38]; (right) Sampson's Monastery network [41].

ties to a group or ideology, but by their ambivalence or their rejection from the mainstream opinions. The Sampson data set is also complex in combining multiple sentiments into a single edge weight. In the original Sampson data set, surveys were administered in which the monks were asked to rank their top three and bottom three peer choices on four qualities. We assign weighted scores to each relation based on the survey data and sum across qualities for a total. If the total is greater than 1, the relationship was modeled with a positive edge. If less than 1, the relationship was assigned a negative edge. The complexity of the edges, as well as the importance of ambivalence, make Sampson a difficult graph to cluster. See Figure 7(right).

Football The Football data set represents football players and clubs from the English Premier League. The data contains 248 Twitter users grouped into 20 clubs in the league. Each user belongs to only one club [17]; see Figure 8 (left) for the network structure.

Olympics The Olympics data set features athletes and organisations that were part of the London 2012 Summer Olympics. The data contains 464 Twitter users grouped into 28 different sports. Each user belongs to only one sport [17]; see Figure 8 (right) for the network structure.

Negative edge augmentation Football and Olympics negative edge augmentation was done using randomized block sampling. 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. The approach is biased to nodes in smaller communities, and this was done intentionally due to the nature of the data sets. Competition is between clubs and teams rather than individual players, so it is appropriate to use randomized block sampling as opposed to simple random sampling where all pairs of players from different clubs and sports are equally likely to be chosen. All generated random edges were checked

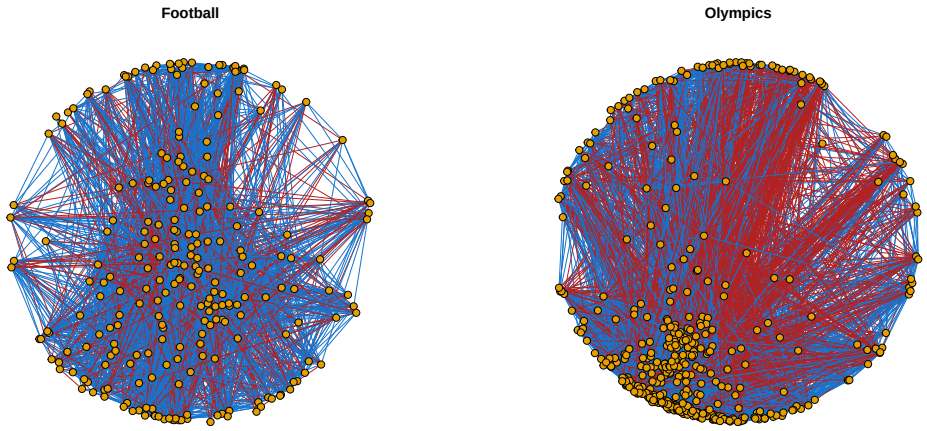

Figure 8: Illustration of the network structure with blue positive edges and red negative edges. The original network have no negative edges [17], and the 20% of negative edges was augmented based on the community label separation for Football (left) and Olympics (right) data sets [17].

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.

5.1.2 Clustering Methodologies

Nine different clustering methods for signed graphs are applied to each data set and listed below with the indexes used to compare the results in Table 2, Figure 9, and Figure 10:

1. *lap_none*: spectral clustering using the signed graph Laplacian [27]
2. *lap_sym*: spectral clustering using the symmetric Laplacian [27]
3. *lap_sym_sep*: spectral clustering using the symmetric separated Laplacian [27]
4. *BNC_none*: balanced normalized cuts [6]
5. *BNC_sym*: symmetric balanced normalized cuts [6]
6. *SPONGE_none*: baseline SPONGE implementation [8]
7. *SPONGE_sym*: symmetric SPONGE [8]
8. *GM*: geometric means [31]
9. *SPM*: matrix power means [33]

The Python package *signet* [7] was used to run the 1.-7. methods, while code provided by the authors of [33] and [31] was used for the final two algorithms [32] [34] For each community, labels were generated for each of the nine clustering methods for k as given in the ground-truth data set. After clustering, the Adjusted Rand Index (ARI) [44] was computed for each labeling and used to compare the accuracy of the methods. The ARI is a popular choice of metric for assessing similarity of partitions and ranges from -1 to 1, with a value near 0 representing a random pairing and 1 representing perfect recovery. For each data set, we compare each method against all others to detect the most (dis)similar methods. The results of this analysis are presented in Table 2, Figure 9, Figure 10 and Appendix 1.

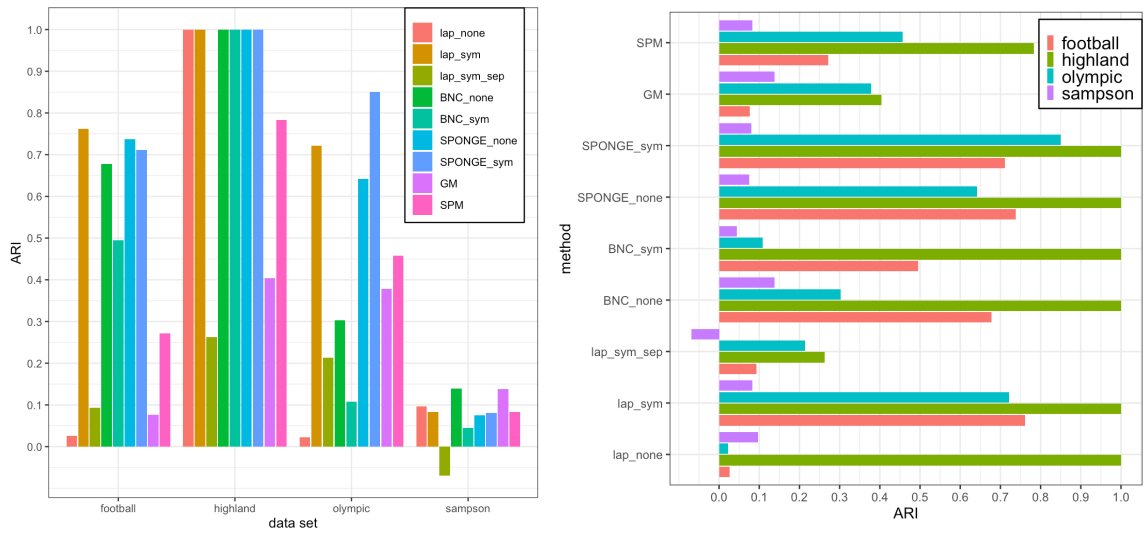

Figure 9: The Adjusted Rand Index (ARI) [44] based comparison of the accuracy of the methods: by data set (left), and by method (right). A value near 0 representing a random pairing and 1 representing perfect recovery.

Six algorithms were able to recover all Highland Tribe's ground-truth community labels, as seen on Table 2 in the second row. The Highland Tribes data set has a high balance score and small size, and this was expected. All nine approaches failed to discover community labels in the Sampson data.

	Laplacians [27]			Balanced Cuts [6]		SPONGE [8]		Power Means[33]	
dataset	_none	_sym	_sym_sep	_none	_sym	_none	_sym	GM	SPM
highland	*1.000*	*1.000*	0.263	*1.000*	*1.000*	*1.000*	*1.000*	0.404	0.783
sampson	0.097	0.138	-0.045	**0.156**	0.093	0.075	0.109	0.138	0.083
olympics	0.016	0.727	0.168	0.357	0.105	0.600	**0.822**	0.379	0.457
football	0.049	**0.781**	0.087	0.603	0.472	0.691	0.703	0.076	0.272

Table 2: Adjusted Rand Index (ARI) illustrated in Figure 9 for nine methods over four data sets considered. Highland tribes is a clusterable data set and 6 out of 9 methods achieve perfect score 1.0. Sampson graph is a complex graph that results in low ARI scores across the boars. Olympics and Football data show great variations in ARI scores per methods, but are consistent over both data sets.

The complex nature of the Sampson data set and the low balance score show that the signed network does not capture the complexity in the labels sufficiently for the clustering approaches to decipher. The bolded scores in Table 2 show the top performer per data set. The symmetric Laplacian and symmetric SPONGE consistently produce top results, as illustrated in Figure 9. We analyze the clustering success with respect to the size, sparseness, balance, and percent of the positive edges of the data sets, and the results are summarized in Figure 10. This experiment does not show clear trends in the effectiveness of a clustering algorithm with respect to the size of the data set within the scope of this experiment, as visualized in Figure 10 (top row) with ARI analysis with respect to the number of vertices (top row, left), number of edges (top row, center), percentage of positive edges (top row,

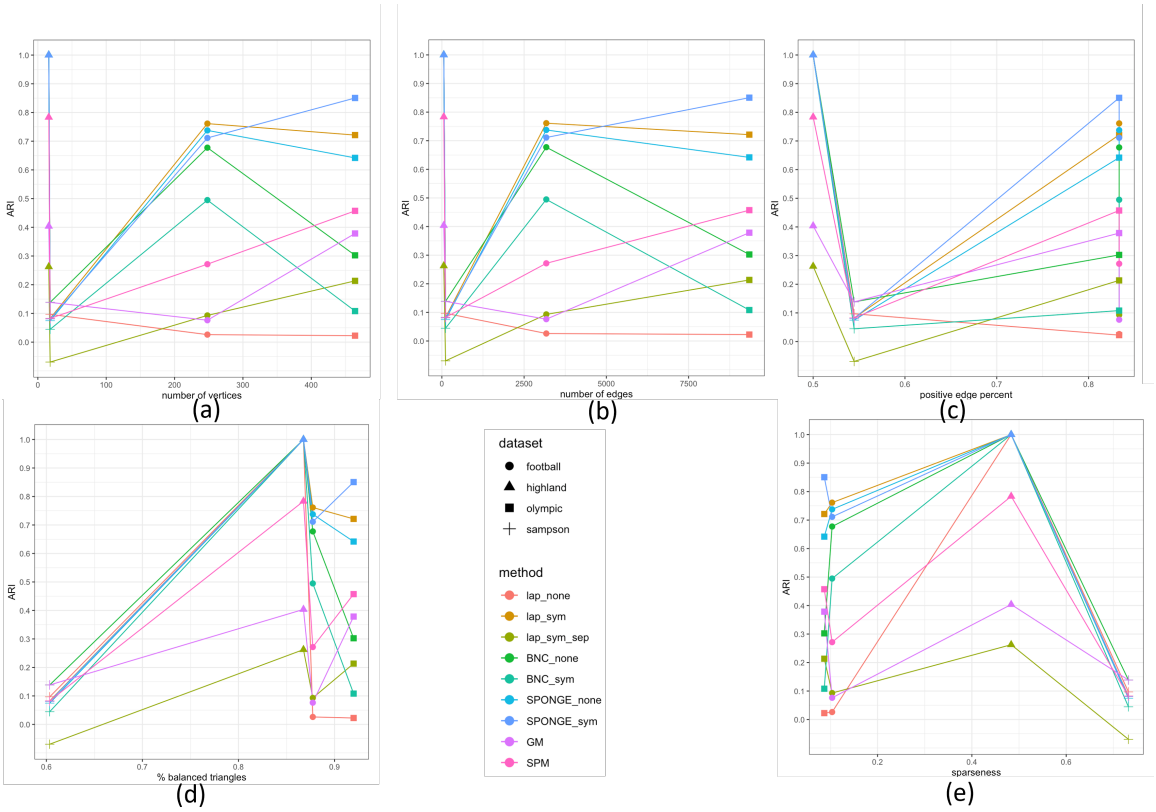

Figure 10: ARI by (a) number of vertices; (b) number of edges; (c) percentage of positive edges. ARI by density per Eq. 1 (d) and percent of balanced triangles (e) for four data sets and nine methods.

right), ARI by number of sparseness (bottom row, left), and percent of balanced triangles (bottom row, right) for four data sets (Football, Highland, Olympics, and Sampson).

The top performing algorithms did not seem to be affected by the percent of positive edges; see Figure 10 (top row). Any effect that was present was likely due to a confounding effect with balance. While a clear pattern does not emerge for sparseness, we can see that the more balanced data sets tend to give better results across most methods, especially for the three that we have previously identified as top performers (symmetric Laplacian, SPONGE, and symmetric SPONGE); see Figure 10 (bottom row). The SPM (matrix power means) [33] approach was an update to the originally proposed geometric means (GM) [31], and SPM has shown in our experiments to do a better job on weakly balanced graphs than the GM.

5.2 Scaling The Signed Graph Clustering Methods: Benchmarking The Performance

In this experiment, we select three leading algorithms from each of the major underlying methodologies and evaluate their scalability on four real signed networks: Wikipedia Elections [30], Slashdot [30], Epinions [30], and Correlates of War [42]. Their characteristics are listed in Table 3. Based on Table 2, we have selected the following three clustering

approaches to evaluate:
1. *lap_sym* - adapted spectral clustering using a symmetric Laplacian matrix [28],
2. *BNC_none* - weighted kernel k-means using the balance normalized cut [6], and
3. *SPONGE_sym* - SPONGE using a symmetric Laplacian [28].

5.2.1 Data sets

	Largest Connected Component			vertex degrees			measures	
dataset	$v = \|V\|$	$e = \|E\|$	% positive	average	median	max	density d	balance
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	119,130	704,267	83.23	11.82	2	3,558	< 0.001	0.890

Table 3: Characteristics of 4 data sets used for comparing scalability: v is a number of edges; 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; balance is the percent of triangles in the graph that are balanced.

Correlates of War The Correlates of War signed graph was built from records of alliances, wars, and militarized interstate disputes between 50 countries from 1816 to 2014 [42]. In this study, we chose to focus on the year 1944. Countries are represented by vertices, $v = 50$, and edges represent relationships between countries, $e = 276$. If two countries were allies, there is a positive edge between them. If two countries were at war or experienced a militarized interstate dispute, there is a negative edge between them. The Correlates of War data set has 3 community labels: Allied Powers, Axis Powers, or remaining neutral [42].

Wikipedia Elections The Wikipedia Elections data set was created from data curated by the Stanford Network Analysis Project (SNAP) Wikipedia vote network [30]. It consists of 7066 Wikipedia users running for and voting in adminship elections prior to January 3, 2008. There were a total of 103,663 votes cast in 2794 elections, and 1235 elections resulted in promotion, while 1559 did not. 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 ensure applicability of the modularity metric on cluster evaluation and to eliminate the trivial clustering problem of isolated vertices and/or disconnected components, which will always be placed in their own cluster. After preparation, there were 7468 nodes and 105,160 edges in the final data set. Based on experiments in prior literature ([31], [33]), k=30 was chosen. The Wikipedia Elections data set is also notable for having outcomes on

the candidate nodes. In this case, we will also compare the proportion of winners, losers, and voters in each cluster [30].

Slashdot Slashdot is a technology website that allows users to tag each other as "friends" or "foes." The data set was curated by SNAP [30] and contains 81,867 nodes and 545,671 edges. Duplicate and self-referencing edges were removed from the data set, and all nodes and edges not in the GCC were dropped to ensure modularity and to eliminate the trivial clusters. After pre-processing, the graph had 82,140 nodes and 500,481 edges [30].

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. Duplicate and self-referencing edges were removed from the data set, and all nodes and edges not in the GCC were dropped to ensure modularity and to eliminate the trivial clusters. After pre-processing, Epinions had $131,828$ users and $841,373$ edges [30].

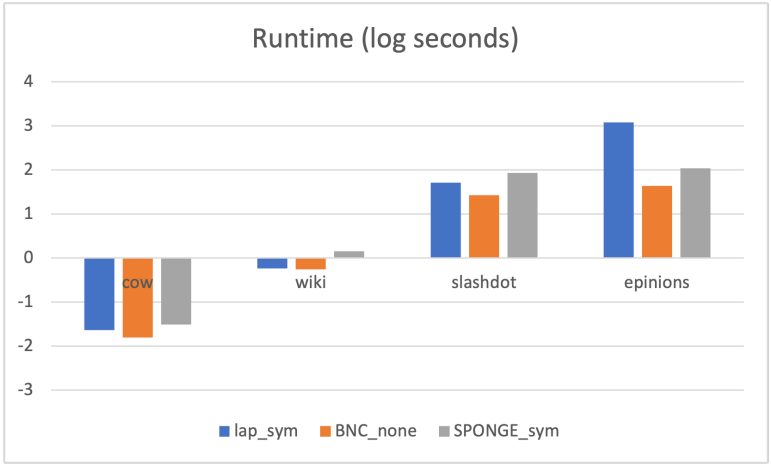

Figure 11: Log runtimes for lap_sym (Laplacian symmetric [27]), BNC_none (Balanced Normalized Cuts[6]), and SPONGE_sym (SPONGE symmetric [8]) on the CoW (Correlates of War [42]), Wiki, Slashdot, and Epinions ([30]).

5.2.2 Runtime and Memory Usage for Three Algorithms

We assess scalability by considering the runtime and memory consumption for each algorithm on each data set. Code was provided for three methodologies by [7] . We made modifications to the original code to measure runtime and memory consumption. All algorithms were executed using a 2.2GHz Quad-Core Intel Core i7 with 16 GB of memory. The runtime of each algorithm was measured using the 'time' library in Python [13].

We find that the runtimes are relatively comparable between $SPONGE\_sym$, BNC_none, and lap_sym approaches, as illustrated in Figure 11. The balance normalized cut was the fastest across all four data sets, while symmetric SPONGE was the slowest (although not outstandingly) on Wikipedia Elections and Slashdot. The Epinions data set clustering via the symmetric Laplacian was significantly slower than the other methods. Next, we measured memory consumption during runtime for each method on the four data sets. Memory

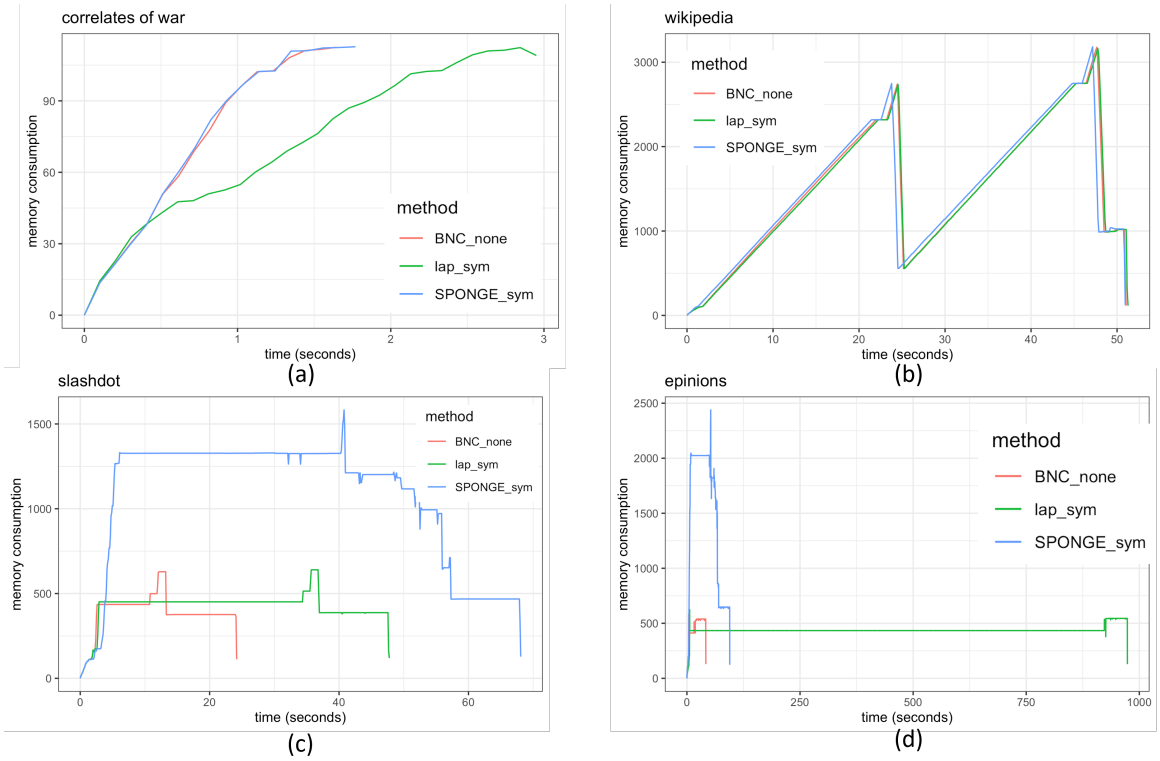

Figure 12: Memory use over time for lap_sym (Laplacian symmetric [27]), BNC_none (Balanced Normalized Cuts[6]), and SPONGE_sym (SPONGE symmetric [8]) on the (a) CoW [42], (b) Wiki [30], (c) Slashdot [30], and (d) Epinions [30].

consumption over time was measured using the Memory Profiler library in Python. Figure 11 shows that all three methods (*lap_sym, BNC_none, SPONGE_sym*) had roughly the same peak memory usage on the two smaller data sets, Correlates of War[42] and Wikipedia Elections [30]. The discrepancies in peak usage are evident for the larger networks, Slashdot and Epinions, in Figure 12; the *SPONGE_sym* approach has the highest memory consumption by a large margin. Additionally, we observed that although the symmetric Laplacian clustering was the slowest to finish on Epinions, it also maintained low and consistent memory consumption throughout its runtime.

The area under the curve for the figures in 12 represents total memory usage by the algorithm. We plotted the total memory usage on the linear Figure 13 (left) and logarithmic Figure 13 (right) scales to compare overall resource usage for each algorithm. The total resource usage is very similar on the smaller Correlates of War and Wikipedia Elections data sets. The symmetric Laplacian *lap_sym* used the most resources to cluster Epinions, the largest network. The balance normalized cut (*BNC_none*) remained the least resource-intensive algorithm of the three methods compared.

5.2.3 Measured Effectiveness

Four data sets do not have known ground-truth communities, so we assess success by computing the percent of positive edges placed within a cluster and the percent of negative

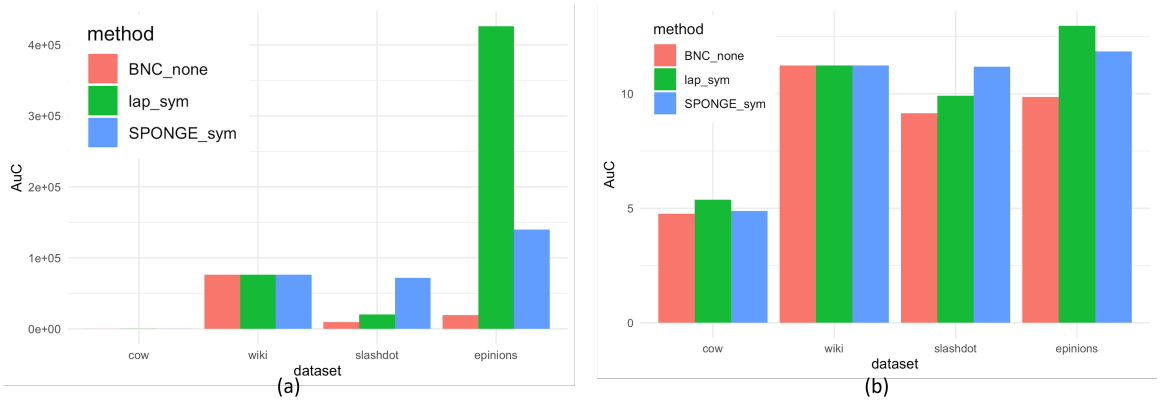

Figure 13: (a) resource and (b) log resource use for lap_sym (Laplacian symmetric [27]), BNC_none (Balanced Normalized Cuts[6]), and SPONGE_sym (SPONGE symmetric [8]) on the CoW [42], Wiki, Slashdot, and Epinions [30].

	lap_sym [27]		BNC_none [6]		SPONGE_sym [8]	
dataset	pos_in	neg_out	pos_in	neg_out	pos_in	neg_out
cow	0.987	0.875	0.996	0.550	1.000	0.300
wiki	0.625	0.667	0.901	0.165	0.591	0.713
slashdot	0.999	0.000	0.963	0.189	1.000	0.000
Epinions	1.000	0.000	0.964	0.190	1.000	0.000

Table 4: Ratio of positive edges within clusters and negative edges between clusters for lap_sym (Laplacian symmetric [27]), BNC_none (Balanced Normalized Cuts[6]), and SPONGE_sym (SPONGE symmetric [8]) on the CoW [42], Wiki, Slashdot, and Epinions [30].

edges placed between clusters. The ratio of positive edges within clusters and negative edges between clusters that each algorithm produces is outlined in Table 4. In a ground-truth clustering of a perfectly modular graph, both metrics would be 1 (100%).

Correlates of War The Correlates of War clustering products of all three algorithms are shown as the adjacency matrices sorted by cluster label in ascending order by cluster size in Figure 14. All three methods successfully identified the *Allied Powers* label, and it is consistently the second largest cluster out of three. The differences in edge classification are shown in Table 4. The symmetric Laplacian took the most balanced approach with 98.7% of the positive edges being placed within clusters and 78.5% of the negative edges being placed between clusters. The balance normalized cut (BNC_none) and symmetric SPONGE (*SPONGE_sym*) both optimized positive edge placement more heavily than negative edge placement, with symmetric SPONGE placing 100% of the positive edges within a cluster but only 30% of the negative edges between clusters.

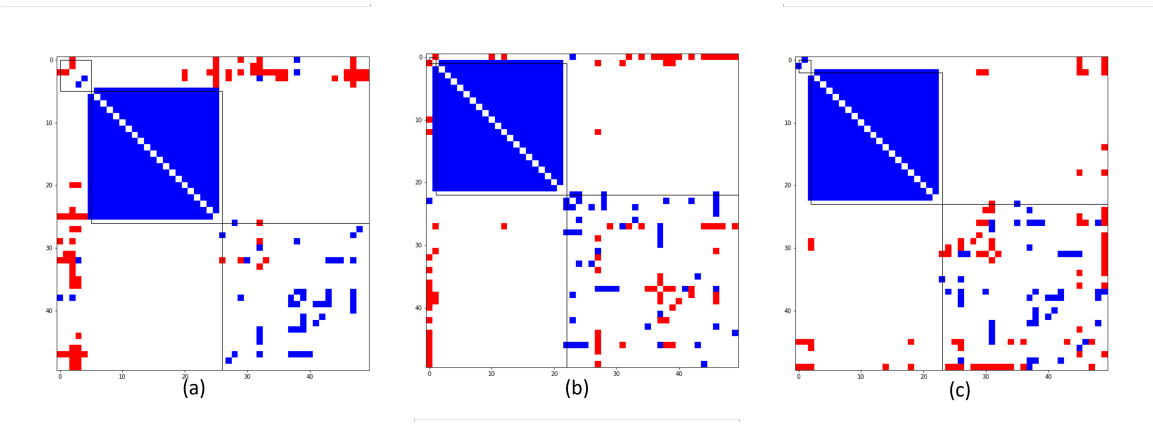

Figure 14: Adjacency matrices for Correlates of War - 1944 [42] sorted by clustering labels for (a) symmetric Laplacian
(b) balance normalized cut, and (c) symmetric SPONGE.

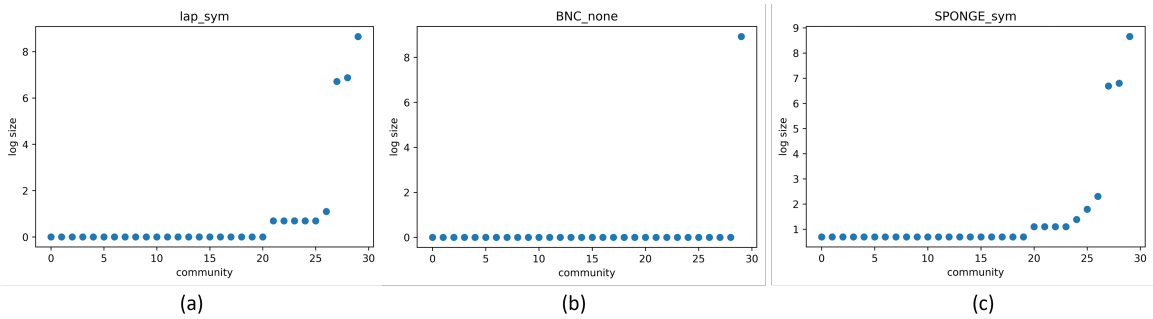

Figure 15: Community size on log scale in ascending order for Wiki data and community discovery methods:
(a) *lap_sym*, Laplacian symmetric [27]; (b) *BNC_none* for Balanced Normalized Cuts[6]; and
(c) *SPONGE_sym* for SPONGE symmetric[8].

Wikipedia Elections The Wikipedia Elections data set clustering using the three methods is hard to visualize using blockmodeling due to the sparseness of the network, as illustrated in Figure 16. The log scale of discovered community size for all three methods is illustrated in Figure 15. All three methods struggled with small cluster sizes in Figure 15, even though we are using a small cluster size in the normalization. Figure 16(center) contains 29 tiny communities and 1 large community so the community lines are not visible as for Figure 16(left) and Figure 16(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 fails on highly sparse networks such as the Wikipedia Elections data set. Cucuringu et al. [8] note the difficulty in recovering community structure in sparse networks with spectral methods, even when normalization is introduced. Dall'Amico et al. further expand on the causes of failure for spectral methods on real sparse graphs, which include an average degree that is small relative to and not proportional with the size of the graph and a power-law distribution of vertex degrees [9].

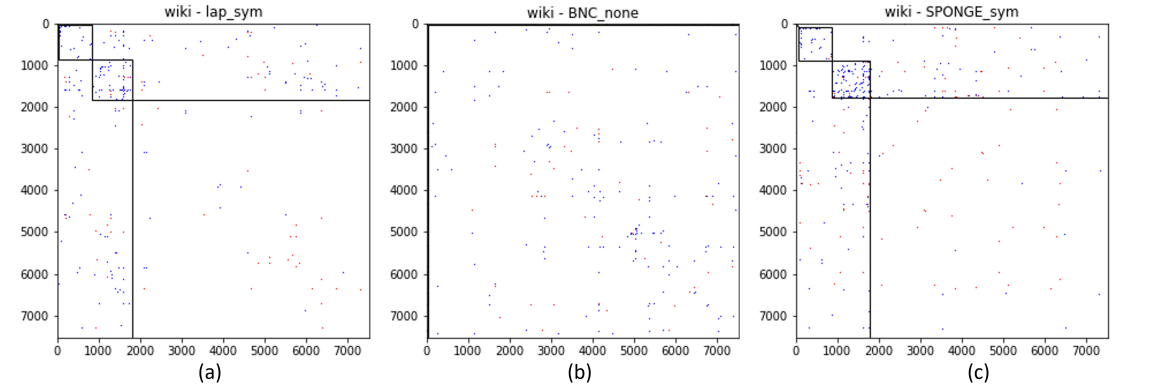

Figure 16: Blockmodels for Wiki [30] community discovery methods: (a) lap_sym Laplacian symmetric [27], (b) BNC_none Balanced Normalized Cuts[6], and (c) SPONGE_sym SPONGE symmetric right. [8]).

Slashdot and Epinions The Slashdot and Epinions clustering results in Table 4 produce 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. The *lap_sym*, *BNC_none*, and *SPONGE_sym* methods rely on the eigenvalue approximation. Eigenvalue approximation is notoriously *unstable* for large matrices [26], and the results we have obtained suggest the calculations in these three methods were so prone to error on the large data sets that meaningful community labels were not found.

5.3 Fast Clustering for Signed Graph (FCSG) using random walk gap: Implementation Considerations

FCSG	highland	sampson	cow	wiki
Number of vertices in largest connected component	16	18	50	7,468
Number of vertices in RWG positive subgraphs	12	18	21	6530
Diameter of RWG positive subgraphs	4	3	1	8

Table 5: Network characteristics of Highland Tribes, Sampson's Monastery, Correlates of War - 1944, and Wikipedia Election. Sampson's Monastery network is the only graph that had an all-positive spanning tree and thus, did not lose any vertices during RWG calculations.

The paper focused on the mathematical characteristics of the algorithm rather than the technical details of implementation [22]. We have implemented the FCSG algorithm in Python 3.8.5 with the NetworkX package based on the information available in [22]. The paper did not discuss efficiency strategies, and our implementation follows the paper guidelines when possible. To run the Random Walk Gap (RWG) algorithm outlined in Alg. 7, the *positive-only subgraph of the network must be a single connected component*, as explained in Section 5.3. This places a large constraint on running the algorithm on a real data set. If it is not a single connected component, clustering will only be performed on the greatest

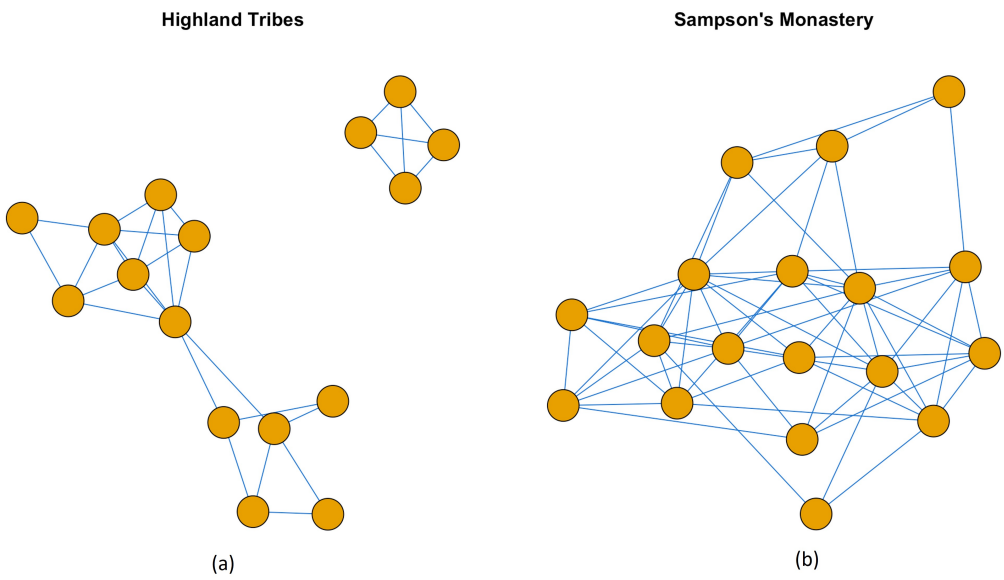

Figure 17: Positive network structure for (a) Highland Tribes and (b) Sampson's Monastery.

communities	highland	sampson	cow	wiki
Ground Truth	3	4	N/A	N/A
FCSG Result	2	3	1	145
ARI	1.00	0.09	N/A	N/A
pos_in	0.931	0.622	0.839	0.487
neg_out	1.000	0.882	0.250	0.518

Table 6: Number of ground-truth communities, number of detected communities via FCSG, and Adjusted Rand Index (ARI) for Highland Tribes and Sampson. For Highland Tribes, all dropped nodes were assigned to their own cluster. Number of detected communities via FCSG, percent of positive edges within clusters, and percent of negative edges between clusters for CoW and Wiki.

connected component, and all nodes outside of the greatest connected component will not be placed into a cluster. Table 5 shows how many vertices are left to cluster after this condition is met. The parameter used in the random walk gap matrix calculation $\mathcal{L}$ must be greater than or equal to the diameter of the positive edge-only subgraph G of Σ. The authors warn that the algorithm begins to degrade in quality for $\mathcal{L} > 5$ and is theoretically unsound for $\mathcal{L}$ greater than or equal to 10. For large and sparse real-life data sets, the diameter exceeds 5, as shown in Table 5 for the Wikipedia data. Additionally, Slashdot and Epinions have diameters of 11 and 14 respectively [30]. The recommended value of $\mathcal{L}$ cannot be used for large, sparse graphs as their diameters exceed the recommended value. The FCSG results in Table 6 show that the method discovers a subset of the communities in Highland Tribes and Sampson's Monastery, and the ARI score for each data set is 1.00 and 0.09 respectively. We are interpreting results in this section with caution due to the reproducibility issue, as discussed in Section 4.4.

The algorithm is relatively fast for the small data set, but our implementation did not scale to the thousands of vertices in the Wikipedia Election data set [30]. For this data, the

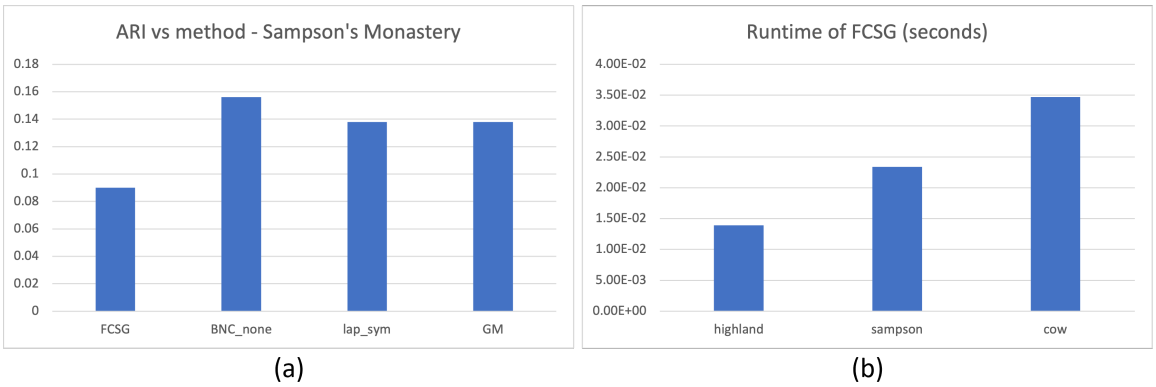

Figure 18: (a) FCSG effectiveness is low: FCSG clustering ARI for Sampson's Monastery compared with the top 3 methods from Table 2; (b) FCSG runtime on small data sets is fast: runtime in seconds on Highland Tribes, Sampson's Monastery, and Correlates of War data sets.

experiments are run on the Texas State University LEAP system [11]. 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 [11]. Figure 18 shows runtimes on a workstation with a 2.2GHz Quad-Core Intel Core i7 and 16 GB of memory for small graphs and Wikipedia Elections run on LEAP. The Wikipedia Elections data set ran for **over 2 weeks** on our most powerful cluster. Unfortunately, this algorithm could not easily be sped up. The algorithm is irregular, making it difficult to parallelize, so we used a serial implementation. Additionally, there was no NetworkX merge function that met the needs of this algorithm; we had to write a custom node merging function that did not scale. Since Epinions and Slashdot are substantially larger than Wikipedia Elections, we did not run FCSG on these data sets due to scalability issues. The RWG clustering algorithm assumptions oppose the real-world graph characteristics, positive edge spanning tree, and small world assumptions, as discussed in Section 4.4. Note that the authors did not publish the code, so the reproducibility of the reported results differs from the original paper. Additionally, we had to raise the parameter L above the recommended value of 5 to 8 (Table 5) to prevent division by zero on the Wikipedia Elections data set, and we had to pre-process the graphs to drop any nodes that were not part of the all-positive greatest connected component. We use the ARI measure to demonstrate the inferior performance of the FCSG algorithm on real graphs in comparison to previously published methods.

6 Conclusion and Future Work

Scalable, effective, and reproducible signed clustering algorithms for community detection on large and sparse networks have been a prominent focus in social network analysis research in the past decade. In this paper, we have compared ten signed graph clustering methods for community discovery. We have evaluated them in terms of effectiveness: ARI score, k-cut score, percent of positive edges within clusters, and percent of negative edges between clusters. We have compared the efficiency of the algorithms on eight real social media graphs and found the top three performers over a wide range of data with ground-truth labels in Section 5.1. The performance of the symmetric Laplacian balance normalized cut

and symmetric SPONGE methods did not scale well with the large size of the data, as all three algorithms rely on eigenvalue approximation. State-of-the-art methods of approximate eigenvector calculation do not scale, and cumulative errors in the algorithm prevented the meaningful interpretation of output results, as shown in Section 5.2. Here, we have found the first central flaws in existing clustering techniques for signed graphs: an inability to scale to large graphs, especially for sparse networks. This study also highlights the importance of ongoing reproducibility discussions within the computer science and computation community. 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 signed data sets 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 does 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 5.3. 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 5.3

Our next step is to compare the state-of-the-art methods with a frustration-cloud based approach [39] and evaluate how consensus-based feature extraction can mitigate the limitations of current approaches to improve efficiency and effectiveness. We are also looking into signed graph clustering where a vertex can belong to multiple communities and/or have a continuous score for community belonging.

References

[1] Altafini, C. (2013) Consensus Problems on Networks With Antagonistic Interactions. *IEEE Transactions on Automatic Control*, **58**(4), 935–946.

[2] Antal, T., Krapivsky, P. L. & Redner, S. (2006) Social Balance on Networks: The Dynamics of Friendship and Enmity. *Physica D: Nonlinear Phenomena*, **224**(1), 130–136.

[3] Arthur, D. & Vassilvitskii, S. (2007) K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

[4] Axelrod, R. & Bennett, D. S. (1993) A Landscape Theory of Aggregation. *British Journal of Political Science*, **23**(2), 211–233.

[5] Chiang, K.-Y., Hsieh, C.-J., Natarajan, N., Tewari, A. & Dhillon, I. S. (2013) Prediction and Clustering in Signed Networks: A Local to Global Perspective. *arXiv:1302.5145 [cs]*. arXiv: 1302.5145.

[6] 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, page 615, Maui, Hawaii, USA. ACM Press.

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

[8] Cucuringu, M., Davies, P., Glielmo, A. & Tyagi, H. (2019b) SPONGE: A generalized eigenproblem for clustering signed networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1088–1098. PMLR.

[9] Dall'Amico, L., Couillet, R. & Tremblay, N. (2020) A unified framework for spectral clustering in sparse graphs. .

[10] Dhillon, I. S., Guan, Y. & Kulis, B. (2004) Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, page 551. ACM Press.

[11] Division of Information Technology, T. S. U. (2020) LEAP - High Performance Computing Cluster. `https://doit.txstate.edu/rc/leap.html`.

[12] Esmailian, P. & Jalili, M. (2015) Community Detection in Signed Networks: the Role of Negative ties in Different Scales. *Scientific Reports*, **5**(1), 14339.

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

[14] Gallier, J. (2016) Spectral Theory of Unsigned and Signed Graphs. Applications to Graph Clustering: a Survey. .

[15] Girvan, M. & Newman, M. E. J. (2002) Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, **99**(12), 7821–7826.

[16] Gómez, S., Jensen, P. & Arenas, A. (2009) Analysis of community structure in networks of correlated data. *Phys. Rev. E*, **80**, 016114.

[17] 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*, pages 118–121.

[18] Harary, F. (1953) On the notion of balance of a signed graph. *Michigan Math. J.*, **2(2)**, 143–146.

[19] 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*, pages 281–286.

[20] Heider, F. (1946) Attitudes and cognitive organization. *J. Psychology*, **21**, 107–112.

[21] Hsieh, C.-J., Chiang, K.-Y. & Dhillon, I. S. (2012) Low Rank Modeling of Signed Networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 507–515, New York, NY, USA. Association for Computing Machinery.

[22] Hua, J., Yu, J. & Yang, M.-S. (2020a) Fast clustering for signed graphs based on random walk gap. *Social Networks*, **60**, 113–128. Social Network Research on Negative Ties and Signed Graphs.

[23] Hua, J., Yu, J. & Yang, M.-S. (2020b) Fast clustering for signed graphs based on random walk gap. *Social Networks*, **60**, 113–128.

[24] 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)*, pages 65–70. IEEE.

[25] Karypis, G. & Kumar, V. (1999) Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Computing 20(1), 359-392. *Siam Journal on Scientific Computing*, **20**.

[26] Knyazev, A. V. (2001) Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM J. Sci. Comput.*, **23**, 517–541.

[27] 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. ACM.

[28] 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. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 559–570. Society for Industrial and Applied Mathematics.

[29] 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*, WWW '10, pages 641–650. ACM.

[30] Leskovec, J. & Krevl, A. (2014) SNAP Datasets: Stanford Large Network Dataset Collection. `http://snap.stanford.edu/data`.

[31] Mercado, P., Tudisco, F. & Hein, M. (2016a) Clustering Signed Networks with the Geometric Mean of Laplacians. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I. & Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4421–4429. Curran Associates, Inc.

[32] Mercado, P., Tudisco, F. & Hein, M. (2016b) *Clustering Signed Networks with the Geometric Mean of Laplacians*.

[33] Mercado, P., Tudisco, F. & Hein, M. (2019a) Spectral Clustering of Signed Graphs via Matrix Power Means. In Chaudhuri, K. & Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4526–4536, Long Beach, California, USA. PMLR.

[34] Mercado, P., Tudisco, F. & Hein, M. (2019b) *Spectral Clustering of Signed Graphs via Matrix Power Means*.

[35] Moore, M. (1978) An international application of Heider's balance theory. *European Journal of Social Psychology*, **8**(3), 401–405.

[36] Moore, M. (1979) Structural balance and international relations. *European Journal of Social Psychology*, **9**(3), 323–326.

[37] Nogueira de Moura, L. (2020) Social network analysis at scale: Graph-based analysis of Twitter trends and communities. Master's thesis, Texas State University.

[38] Read, K. (1954) Cultures of the Central Highlands, New Guinea. *Southwestern Journal of Anthropology*, **10**(1), 1–43.

[39] Rusnak, L. & Tešić, J. (2021) Characterizing Attitudinal Network Graphs through Frustration Cloud. *Data Mining and Knowledge Discovery*, **6**(35).

[40] Saberi, M., Khosrowabadi, R., Khatibi, A., Misic, B. & Jafari, G. (2021) Topological impact of negative links on the stability of resting-state brain network. *Scientific Reports*, **11**(1), 2176.

[41] Sampson, S. (1968) A novitiate in a period of change: An experimental and case study of relationships. Ph.D. thesis, Cornell University.

[42] Sarkees, M. R. & Wayman, F. (2010) Resort to War: 1816 - 2007. `https://correlatesofwar.org/data-sets/COW-war/cow-wars`.

[43] Shi, J. & Malik, J. (2000) Normalized Cuts and Image Segmentation. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, **22**(8), 18.

[44] Steinley, D. (2004) Properties of the Hubert-Arabie Adjusted Rand Index. *Psychological methods*, **9**, 386–96.

[45] 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.

[46] von Luxburg, U. (2007) A Tutorial on Spectral Clustering. *arXiv:0711.0189 [cs]*. arXiv: 0711.0189.

[47] Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S. et al. (2001) Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584.

[48] White, H. C., Boorman, S. A. & Breiger, R. L. (1976) Social Structure from Multiple Networks. I. Blockmodels of Roles and Positions. *American Journal of Sociology*, **81**(4), 730–780.

[49] Yang, B., Cheung, W. & Liu, J. (2007) Community Mining from Signed Social Networks. *IEEE Transactions on Knowledge and Data Engineering*, **19**(10), 1333–1348.

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

[51] Zheng, Q. & Skillicorn, D. (2015) Spectral Embedding of Signed Networks. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 55–63. Society for Industrial and Applied Mathematics.