GraphC: Parameter-free Hierarchical Clustering of Signed Graph Networks

Muhieddine Shebaro *IEEE Student Member*, Lucas Rusnak, Martin Burtscher *IEEE Senior Member*, Jelena Tešić *IEEE Senior Member*

***Abstract*—Spectral clustering methodologies, when extended to accommodate signed graphs, have encountered notable limitations in effectively encapsulating inherent grouping relationships. Recent findings underscore a substantial deterioration in the efficacy of spectral clustering methods when applied to expansive signed networks. We introduce a scalable hierarchical Graph Clustering algorithm denominated *GraphC*. This algorithm is adept at discerning optimal clusters within signed networks of varying magnitudes. *GraphC* endeavors to uphold the integrity of positive edge fractions within communities during partitioning, concurrently maximizing the negative edge fractions between communities. Importantly, *GraphC* obviates the necessity for a predetermined cluster count (denoted as k). Empirical substantiation of *GraphC*'s efficacy is provided through a comprehensive evaluation involving fourteen datasets juxtaposed against ten baseline signed graph clustering algorithms. The algorithm's scalability is demonstrated through its application to extensive signed graphs drawn from Amazon-sourced datasets, each comprising tens of millions of vertices and edges. A noteworthy accomplishment is evidenced, with an average cumulative enhancement of 18.64% (comprising the summation of positive edge fractions within communities and negative edge fractions between communities) over the second-best baseline for each respective signed graph. It is imperative to note that this evaluation excludes instances wherein all baseline algorithms failed to execute comprehensively.**

***Index Terms*—clustering, signed network, community detection, and structural balance.**

I. Introduction

COMMUNITIES within a network are formally defined as agglomerations of vertices characterized by denser interconnections among constituent vertices than with the broader network. These communities may exhibit either distinct vertex sets or overlap, where vertices partake in multiple communities. The discernment of community structure holds paramount importance, unveiling latent insights into relationships and dynamic mechanisms within intricate networks spanning various domains, ranging from biological to social networks. he inherent complexity of these network systems frequently engenders substantial datasets surpassing individual systems' computational capacities, necessitating data partitioning for distributed processing. The efficiency of such partitioning schemes in community detection algorithms is contingent upon various factors. Numerous methodologies, encompassing local optima pursuit, statistical inference, and machine learning, have been posited to unveil community structure. While state-of-the-art community discovery algorithms in unsigned graphs adeptly handle vast networks comprising millions of vertices and edges [1], the modeling of unsigned graphs falls short in capturing the intricate relationships within networks. The ascendancy of signed graph modeling as a more fruitful and meaningful data representation is evident. Recent investigations demonstrate the struggles of contemporary signed graph methods in recovering communities within graphs featuring mere thousands of vertices and hundreds of thousands of edges [2]. Concurrently, extant research underscores the challenges posed by spectral methods in recovering community structure within sparse networks, even with the incorporation of normalization techniques [3]. This aligns with our recent findings, elucidating a pronounced degradation in the performance of spectral methods for clustering large, sparse signed networks derived from authentic data sources [4]. In the realm of signed networks, balance theory stands as a pivotal concept, elucidating the evolution of attitudes within networks. Established by Heider [5] and subsequently formalized mathematically by Harary, who introduced k-way balance [6], [7], balance theory has found applications in predicting edge sentiment, content and product recommendations, and anomaly detection in various domains [8], [9], [10], [11].

A. Spectral Clustering of Real Signed Graphs

The inception of spectral clustering dates back to 1973, as proposed by Donath and Hoffman [12]. This methodology posits that graph partitioning can be achieved by leveraging the eigenvectors of the adjacency matrix. The spectral clustering process can be succinctly encapsulated in four steps: (1) computation of the Laplacian variant and identification of clusters (k); (2) derivation of k eigenvectors corresponding to the k smallest eigenvalues; (3) formation of the eigenvector matrix U to reduce dimensionality; and (4) application of k-means++ to cluster features. The SigNeT package, employed for spectral methods in signed networks, encompasses multiple Laplacian variants, all presumed to be positive semi-definite [13]. The scalability of spectral clustering algorithms is hindered in the context of real signed networks: (i) the substantial time required for the solver to formulate eigenvectors with their associated eigenvalues, and (ii) the susceptibility to spectral pollution or eigenvalue pollution, manifesting as instability in the approximation implementations for large matrices and consequential error fluctuations [14]. A recent survey has provided a proof-of-concept and practical evaluation of these limitations in actual signed graphs [4]. The survey falls short of delineating the breaking points of signed Laplacians concerning algorithmic assumptions (e.g., small world, diameter)

M. Shebaro, L. Rusnak, M. Burtscher, and J. Tešić are with the Texas State University, San Marcos, USA. -mail: m.shebaro, lucas.rusnak, burtscher, jtesic@txstate.edu.

Manuscript received June 1, 2024; revised X Y 2024

and characteristics specific to signed networks (e.g., density, sparsity) [4].

Another significant drawback inherent in contemporary signed graph clustering algorithms pertains to the challenging task of determining the suitable number of communities (k) before algorithmic execution. A common approach for dealing with this is to utilize the elbow method. However, this method is subjective in nature because it introduces variability, rendering it dependent on individual perspectives. Furthermore, the decision-making process is complicated by various factors and network characteristics, including sparsity, average degree, and overall network structure, which collectively influence the selection of the optimal number of communities. The nuanced evaluation of why one community partitioning surpasses another is not clear in the absence of meaningful ground truth, as underscored in [4].

B. Research Contributions

GraphC is a term contraction of a *graph clustering* phrase. *GraphC* is a parameter-free clustering algorithm for signed graphs. G_i notes the signed graph with an unsigned topology G (depicted in Figure 1), and G_{ij} symbolize a collection of vertex sets discovered by clustering algorithm j. *GraphC* reformulates the clustering problem by shifting the emphasis from minimizing the combined count of positive edges between communities and negative edges within communities. The loss is formulated in terms of minimizing the fraction of positive edges *between* communities and the fraction of negative edges *within* communities for algorithm j applied to the signed graph G_i. We quantify the positive edges outside communities as $pos_{out}(G_{ij})$ (Eq. 4), and the negative edges within communities as $neg_{in}(G_{ij})$ (Eq. 5) for algorithm j operating on the signed graph G_i. *GraphC* identifies optimal clusters without the need for a predefined k and a spectral decomposition solver. **Paper Contributions:**

1) *Spectral-Balance duality:* we prove that Harary cuts can be obtained using a spectral approach from a *balanced* signed network using the eigenvector with a 0 eigenvalue that expresses the sentiment between vertices.
2) *GraphC* implements this theoretical novelty and directly performs Harary cuts on the balanced signed graph by purging the negative edges than it is using Laplacian.
3) *GraphC* algorithm can process signed graphs with tens of millions of vertices and edges. Parameterized trade-off between speed and the overall quality of the clustering that allows the algorithm to scale on a single workstation.
4) *GraphC* is k-independent: it does not require a predefined number of clusters to produce a high-quality clustering. The parameters used in the algorithm are dependent on computing resources.
5) *GraphC* addresses the positive-negative edge imbalance that naturally occurs in most signed networks as it takes equally into account the contribution of positive edges and the negative edges in the graph for clustering optimization.

II. Related Work

The diapason of existing approaches for community detection in signed networks forms a diverse landscape. Xhiang et al. propose an algorithm for clustering signed graphs utilizing the balanced normalized cut, demonstrating commendable clustering results and recognized as an efficient approach [15]. The *Signed Positive over Negative Generalized Eigenproblem* spectral-based approach necessitates the specification of k as it formulates clustering as a generalized eigenvalue problem, enhancing a well-defined objective function [16]. The approach relying on the signed Hessian for clustering also requires a predefined k. It emphasizes the advantages of non-backtracking operators with the computational and memory efficiencies inherent in real symmetric matrices [17].

He et al. [18] use modified social network analysis and triangle balancing heuristics ("friend of my friend is my friend") to address the issue for cluster discovery based on a modified version of Heider balance theory. First, the authors employ a signed mixed-path aggregation (SIMPA) method to construct the vertex embedding. Consequently, this vertex embedding is utilized to produce probabilities for cluster assignments. The clustering process involves training with a weighted combination of supervised and unsupervised loss functions, with the unsupervised component being a probabilistic balanced normalized cut. The algorithm uses labels from SPONGE (SPO) for some signed graphs to execute the clustering, building a dependency on other signed graph clustering algorithms. Combining positive and negative Laplacians using matrix power means is also proposed as a modification of the Laplacian [19]. Note that the authors do not provide an implementation for the baseline comparison in [19]. Graph neural networks (GNN) that rely on the social balance theory have been used for signed network clustering as well and are typically used on data with a significant positive class imbalance in [20] and [21]. The primary obstacle in employing graph neural networks for categorizing signed networks lies in the accurate management of negative connections and the integration of both positive and negative connections into a unified model that can effectively deduce node representations [22]. On the the hand, some researchers [23] argue that using opposing edges in Graph Neural Networks (GNNs) when developing a recommendation system for signed networks can be problematic. They believe that these opposing edges can disrupt the network's homophily, which is the tendency of nodes to connect with similar ones. This disruption can make the transmission of messages to dissimilar nodes ineffective, potentially compromising the performance of the recommendation system. Hence, careful consideration is needed when designing GNNs for such applications.

Gholami et al. [24] propose a new approach for detecting overlapping communities based on Neutrosophic theory, and it consists of two phases. In the initial stage, the Signed Graph Convolutional Network (SGCN) is used to position nodes within a space of reduced dimensions. This step effectively encapsulates the architecture of the signed network and portrays it in a more compact space. Following this, the neutrosophic c-means clustering method is employed in the subsequent stage

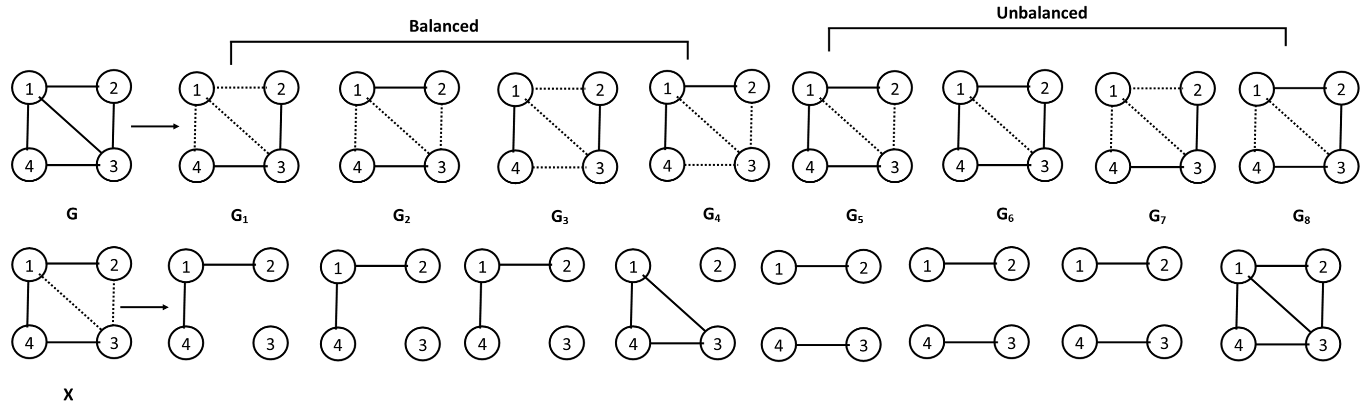

Fig. 1. **Dotted** lines are negative edges whereas the **sold** lines are positive edges. Top: A graph G, a balanced graph G_1 obtained by switching (changing the sign of the edges originating in) v_1, a balanced graph G_2 obtained by switching v_1 and v_2, a balanced graph G_3 by switching v_2 and v_3, and a balanced graph G_4 by switching v_4. G_5, G_6, G_7 and G_8 are examples of unbalanced states. Bottom: Harary Cuts of nearest stable states introduced in [25] from G_5 graph.

to allocate nodes to various communities.

III. Balanced-Spectral Duality in Signed Networks

In this section, we lay the groundwork for the terminology and properties of baseline signed graphs and introduce the duality theory of near-balanced states of the signed graph and spectral analysis.

A. Fundamental Cycle Basis

Definition 3.1: Graph G_i is a **subgraph** of a Graph G if **all** edges and vertices of G_i are contained in G.

Definition 3.2: **Path** is a sequence of distinct edges m that connect a sequence of distinct vertices n in a graph. **Connected Graph** has a path that joins any two vertices **Cycle** is a path that begins and ends at the same vertex **Cycle Basis** is a set of simple cycles that forms a basis of the cycle space.

Definition 3.3: For the underlying Graph G, let T be the spanning tree of G, and let an edge m be an edge in G between vertices x and y that is *NOT* in the spanning tree T Since the spanning tree spans all vertices, a unique path in T between vertices x and y does not include m. **The fundamental cycle** is any cycle that is built using path in T plus edge m in graph G.

Corollary 3.1: A fundamental cycle basis may be formed from a spanning tree or spanning forest of the given Graph by selecting the cycles formed by combining a path in the tree and a single edge outside the tree. For the Graph G with N vertices and M edges, there are precisely $M - N + 1$ fundamental cycles.

B. Balanced Signed Graphs

Definition 3.4: **Signed graph** $G = (G, G)$ consists of underlying unsigned graph G and an edge signing function $G : m \rightarrow \{+1, -1\}$. The edge m can be positive m^+ or negative m^-. **Sign** of a sub-graph is *product* of the edges signs. **Balanced Signed graph** is a signed graph where every cycle is positive **Frustration** of a signed graph is defined as the number of candidate edges whose sign needs to be switched for the graph to reach the balanced state Figure 1 (Top, balanced states) shows an example signed network and its balanced states.

Theorem 3.1 ([6]): If a signed subgraph G' is balanced, the following are equivalent:

1) G' is balanced. (All circles are positive.)
2) For every vertex pair $(v_i, v_j) in G'$, all (v_i, v_j)-paths have the same sign.
3) $Fr(G') = 0$.
4) A vertex set can be divided into two groups, U and W. This division of the vertex set into (U, W) is known as the Harary-bipartition.

A Harary cut is defined as the deletion of the opposing edges in each balanced state in the frustration cloud of the signed Graph as illustrated in Figure 1 (Bottom). The trivial unbalanced signed Graph in this Figure consisting of 4 vertices and five edges yields eight total nearest balanced states. Note that a Harary cut can yield multiple connected components, and the input graph might contain many initially connected components where Harary cuts and balancing take place in every element. Prior, we have characterized signed graphs through the frustration cloud [25], a collection of nearest balanced states, and then introduced an improved and parallelizable way to discover the nearest balanced states in extensive signed data efficiently [26].

C. Spectral Clustering and Balanced States

A signed graph is considered *balanced* if the graph has no cycles with an odd number of negative edges. *switch* operation involves changing the sign of all edges connected to a specific vertex as observed when switching v_1 in G to obtain G_1 in Figure 1. Switching equivalence in this context means that two balanced signed graphs can be transformed into each other through a series of switch operations, as shown in the following example. The balanced signed graphs for the same underlying Graph G in Figure 1 (Top) are switching equivalents [27]. G_2 and G_3 in Figure 1 (Top) are switching equivalent because we can obtain G_3 by switching v_1 and v_3 in G_2. The critical point is that switching operations preserve the balanced nature of the Graph as they do not alter the overall balance or imbalance of the Graph [25]. Not only do we get the same eigenvalues, but the eigenvectors only differ by a multiple of -1 for each vertex switched.

TABLE I
BALANCED SIGNED GRAPHS G, G_1, AND G_2 ARE ISOSPECTRAL: THEY HAVE IDENTICAL EIGENVALUES, AND THEIR EIGENVECTORS DIFFER. G_5 IS AN UNBALANCED GRAPH, AND ITS EIGENVALUES AND EIGENVECTORS DIFFER.

	Balanced Graphs												Unbalanced Graph			
Graph	G				G_1				G_2				G_5			
Eigen values	0	2	4	4	0	2	4	4	0	2	4	4	$2-\sqrt{2}$	$3-\sqrt{3}$	$2+\sqrt{2}$	$3+\sqrt{3}$
Eigen vectors	1	0	1	0	1	0	1	0	1	1	1	1	1	1	-1	1
	1	1	0	1	-1	1	0	1	1	1	0	1	$\sqrt{2}$	0	$\sqrt{2}$	0
	1	0	-1	-2	-1	0	1	-2	-1	0	1	2	-1	1	1	1
	1	-1	0	1	-1	-1	0	1	-1	1	0	-1	0	$\frac{1+\sqrt{3}}{2}$	0	$\frac{1-\sqrt{3}}{2}$

Note that the spectral clustering results in zero eigenvalues as many times as there are clusters in the data [27]. The zero eigenvalues have an eigenspace with a basis of vectors that only have zeros and ones, and these vectors indicate which vertices belong to each cluster [27]. In Table I, for a connected graph G of 4 vertices with all positive edges, the eigenvector with the 0 eigenvalue consists entirely of ones.

All balanced signed graphs are switching equivalent to all positive signed graphs with the same topology. This implies that when performing spectral analysis on such graphs, the eigenvalues remain the same, as illustrated in Table I. The corresponding eigenvectors only differ by a multiple of -1 for each switched vertex, as illustrated with G, G_1, and G_1 eigenvalues and eigenvectors from Figure 1 in Table I. This property makes the study of spectral characteristics more manageable, as it allows for a direct comparison between different balanced signed graphs while preserving their essential structural features. The multiplicity of the 0 eigenvalue counts the number of connected components of the underlying graph when balanced. The Laplacian for a balanced signed graph G_{ij} can be decomposed into

$$\mathbf{L}_G = \mathbf{I}_W \mathbf{L}_G \mathbf{I}_W \tag{1}$$

where G is the Laplacian of the underlying Graph, and $\mathbf{I}_W$ is the identity matrix, except the diagonal entries corresponding to the vertices of $W \subseteq V$ are -1.

Theorem 3.2: The Laplacian matrices of balanced signed graphs of the same underlying Graph are isospectral. Table I exemplifies this using the same Graph G in Figure 1 (Top, balanced states) whereas G_5 column in Table I shows the spectral clustering approach with an unbalanced state.

Proof 3.1: Switching does not change the cycle signs. The signs of its cycles determine the characteristic polynomial [28].

Theorem 3.3: For a connected balanced signed graph G_{ij}, the entries of the 0-eigenvector are $+1$ or -1. Partitioning the vertices along the entry values of the 0-eigenvector corresponds to the Harary cut of the balanced signed Graph.

Proof 3.2: Since G_{ij} is balanced, it is isospectral to the underlying Graph G. Since G is connected, the dimension of the 0-eigenspace is 1, and the vector $\mathbf{1}$ consisting of all 1's is a basis for that eigenspace Switching a vertex v negates both row and column v in $\mathbf{L}_G$ to produce $\mathbf{L}_G$, thus negating the v-entry of the basis vector $\mathbf{1}$.

On the other hand, if a signed graph is not balanced, 0 is not an eigenvalue, and the structural and sentiment framework of the signed graph *dissolves into the geometry*, as illustrated in Table I for G_5. Spectral methods determine outcomes based on the geometry of the eigenvectors. If there is no eigenvector associated with selecting the graph structure (no 0 eigenvector), the graph structure is not directly considered when unbalanced.

Thus, we conclude that the Harary cuts can also be performed using a spectral approach, and vice versa. After balancing a signed network, we can construct its balanced Laplacian matrix, and the spectral decomposition of the matrix yields the eigenvectors with their associated eigenvalues. f balanced d and connected, there is a 1 dimensional kernel, and the associated 0 eigenvector entries give us the Harary cut. If the graph is disconnected, this may be performed on each connected component for each dimension of the kernel.

IV. MEASURING SIGNED GRAPH CLUSTERING EFFICACY

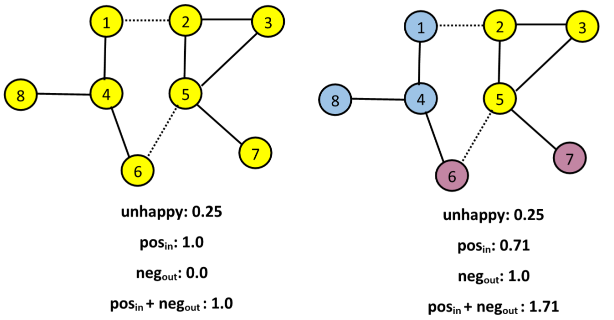

Fig. 2. An illustration emphasizing the advantage of measuring the quality of a clustering assignment in a signed graph using the summation of the fraction of positive edges within communities and the fraction of negative edges between communities. **Dotted** lines are negative edges whereas the **sold** lines are positive edges.

Consider two clustering assignments for the network illustrated in Figure 2. In the absence of ground truth and training data for the community assignment in signed networks, state-of-the-art research uses the *unhappy ratio*. he *unhappy ratio* U_i is defined in Eq. 2 for a set of communities produced by method j in the signed graph G_i:

$$U_{ij} = \frac{pos_{between} + neg_{within}}{pos_{between} + pos_{within} + neg_{between} + neg_{within}} \tag{2}$$

The $pos_{between}$, pos_{within}, neg_{within}, and $neg_{between}$ are the number of positive edges between communities, the number of positive edges within communities, the number of negative edges within communities, and negative edges between communities, respectively. Note that the *unhappy* ratio was used to measure the quality of the clustering assignment in SSSNet experiments [18].

The *unhappy* ratio for both clustering approaches in Figure 2 is 0.25. he clustering assignment on the right seems more effective as it avoids disregarding *all* negative edges, successfully segregates two communities (1,4,8 and 2,3,5), and accurately classifies *most* positive edges. he *unhappy* ratio does not capture that measure. t has been shown that the measure fails to capture the quality of clustering real signed networks, as the *unhappy* ratio favors positive edges and trivial clustering cases in networks where the typical number of negative edges is 5-10 times smaller [4].

Here is an example: a signed graph with 1000 edges is given, where 900 edges are positive and the remaining 100 are negative. ets assume the first clustering has 400 $pos_{between}$, 500 pos_{within}, 50 neg_{within}, and 50 $neg_{between}$, resulting in an unhappy score U_1 of 0.45. The second clustering has 0 $pos_{between}$, 900 pos_{within}, 100 neg_{within}, and 0 $neg_{between}$, resulting in an unhappy score U_2 of 0.1. The distribution of positive and negative edges in this hypothetical signed Graph is imbalanced. Based on the unhappy scores, one might conclude that the second clustering is superior. A closer inspection reveals that the second clustering is trivial, as it assigns all vertices to one cluster. he first clustering, on the other hand, can correctly separate some of the negative edges across clusters and preserve some of the positive edges within clusters. Thus, the first clustering should be the optimal solution, which contradicts the unhappy ratio measure. his is analogous to the accuracy measure in a machine-learning setting when the trivial class is favored in binary classification.

On the other hand, measuring the summation of the fraction of positive edges within communities (pos_{in}) and the fraction of negative edges (neg_{out}) captures the clustering quality well for Figure 2, [4]. he unhappy score for both clustering assignments is the same (0.25) even though the left assignment did not account for *any* negative edges while the right figure accounted for both types of edges. or our proposed measurement, the left has a summation of 1.0, whereas the right has a summation of 1.71, which reflects the true clustering quality. n this work, we redefine the *unhappy ratio* to count for *both* positive and negative edges as:

$$U_{ij} = \frac{pos_{between}}{pos_{between} + pos_{within}} + \frac{neg_{within}}{neg_{within} + neg_{between}} \tag{3}$$

Our goal is to minimize the new *unhappy ratio*, that is, the loss measure for a signed graph G_i and clustering algorithm j. rewrite it as a loss function, as our objective in clustering is now to minimize $\mathcal{L}(G_{ij})$ by splitting nodes into communities. We define new pos_{out} and neg_{in} as

$$pos_{out} = \frac{pos_{between}}{pos_{between} + pos_{within}} \tag{4}$$

$$neg_{in} = \frac{neg_{within}}{neg_{within} + neg_{between}} \tag{5}$$

The goal is reformulated to minimize the fraction of violating negative edges $neg_{in}(G_{ij})$ as well as the fraction of violating positive edges $pos_{out}(G_{ij})$ simultaneously. e define the violating edges in this context as the total number of positive edges between clusters and negative edges inside them.

$$U_{ij} = \mathcal{L}_{G_{ij}} = pos_{out} + neg_{in} \tag{6}$$

Symmetrically, we redefine pos_{in} (Eq. 7) and neg_{out} (Eq. 8) measures in terms of fractions to account for the positive and negative edge imbalance.

$$pos_{in} = \frac{pos_{within}}{pos_{between} + pos_{within}} \tag{7}$$

$$neg_{out} = \frac{neg_{between}}{neg_{within} + neg_{between}} \tag{8}$$

Next, we generalize the loss function to accommodate for the graph characteristics:

$$\mathcal{L}_{G_{ij}}(\alpha, \beta) = \beta(\alpha pos_{out} + (1-\alpha) neg_{in}) + (1-\beta) * \frac{|V_{iso}|}{|V|} \tag{9}$$

The α controls the importance between pos_{out} and neg_{in}. Setting α =0.5 gives equal importance to both. e β parameter controls the fraction of isolated vertices $|V_{iso}|$ produced in the process; —V— is the total number of vertices in the graph. f $\beta = 0$, then the algorithm will completely ignore minimizing the pos_{out} and neg_{in} and will find the cuts that yield the least amount of isolated vertices possible. ote that $\mathcal{L}_{G_{ij}}$ in Eq. 6 is equal to $\mathcal{L}_{G_{ij}}(0.5, 1)$ in Eq. 9.

V. GraphC Methodology

In Section III, we have shown that the Laplacian matrices of balanced signed graphs of the same underlying Graph are isospectral and that a Harary cut can be performed through a spectral approach as well. he proposed *GraphC* approach searches for the best Harary cut based on a quality measure 9 and a set of stopping criteria to automatically halt the algorithm execution to return the clustering labels of each node. The *GraphC* algorithm flow is illustrated in Figure 3. First, a signed network might contain several initial connected components. We start by labeling all vertices that are in the same connected component with the same label to produce an initial clustering label assignment set $\mathcal{C}$ as shown in Algorithm 1. Isolated vertices are given their unique labels. or two isolated nodes, the algorithm assigns different labels to each node. ore specifically, the algorithm first checks if the connected component at hand is in a set called 'processed.' Then, it restores the original signs using the original signed network if it isn't in that set. ext, it checks if it has surpassed the time limit and ensures that the size of this connected component is more significant than $Gamma$. After the algorithm finds the best Harary cuts based on the loss function defined in Eq. 9, it checks whether the Harary split improves the overall quality of the clustering assignments according to Eq. 10. Finally, the algorithm commits that split if it leads to an improvement greater than ϵ.

Selecting the Best Harary Cut We use GraphB+ to create several stable states and perform Harary cuts. To find the best Harary cut, we pick the one that results in the slightest loss, as defined in Equation 9. After selecting the Harary cut, we employ the Depth-First Search (DFS) to create the connected components. BFS starts at the root vertex and explores each branch as far as possible before backtracking. t continues this process until it has visited all vertices connected to the root vertex. The time it takes for DFS to run is proportional to the number of vertices ($|V|$) plus the number of edges ($|E|$).

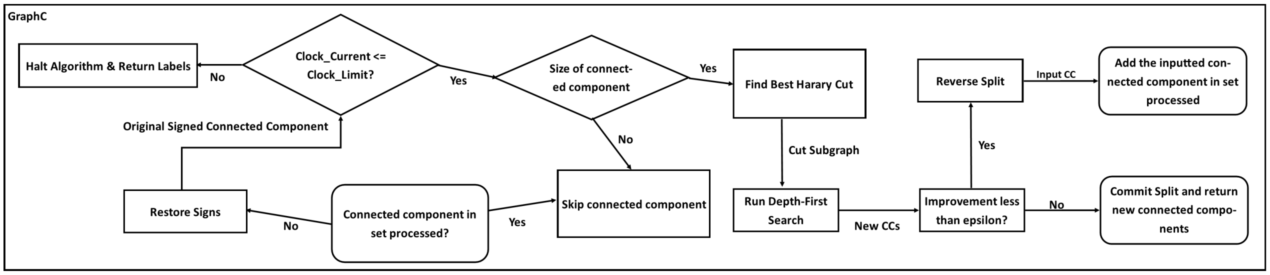
GraphC
Halt Algorithm & Return Labels
No
Clock_Current <= Clock_Limit?
Yes
Size of connected component
Yes
No
Find Best Harary Cut
Cut Subgraph
Reverse Split
Input CC
Add the inputted connected component in set processed
Yes
Original Signed Connected Component
Restore Signs
No
Connected component in set processed?
Yes
Skip connected component
Run Depth-First Search
New CCs
Improvement less than epsilon?
No
Commit Split and return new connected components

Fig. 3. The *GraphC* pipeline: the starting state is ”Connected component in set processed?” block.

```
Algorithm 1: Split and Label Components
  Data: $G_{ij}$, label_counter
  Result: Clustering label assignment set
          $\mathcal{C} = \{C_v\}, v \in V$
  label_counter=0;
  for $CC,\ CC \in G_{ij}$ do
      for $v,\ v \in CC$ do
          $C_v$=label_counter;
      end
      label_counter++;
  end
```

The process of selecting the best Harary cut is shown in Algorithm 2.

```
Algorithm 2: Best Harary Cut
  Data: $G_{ij}$, Set R, label_selected, $\mathcal{C}$, $\alpha$, $\beta$, $I$
  Result: $G^{FC}$
  for every vertex $v$ in $G_{ij}$ do
      if $C_v \neq$ label_selected then
          R = R + $v$ ;
      end
  end
  $G^F = G \setminus R$;
  $G'$ =GraphBplus($G^F, I$) ;
  $G^{FC} = G' | \{min\mathcal{L}_{\alpha,\beta,G^F}\}$;
```

Stopping Criteria: After finding the best Harary cut of a particular connected component, the algorithm will compute the overall quality of the new clusters of the entire signed network, which is similar to Equation 9 and it is as follows:

$$\mathcal{L}^t_{G_{ij}} = (pos_{out} + neg_{in}) \qquad (10)$$

Suppose the new clusters yield an improvement less than ϵ, which is also predefined (similar to the condition k-means++'s stopping case). In that case, the algorithm is going to undo the last split of that connected component and add that connected component into a set called $processed$ such that if the algorithm iterates again to look for other connected components to split, whether they are newly formed or not, the algorithm is going to ignore any connected components that are in set $processed$. The algorithm is going to restore the original signs of the target-connected component before being fed to GraphB+. On the other hand, if it leads to an improvement of more than ϵ, then the algorithm will commit the changes and proceed to the next eligible connected component. Algorithm 3 demonstrates the entire algorithm execution. Finally, an optional time limit t_l can be inputted such that if the algorithm exceeds that limit, the algorithm will halt and return the labels. Note that setting the time limit as -1 means that the algorithm can execute for unlimited time as long as other stopping criteria have not been triggered yet.

Diminishing Returns (Fostering Scalability): An intrinsic phenomenon in hierarchical clustering algorithms is that the deeper we go, the fewer potential improvements we get. Finding the best Harary cut of every connected component is computationally expensive, especially on massive signed graphs, as they might have millions of relatively small connected components, which could potentially yield marginal improvements if any at all. his justifies the use of the $Gamma$ optional parameter to save computation. t represents a trade-off between efficiency and performance. If a connected component's size is less than $Gamma$, the algorithm is going to ignore that component and will not find the best Harary cut even if it is not in set $processed$. Figure 4 illustrates a typical execution of the algorithm on the PPI signed Graph, and it shows the gradual decreasing improvements over each Harary split. he overall improvement started from around 0.7 to a meager 0.0007 in the 5th Harary cut.

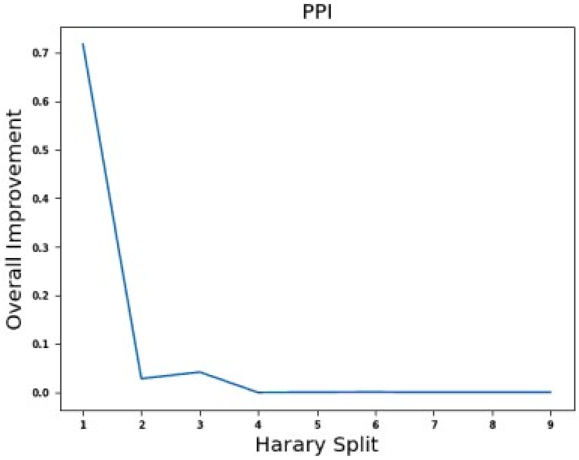

Fig. 4. Example output of the overall improvements with each **committed** Harary split of every connected component of our proposed algorithm on PPI signed graph.

Algorithm 3: Choose component to split

```
Data: Signed graph $G_{ij}$, spanning trees sampling
      method $M$, $I$, $\alpha$, $\beta$, $\epsilon$, $Gamma$, Time limit $t_l$
Result: Clustering label assignment set
        $\mathcal{C} = \{C_v\}, v \in V$
label_counter=0;
Call Alg. 1 with inputs $G, label\_counter, \mathcal{C}$;
set R = $\emptyset$, processed = $\emptyset$;
while true do
    if $t \geq t_l$ then
        break;
    end
    label_selected=-1 ;
    terminate=true ;
    for label $l, l \in \mathcal{C}$ do
        if $|CC_l| \leq Gamma$ then
            continue;
        end
        if $l$ is not in set processed then
            label_selected=$l$;
            terminate = false;
        end
    end
    if terminate = true then
        break;
    end
    $\mathcal{C}_t = \mathcal{C}$;
    label_counter_temp = label_counter ;
    $G^{FC}$ = Call Alg. 2 ($G_{ij}$, Set R,label_selected,
      $\mathcal{C}$,$\alpha$, $\beta$, $I$);
    for $CC \in G^{FC}$ do
        for vertex $i \in CC$ do
            $\mathcal{C}_\rangle$=label_counter;
        end
        label_counter++;
    end
    Compute $\mathcal{U}_{t+1,G}$ ;
    if $\mathcal{U}_{t,G} - \mathcal{U}_{t+1,G} \leq \epsilon$ then
        $\mathcal{C} = \mathcal{C}_t$;
        label_counter = label_counter_temp;
        Append label_selected to processed;
        break;
    end
    Set $\mathcal{U}_{t,G} = \mathcal{U}_{t+1,G}$ ;
    $R = \emptyset$ ;
end
```

Illustrative Execution on Highland Tribes Figure 5 shows the execution of our proposed algorithm. Assume that I = 1000, $\alpha = 0.5$ and $\beta = 1$, $Gamma = 2$, $\epsilon = 0.00000001$, and t_l = -1. First, the algorithm checks if it exceeded the time limit t_l; it proceeds since there is no time limit. hen, the algorithm will check whether the initial component at level 0 is in $processed$; it proceeds to the next step because the $processed$ set is empty initially. t restores the original signs and finds the best Harary cut for this component. This will result in two connected components at level 1. The same above steps are repeated for both components. The component on the left at level 0 will trigger a stopping criterion, which is when the improvement is less than $epsilon$ after performing the Harary split. The algorithm will undo the split and commit the connected component with a red dashed circle, and the vertices within that component will be given a unique final label. Eventually, the two components at level 1 will also trigger the same stopping criteria, resulting in 3 clusters.

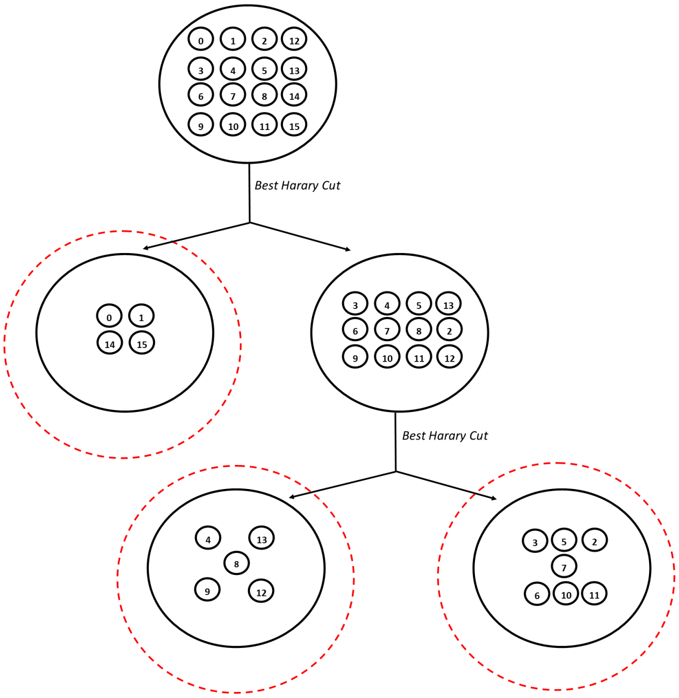

Fig. 5. The execution of the *GraphC* algorithm on a Highland signed graph comprising 16 vertices and 58 edges.

VI. Proof of Concept Implementation

A. Baselines

For comparing the performance of our proposed algorithm, we use ten baselines, and they are as follows:

- **Laplacian_none** [29]: spectral clustering using the signed graph Laplacian.
- **Laplacian_sym** [29]: spectral clustering using the symmetric Laplacian.
- **BNC_none** [15]: balanced normalized cuts.
- **BNC_sym** [15]: symmetric balanced normalized cuts.
- **SPONGE_none** [16]: baseline SPONGE implementation.
- **SPONGE_sym** [16]: symmetric SPONGE implementation.
- **Hessian** [17]: clustering based on signed Bethe Hessian.
- **A_sym** [30]: spectral clustering using symmetric adjacency matrix.
- **dns** [31]: clusters the Graph using eigenvectors of the bns Laplacian matrix.
- **sns** [31]: clusters the Graph using eigenvectors of the sns Laplacian matrix.

B. Implementation

The proposed algorithm's implementation is in C++. The algorithm automatically discovers optimal communities without a predefined number of clusters k. t finds and commits the optimal Harary cut for each connected component if it satisfies the following conditions: it is not in set $processed$, did not exceed the time limit t_l, its size is more significant than $Gamma$, that Harary cut leads to an improvement greater than ϵ. The parameters used for all Konect signed graphs across the board are I = 1000, α = 0.5, β = 1, ϵ = 0.00000001, $Gamma$ = 2, and t_l = 100000 except WikiConflict and WikiPolitics where $Gamma$ = 10 to speed up the execution. or the baseline methods, the implementation is in Python, and kmeans++ is used to cluster the corresponding matrices with n_init =10 (number of times the k-means++ algorithm is run with different centroid seeds) and default values were used for other parameters for k-means++ in the *sklearn* python package [32] and SigNet [30] is used to construct the matrices. he k parameter for these baselines is chosen based on two recent research papers [18][4], and they are defined in Table II. f a signed graph does not have a known used k hyperparameter in the literature, we assigned a k to it that is equal to the k (known) of the most similar signed Graph in terms of size. or all signed graphs, preprocessing was applied where self-edges, inconsistent edges, and duplicate edges (first is kept) are purged. Neutral edges are treated as positive edges. The parameters used for all Amazon signed graphs across the board are I = 50, α = 0.5, β = 1, ϵ = 0.00000001, and $Gamma$ = 40, t_l = -1. he code for running the baselines and the implementation of our algorithm is available in https://anonymous.4open.science/r/graphC-2046/.

C. Setup

The operating system used for the experiments is Linux Ubuntu 20.04.3, running on the 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz with 16 physical cores. It has one socket, two threads per core, and eight cores per socket. The architecture is X86_x64. The GPU is Nvidia GeForce RTX 3070 and has 8GB of memory. Its driver version is 495.29.05, and the CUDA version is 11.5. The cache configuration is L1d : 384 KiB, L1i : 256 KiB, L2 : 4 MiB, L3 : 16 MiB. The CPU op is 32-bit and 64-bit.

D. Signed Graph Datasets

Konect and other signed graphs and their characteristics are described in Table II. *Highland* is the signed social network of tribes of the GahukuGama alliance structure of the Eastern Central Highlands of New Guinea, from Kenneth Read [33]. *Sampson25*, which models sentiment over time between novice monks in a New England monastery captured by Sampson [34]. *Congress* is a signed network where vertices are politicians speaking in the United States Congress, and a directed edge denotes that a speaker mentions another speaker [33]. In the *Chess* network, each vertex is a chess player, and a directed edge represents a game with the white player having an outgoing edge and the black player having an ingoing edge. The weight of the edge represents the outcome [33]. *BitcoinAlpha* is a user-user trust/distrust network from the Bitcoin Alpha platform on which Bitcoins are traded [33]. *BitcoinOTC* is a user-user trust/distrust network from the Bitcoin OTC platform on which Bitcoins are traded [33]. *TwitterRef* captures data from Twitter concerning the 2016 Italian Referendum. Different stances between users signify a negative tie, while the same stances indicate a positive link [35]. *WikiElec* is the network of users from the English Wikipedia that voted for and against each other in admin elections [33]. *SlashdotZoo* is the reply network of the technology website Slashdot. Vertices are users, and edges are replies [33]. The edges of *WikiConflict* represent positive and negative conflicts between users of the English Wikipedia [33]. *WikiPolitics* is an undirected signed network that contains interactions between the users of the English Wikipedia that have edited pages about politics. Each interaction, such as text editing and votes, is given a positive or negative value [33]. *Epinions* is the trust and distrust network of Epinions, an online product rating site. It incorporates individual users connected by directed trust and distrust links [33]. The characteristics of the signed graphs are listed in Table II. *PPI* models the protein-protein interaction network [18]. *WikiRfa* describes voting information for electing Wikipedia managers [18].

Amazon dataset consists of *seventeen* signed graphs derived from the Amazon rating and review files [36]. The dataset contains product reviews and metadata from Amazon, spanning May 1996 to July 2014. Rating score is mapped into an edge between the user and the product as follows $(5, 4) \rightarrow m^+$, $3 \rightarrow m$ (no sign), and $(2, 1) \rightarrow m^-$ [36]. The characteristics of the most significant connected component are outlined in Table IV. For example, clustering a bipartite graph of users and items in a recommendation system context can reveal groups of users who have similar preferences, as well as groups of items that are popular among certain types of users. Another example is clustering a bipartite graph of authors and papers, which can identify research communities and topics.

VII. Konect Signed Graphs Experiment

The number of edges of the most significant connected component is computed locally in our machine with pre-processing techniques such as removing duplicate and inconsistent edges. Our algorithm also detects the number of clusters (including isolated vertices) and the number of Harary split operations executed for various signed networks.

The algorithms are evaluated using pos_{in} and neg_{out}, which are the fraction of positive edges that are within communities and the fraction of opposing edges that are between communities First, we intend to observe and visualize the effect of increasing the number of subsequent Harary splits (that cause an overall improvement in Equation 10) on the two evaluation metrics Figure 6 represents the execution of our algorithm on the WikiElec signed Graph Initially, pos_{in} = 1.0, neg_{out} = 0.0 because all edges are in one cluster For each split, our algorithm cuts the connected component in a way that minimizes the pos_{within} lost and maximizes the $neg_{between}$ gained. Moreover, losing pos_{within} with each split is inevitable

TABLE II
KONECT + BENCHMARK: KONECT [33] PLUS TWITTERREFERENDUM [35], PPI [18], AND WIKIRFA [18] SIGNED GRAPHS. LCC STANDS FOR THE LARGEST CONNECTED COMPONENT, k IS THE PREDEFINED NUMBER OF CLUSTERS FOR SPECTRAL CLUSTERING METHODS, AND *GraphC* RESULTS INCLUDE THE RESULTING NUMBER OF CLUSTERS WITH MORE OR EQUAL TO 5 ELEMENTS AS WELL AS CLUSTERS WITH LESS THAN F VE ELEMENTS. THE LAST COLUMN IS THE NUMBER OF SPLITS PRODUCED AFTER THE ALGORITHM'S EXECUTION.

	LCC		k	GraphC		
Dataset	**# vertices**	**# edges**	**# clusters**	**# clusters$\geq$5**	**# clusters$<$5**	**# splits**
Highland	16	58	3	1	2	2
Sampson25	25	165	4	2	2	3
Congress	219	521	11	2	9	3
PPI	3,058	11,860	10	29	887	16
BitcoinAlpha	3,775	14,120	10	14	201	17
BitcoinOTC	5,875	21,489	20	22	518	10
Chess	7,115	55,779	30	46	1,014	41
TwitterRef	10,864	251,396	50	4	234	8
SlashdotZoo	79,116	467,731	100	245	14,603	53
Epinions	119,130	704,267	100	584	27,498	237
WikiRfa	7,634	175,787	30	25	1,419	24
WikiElec	7,066	100,667	30	38	1,528	25
WikiConflict	113,123	2,025,910	100	208	66,083	32
WikiPolitics	137,740	715,334	100	875	18,964	89

The rate of increase of $neg_{between}$ is far greater than the rate of decrease of pos_{within} edges during the splits from 1 to 10, leading to significant improvement in the overall quality of the clusters Beyond ten cuts, the improvements reach an expected plateau where gradually decreasing improvements occur as a result of cutting more deeply into the connected components This is where the optional $Gamma$ parameter comes into play, where we can prompt the algorithm to halt before or right at the plateau to save running time Table II presents the resultant number of clusters (including isolated vertices) detected as well as the number of Harary split operations executed for the benchmarks It is noteworthy to mention that methods that possess the k parameter usually fuse the supposedly isolated vertices into arbitrary clusters to comply with the restrictions of what the k parameter imposes However, our algorithm is able to freely separate the isolated vertices and detect as many clusters as possible as long as it yields an improvement more significant than ϵ in the overall value in Equation 10 Finally, the performance and execution time of our proposed algorithm are examined against ten baselines, as shown in Table III The empty values in some of the cells are inputted for two reasons Either because the corresponding baseline took more than two days of execution or because some of the eigenvectors during the spectral decomposition did not converge, especially for large signed graphs, causing an error In all tested signed graphs of diverse sizes and densities, our proposed algorithm achieves superior and greatest pos_{in} and neg_{out} While the proposed algorithm's execution time is more significant than some of the methods on some graphs, such as WikiElec, this is ascribed to the fact that the choice of k in these methods is relatively small, which causes them to have a lower execution time. Furthermore, these baselines swiftly deteriorate with increasing graph size and k parameter, causing them to have a much longer execution time than our algorithm.

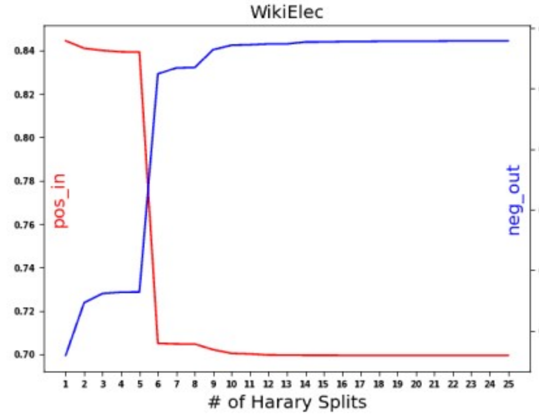

Fig. 6. pos_{in} and neg_{out} graph for WikiElec with each Harary split during the algorithm execution.

VIII. Amazon Ratings and Reviews Modeling

Amazon ratings and reviews [36] are mapped to signed graphs where edges are negative for 0,1,2 ratings, have no sign for three ratings, and have a plus sign for 4 and 5 ratings We prove that our algorithm is capable of scaling to a signed network with millions of vertices and edges by running it on the Amazon signed graphs as in Table IV For instance, *GraphC* successfully ran the most extensive Amazon graph (Book) with a reasonable amount of time, which is 31 hours with high pos_{in} and neg_{out} Therefore, our algorithm is capable of detecting clusters automatically in a scalable fashion In addition, from Table IV, we can observe that the bipartite graph of Amazon reviews/ratings has a skewed distribution of vertices and edges, where most of the vertices are sparsely connected, and only a few vertices dominate the market For instance, for the Books signed graph, about 1 million clusters contain less than five users/books, whereas a relatively meager 32 thousand clusters have more than five users/books Although these clusters contain a mix of users and items and no quantification of the number of users each item has in its cluster is done, this may correlate with the "Long Tail Phenomenon" [37] where a large number of products or services that are not very popular can jointly have a significant share of the market that may overtake or be comparable to the concurrent bestsellers that are very successful.

A. Parameters Study & Analysis

In order to better understand the parameters I, $Gamma$, and ϵ and their effect on the clustering process, we run *GraphC* on the Chess signed graph. First, we start by running the algorithm with $Gamma \in \{2, 1000, 2000, 2500, 3500\}$ with the other parameters fixed as I = 50, α = 0.5, β = 1, ϵ = 0.00000001, and t_l = -1. As observed in the four graphs in Figure 7, increasing $Gamma$ deteriorates the clustering quality where the number of splits and the execution time decrease, resulting in a lower number of communities. This is ascribed to the fact that a higher value of $Gamma$ means that the algorithm will disregard even larger connected components that have the potential to improve the overall quality of

TABLE III
EVALUATION RESULTS OF OUR PROPOSED ALGORITHM AGAINST TEN DIFFERENT BASELINES ON 14 DATASETS IN TERMS OF $pos_{in}(G)$ AND $neg_{out}(G)$, AND TIME IN SECONDS THE EMPTY CELL INDICATES THE METHOD FAILED TO RUN ON THE DATASET. _BEST RESULTS ARE UNDERSCORED. GRAPHIC FINDS A CLUSTERING THAT CONSISTENTLY ACHIEVES A GOOD COMPROMISE BETWEEN $pos_{in}(G)$ AND $neg_{out}(G)$ (POS AND NEG IN THIS TABLE AS PERCENTAGES) ON THE KONECT DATASET WITH **BOTH** HAVING HIGH VALUES UNLIKE OTHER BASELINES WHERE THEY FAIL OR SOLELY FOCUS ON EITHER EVALUATION METRIC.

	GraphC			**Hessian**			$\mathbf{A_{sym}}$			$\mathbf{L_{none}}$			$\mathbf{L_{sym}}$			$\mathbf{BNC_{none}}$			$\mathbf{BNC_{sym}}$			$\mathbf{SPO_{none}}$			$\mathbf{SPO_{sym}}$			**dns**			**sns**		
Dataset	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**	**pos**	**neg**	**t**
Highland	**93**	**100**	0.1	93	100	0.2	93	100	0.1	93	100	0.1	93	100	0.1	93	100	0.1	93	100	0.1	93	100	0.19	93	100	0.17	93	100	0.2	93	1	0.1
Sampson25	**70**	**90**	0.2	63	93	0.07	60	91	0.04	59	79	0.1	63	92	0.03	78	65	0.03	62	91	0.04	63	92	0.04	64	92	0.04	70	85	0.03	70	82	0.03
Congress	**93**	**100**	0.77	62	100	0.11	70	100	0.11	86	97	0.12	24	100	0.11	39	77	0.13	93	97	0.14	70	84	0.14	67	100	0.12	94	95	0.12	98	33	0.12
PPI	**81**	**98**	9	79	35	1	100	2	1	100	0.87	1	100	0.98	1	93	10	1	100	0.98	1	100	0.1	1	100	1	1	100	0.72	1	100	0.77	1
BitcoinAlpha	**81**	**86**	14	37	89	2	100	0.5	2	30	68	3	72	24	3	55	66	2	100	3	2	91	3	3	100	5	3	100	4	2	100	7	2
BitcoinOTC	**82**	**90**	22	30	94	4	71	42	4	21	80	7	61	43	4	86	61	4	100	35	4	62	30	8	100	28	4	100	8	4	100	8	4
Chess	**39**	**84**	45	40	70	7	100	0.17	7							65	34	7	100	0.21	7	100	0	12	100	0.51	8	100	0.2	7	100	0.2	7
TwitterRef	**89**	**100**	446	11	98	31	65	90	31				67	94	31	23	93	30	100	0.66	31	97	4	60	65	94	31	100	0.6	30	100	0.55	31
SlashdotZoo	**69**	**86**	716																														
Epinions	**80**	**88**	2094																														
WikiRfa	**61**	**81**	83	54	59	18	79	30	18	100	0.003	19	74	35	18	100	2	18	100	0.01	18	9	11	2	91	6	19	100	0.02	18	100	0.01	18
WikiElec	**70**	**79**	60	19	92	11	61	65	11	100	0.05	21	100	0.05	11	51	58	11	1	0.11	11	92	12	20	1	0.74	11	1	0.17	11	1	0.16	11
WikiConflict	**94**	**86**	1235																														
WikiPolitics	**82**	**85**	1279																														

TABLE IV
AMAZON RATINGS AND REVIEWS [36] MAPPED TO SIGNED GRAPHS. *GraphC* RESULTS INCLUDE $pos_{in}(G)$ AND $neg_{out}(G)$ (POS AND NEG IN THIS TABLE AS PERCENTAGES) AND THE RESULTING NUMBER OF CLUSTERS WITH MORE OR EQUAL TO 5 ELEMENTS AS WELL AS CLUSTERS WITH LESS THAN FIVE ELEMENTS.

Amazon	**LCC**		**GraphC**					
Ratings	**# vertices**	**# edges**	**pos**	**neg**	**# splits**	**time (s)**	**# clusters≥5**	**# clusters<5**
Books	9,973,735	22,268,630	73	86	22	115,561	31,988	1,003,734
Electronics	4,523,296	7,734,582	88	80	7	23,516	9,208	862,970
Jewelry	3,796,967	5,484,633	81	90	11	25,650	15,802	648,695
TV	2,236,744	4,573,784	74	87	17	19,019	4,456	281,653
Vinyl	1,959,693	3,684,143	74	87	16	16,420	5,718	169,493
Outdoors	2,147,848	3,075,419	92	80	15	13,613	12,294	393,579
AndrApp	1,373,018	2,631,009	77	89	24	1,642	1,462	205,336
Games	1,489,764	2,142,593 92	82	22	1,662	280,356	8,111	272,245
Automoto	950,831	1,239,450	94	79	13	783	8,515	202,749
Garden	735,815	939,679	93	85	13	436	4,636	153,452
Baby	559,040	892,231	79	91	14	489	2,367	94,474
Music	525,522	702,584	87	83	23	571	6,124	109,287
Video	433,702	572,834	85	93	14	364	1,401	46,481
Instruments	355,507	457,140	90	85	15	281	3,536	61,964
Reviews	**# vertices**	**# edges**	**pos**	**neg**	**# splits**	**time (s)**	**clusters≥5**	**# clusters<5**
Core Music	9,109	64,706	0.52	0.848	14	9.57	38	746
Core Video	6,815	37,126	0.572	0.852	13	5.54	26	737
Core Instrum	2,329	10,261	0.510	0.928	8	1.90	20	159

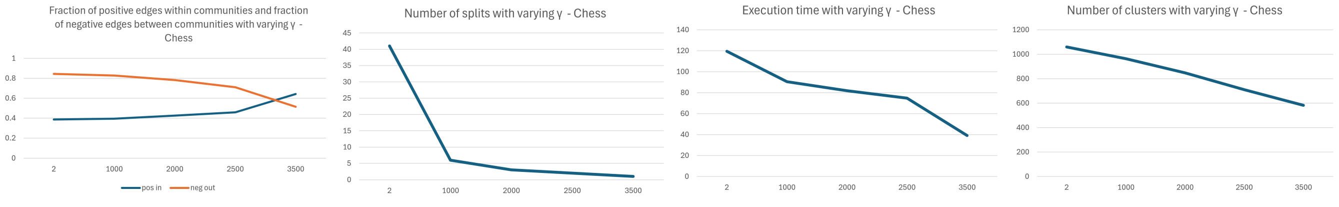

Fig. 7. The effect of varying $Gamma$ on the number of clusters, clustering quality, the execution time, and number of Harary splits on the Chess signed graph. The y-axis is pos_{in} and neg_{out} as in equations 7 and 8 respectively, number of splits, execution time, and number o communities for the four graphs respectively (from left to right). The x-axis is the $Gamma$ value. Any connected component of size less than $Gamma$ will be ignored by *GraphC*.

the clustering if split. This, however, saves up computation because the algorithm won't search for the best split for a connected component of size smaller than $Gamma$. Second, we run the algorithm with $I \in \{1000, 800, 500, 50, 10\}$ with the other parameters fixed as $Gamma = 2$, $\alpha = 0.5$, $\beta = 1$, $\epsilon = 0.00000001$, and $t_l = $ -1. According to the four graphs in Figure 8, decreasing the number of iterations for searching for the best Harary split for a connected component gradually degrades the overall quality of the clustering, reducing the execution time tremendously. On the other hand, the number of splits and clusters seems to stay the same or fluctuate around 41 splits and 1060 clusters. Finally, we run our algorithm

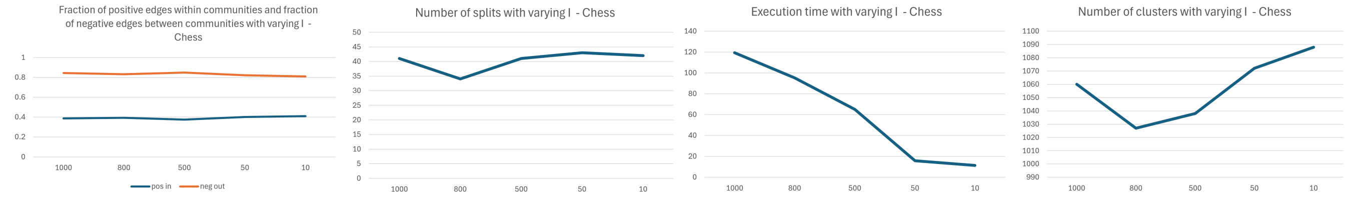

Fig. 8. The effect of varying I (number of iterations per connected component) on the number of clusters, clustering quality, execution time, and number of Harary splits on the Chess signed graph. The y-axis is pos_{in} and neg_{out} as in equations 7 and 8 respectively, number of splits, execution time, and number o communities for the four graphs, respectively (from left to right). The x-axis is the I value.

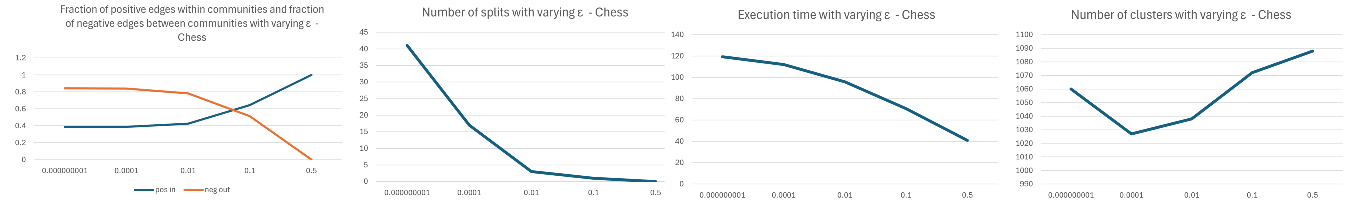

Fig. 9. The effect of varying ϵ on the number of clusters, clustering quality, execution time, and number of Harary splits on the Chess signed graph. From left to right, the y-axis is pos_{in} and neg_{out} as in equations 7 and 8 respectively. For this experiment, use only the number of clusters larger than 1 or 2 if for Amazon. (The effect of varying ϵ on the number of clusters), number of splits, execution time, and number of communities for the four graphs, respectively (from left to right).

with $\epsilon \in \{0.000000001, 0.0001, 0.01, 0.1, 0.5\}$ with the other parameters fixed as $Gamma = 2$, $\alpha = 0.5$, $\beta = 1$, $I = 1000$, and $t_l =$ -1. Increasing ϵ as observed by the last four graphs in Figure 9 discourages splits from taking place because there is a higher chance the overall improvement will be less than ϵ which prevents a cut and more connected components will be added to the set $processed$. This naturally degrades the quality of the clustering process and further reduces the time taken by the algorithm execution. The number of clusters seems to be almost the same as when we varied I, fluctuating around 1060. Overall, we can deduce that in order to obtain the best possible clustering, one should increase I and decrease $Gamma$ and ϵ as much as possible as long as the machine's resources can handle it regardless of the type, density, or size of the signed network.

IX. Conclusion

The arrangement of communities within a network can unveil concealed insights into the connections and dynamic processes within intricate systems. The identification of communities is a widely utilized approach across various domains, ranging from biological to social networks, to extract valuable information. Recent studies have highlighted the challenge of restoring community structure in sparse networks using spectral methods. Current issues in signed graph clustering algorithms are choosing an appropriate number of clusters k, eigenvalue pollution, no scalability to large graphs, and the reliance on ground truth to perform clustering. For this, we take a different approach and propose a novel and scalable hierarchical clustering algorithm for a signed network capable of automatically detecting optimal clusters without a predefined k using the GraphB+ algorithm. Our algorithm is also flexible in the sense that we can adjust the aforementioned optional parameters to achieve any desired outcome, such as controlling how many isolated vertices we want. Moreover, we foster scalability by introducing the $Gamma$ optional parameter that acts as a trade-off between efficiency and scalability. We evaluated our proposed algorithm as well as baselines for the pos_{in} and neg_{out} metrics. Experimental results show that our algorithm achieve high values for these two metrics on all datasets across the board. In contrast, the baseline meth ds fail to scale or obtain a clustering assignment close to optimal. More work is needed to parallelize our algorithm further to achieve a substantial speedup. Exploring how we can take advantage of the proposed consensus features [25] in order to improve the overall quality of the clustering assignment is also an exciting route we want to take.

References

[1] V. A. Traag and L. Šubelj, "Large network community detection by fast label propagation." *Scientific Reports*, vol. 13, no. 1, 2023.

[2] *Maria Tomasso*, L. Rusnak, and **Jelena Tešić**, "Cluster boosting and data discovery in social networks," in *Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing (SAC)*, 2022.

[3] M. Cucuringu, A. V. Singh, D. Sulem, and H. Tyagi, "Regularized spectral methods for clustering signed networks," *Journal of Machine Learning Research*, vol. 22, no. 264, pp. 1–79, 2021.

[4] *Maria Tomasso*, L. Rusnak, and **Jelena Tešić**, "Advances in scaling community discovery methods for signed graph networks," *Oxford Journal of Complex Networks*, vol. 10, no. 3, 06 2022.

[5] R. P. Abelson and M. J. Rosenberg, "Symbolic psycho-logic: A model of attitudinal cognition," *Behavioral Science*, vol. 3, no. 1, pp. 1–13, 1958.

[6] D. Cartwright and F. Harary, "Structural balance: a generalization of Heider's theory," *Psychological Rev.*, vol. 63, pp. 277–293, 1956.

[7] F. Harary and D. Cartwright, "On the coloring of signed graphs." *Elemente der Mathematik*, vol. 23, pp. 85–89, 1968. [Online]. Available: http://eudml.org/doc/140892

[8] T. Derr, Z. Wang, J. Dacon, and J. Tang, "Link and interaction polarity predictions in signed networks," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–14, 2020.

[9] K. Garimella, T. Smith, R. Weiss, and R. West, "Political polarization in online news consumption," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 15, 2021, pp. 152–162.

[10] R. Interian, R. G. Marzo, I. Mendoza, and C. C. Ribeiro, “Network polarization, filter bubbles, and echo chambers: An annotated review of measures, models, and case studies,” *arXiv preprint arXiv:2207.13799*, 2022.

[11] V. Amelkin and A. K. Singh, “Fighting opinion control in social networks via link recommendation,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 677–685.

[12] W. E. Donath and A. J. Hoffman, “Lower bounds for the partitioning of graphs,” *IBM Journal of Research and Development*, vol. 17, no. 5, pp. 420–425, 1973.

[13] M. O. et al., “Initial release of the signet package,” Sep. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1435036

[14] L. Boulton, “Spectral pollution and eigenvalue bounds,” *Applied Numerical Mathematics*, vol. 99, pp. 1–23, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168927415001270

[15] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon, “Scalable clustering of signed networks using balance normalized cut.” *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 615 – 624, 2012.

[16] M. Cucuringu, A. V. Singh, D. Sulem, and H. Tyagi, “Regularized spectral methods for clustering signed networks.” *Journal of Machine Learning Research*, vol. 22, pp. 1 – 79, 2021.

[17] A. Saade, F. Krzakala, and L. Zdeborová, “Spectral clustering of graphs with the bethe hessian,” *ArXiv*, vol. abs/1406.1880, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:919316

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

[19] P. Mercado, F. Tudisco, and M. Hein, “Spectral clustering of signed graphs via matrix power means,” *CoRR*, vol. abs/1905.06230, 2019. [Online]. Available: http://arxiv.org/abs/1905.06230

[20] Y.-C. Lee, N. Seo, K. Han, and S.-W. Kim, “Asine: Adversarial signed network embedding,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 609–618. [Online]. Available: https://doi.org/10.1145/3397271.3401079

[21] Y. Chen, T. Qian, H. Liu, and K. Sun, “”bridge”: Enhanced signed directed network embedding,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 773–782. [Online]. Available: https://doi.org/10.1145/3269206.3271738

[22] T. Derr, Y. Ma, and J. Tang, “Signed graph convolutional network,” *CoRR*, vol. abs/1808.06354, 2018. [Online]. Available: http://arxiv.org/abs/1808.06354

[23] C. Seo, K. Jeong, S. Lim, and W. Shin, “Siren: Sign-aware recommendation using graph neural networks,” *CoRR*, vol. abs/2108.08735, 2021. [Online]. Available: https://arxiv.org/abs/2108.08735

[24] M. Gholami, A. Sheikhahmadi, K. Khamforoosh, M. Jalili, and F. Veisi, “A new approach to finding overlapping community structure in signed networks based on Neutrosophic theory,” *Journal of Complex Networks*, vol. 12, no. 1, p. cnad051, 01 2024. [Online]. Available: https://doi.org/10.1093/comnet/cnad051

[25] L. Rusnak and **Jelena Tešić**, “Characterizing attitudinal network graphs through frustration cloud,” *Data Mining and Knowledge Discovery*, vol. 6, November 2021.

[26] *Ghadeer Alabandi*, **Jelena Tešić**, L. Rusnak, and M. Burtscher, “Discovering and balancing fundamental cycles in large signed graphs,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476153

[27] A. Marsden, “Eigenvalues of the laplacian and their relationship to the connectedness,” 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:17239810

[28] G. Chen, V. Liu, E. Robinson, L. J. Rusnak, and K. Wang, “A characterization of oriented hypergraphic laplacian and adjacency matrix coefficients,” *Linear Algebra and its Applications*, vol. 556, pp. 323 – 341, 2018.

[29] J. Kunegis, A. Lommatzsch, and C. Bauckhage, “The slashdot zoo : mining a social network with negative edges.” *Proceedings of the 18th international conference on World wide web*, pp. 741 – 750, 2009.

[30] M. Cucuringu, P. Davies, A. Glielmo, and H. Tyagi, *SigNet*, https://github.com/alan-turing-institute/SigNet, 2019.

[31] Q. Zheng and D. Skillicorn, “Spectral embedding of signed networks,” in *Proceedings of the 2015 SIAM International Conference on Data Mining (SDM)*. SIAM, 2015, pp. 55–63. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611974010.7

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] J. Kunegis, “KONECT – The Koblenz Network Collection,” in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW ’13. ACM, 2013, pp. 1343–1350. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488173

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

[35] M. Lai, V. Patti, G. Ruffo, and P. Rosso, “Stance evolution and twitter interactions in an italian political debate,” in *Natural Language Processing and Information Systems*, M. Silberztein, F. Atigui, E. Kornyshova, E. Métais, and F. Meziane, Eds. Cham: Springer International Publishing, 2018, pp. 15–27.

[36] R. He and J. McAuley, “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering,” in *Proceedings of the 25th International Conference on WWW*, 2016, pp. 507–517.

[37] C. Anderson, *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

Muhieddine Shebaro is a Ph.D. student at the Department of Computer Science at Texas State University. He received his B.S.c degree in Computer Science from Beirut Arab University, Lebanon, in 2021. His research is the intersection of network science, deep learning, and data science areas.

Martin Burtscher, Ph.D. is a Professor in the Department of Computer Science at Texas State University. He received a BS/MS degree from ET Zurich and a PhD degree from the University of Colorado in Boulder. Martin’s research focuses on the parallelization of graph algorithms and complex programs for GPUs and on the sy thesis of high-speed lossy and lossless data-compression algorithm . He has co-authored over 125 peer-reviewed scientific publications. Martin is an ACM distinguished member and an IEEE senior member.

Lucas Rusnak, Ph.D. is an Associate Professor in the Department of Mathematics at Texas State University. He received a PhD in Mathematics from Binghamton University in 2010 and has published over ten journal papers. Lucas’ research interests lie in the areas of network science, balanced theory, and oriented hypergraph analysis.

Jelena Tešić, Ph.D. is an Assistant Professor at Texas State University. She received her Ph.D. in 2004 and M.Sc. in 1999 in Electrical and Computer Engineering from the University of California Santa Barbara, USA, and EE Dipl. I e rned my Ing. degree in 1998 from the University of Belgrade, Serbia. Dr. Tešić ’s research focuses on large unstructured data analysis, and she has authored over 85 peer-reviewed scientific papers and holds six US patents. Jelena is an IEEE senior member and has served as an IEEE Multimedia Magazine Guest Editor.