Information Sciences 760 (2027) 124130

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

GraphC: Hierarchical clustering of signed graph networks

Muhieddine Shebaro[a,*], Lucas Rusnak[b], Martin Burtscher[a], Jelena Tešić[a]

[a] Department of Computer Science, Texas State University, 601 University Drive, San Marcos, TX 78666, USA
[b] Department of Mathematics, Texas State University, 601 University Drive, San Marcos, TX 78666, USA

HIGHLIGHTS

- graphC is a k-independent hierarchical clustering method for signed graphs.
- It scales to large sparse signed networks with tens of millions of edges.
- It avoids spectral decomposition and eigenvalue pollution issues without requiring training or hyperparameter tuning.
- It treats positive and negative edges symmetrically.
- It outperforms the second-best algorithm across 14 datasets by 18.6% on average.

ARTICLE INFO

Keywords:
Signed network
Community detection
Balance theory
Balanced states

ABSTRACT

When applied to signed graphs, spectral clustering methods often struggle to capture groupings accurately. Recent research shows that these methods become less effective as the size of the signed network increases. To address this problem, we present a new, scalable, hierarchical graph-clustering algorithm, *graphC*, for identifying the optimal clusters in networks of varying sizes. *graphC* maintains a high proportion of positive edges within each cluster while maximizing the negative edges between clusters. One key feature of *graphC* is that it does not require the number of clusters (k) to be predefined. We validate the performance of *graphC* by comparing it to ten other signed-graph clustering algorithms on fourteen datasets. We show that *graphC* scales to large signed graphs from Amazon, which include tens of millions of vertices and edges. On average, *graphC* outperforms the second-best algorithm by 18.6% in terms of maximizing the positive edge density within clusters and the negative edge density between clusters. The comparison excludes cases where the third-party algorithms fail to complete their tasks.

1. Introduction

Communities within a network are sets of vertices characterized by denser interconnections than with the broader network. These communities may diverge, and vertices may belong to multiple communities. Detecting community structure is important because it provides insights into relationships and dynamic mechanisms within complex networks spanning diverse domains, from biological to social networks. The inherent complexity often produces datasets exceeding computational capacity, necessitating distributed or high-performance computing and requiring data partitioning for processing. The efficiency of such partitioning schemes in community-detection algorithms depends on various factors, including graph topology, sparsity, and the objective function used to assess clustering quality. While state-of-the-art community-discovery algorithms in unsigned graphs adeptly handle vast networks

* Corresponding author.
Email address: shebaro.m@gmail.com (M. Shebaro).

https://doi.org/10.1016/j.ins.2026.124130
Received 9 July 2025; Received in revised form 6 September 2026; Accepted 6 September 2026
Available online 8 September 2026

comprising millions of vertices and edges [1], the modeling of unsigned graphs falls short in capturing complex relationships driven by both positive and negative interactions. Signed graphs explicitly encode antagonistic or distrustful relations and provide more meaningful representations of data in domains such as recommender systems, social media, trust-distrust platforms, and biological networks [2]. In recommendation systems, for example, high ratings, likes, and endorsements naturally induce positive edges, while low ratings, dislikes, and complaints induce negative edges between users and items. In trust–distrust networks, explicit trust edges contrast with distrust or blocking edges. In protein–protein interaction networks, activating and inhibiting relations can also be modeled as positive and negative edges, respectively. In all of these cases, the ability to recover communities that simultaneously respect positive and negative structures is crucial for downstream tasks such as recommendation, anomaly detection, and functional module discovery.

Despite this modeling power, community discovery methods for signed graphs struggle to recover communities in graphs with merely thousands of vertices and hundreds of thousands of edges [2]. The latest research underscores the challenges that spectral methods pose for the recovery of community structure in sparse networks, even when normalization techniques are incorporated [3]. This finding aligns with our observations: spectral methods degrade significantly when clustering large, sparse, signed networks derived from real-world data sources [4]. In particular, signed Laplacians can suffer from eigenvalue pollution on large, ill-conditioned matrices, and the geometric interpretation of eigenvectors becomes increasingly unstable as graph size grows. These issues make it difficult to use conventional spectral clustering as a reliable backbone for large-scale signed community detection. Balance theory provides a complementary perspective that focuses on the sign structure of cycles rather than only on the spectrum of a Laplacian. Originally proposed by Heider [5] and later formalized by Harary [6], balance theory explains the evolution of attitudes and alliances in networks. It characterizes structurally "balanced" signed graphs in which all cycles are positive. Harary's work on k-way balance, Harary bipartitions, and the frustration index has led to a rich theoretical framework that underpins many modern algorithms for signed networks, including methods for link sign prediction [7], content recommendation, and anomaly detection. However, existing balance-based clustering approaches often either require strong structural assumptions or do not scale gracefully to massive graphs.

A key limitation of many state-of-the-art signed graph clustering algorithms is that the optimal number of communities k must be specified before the algorithm is run. Although widely adopted, the elbow method is unreliable in practice [8]. Eigenvector pollution, network sparsity, average degree, and overall structure can influence the choice of k in real large graphs [4]. If ground truth is unavailable, the adopted metrics may not accurately reflect the quality of communities discovered by the methods [4]. Thus, techniques that do not require a predefined k are preferable, especially in realistic settings where the number of communities is unknown and may change over time. Beyond this algorithmic convenience, a k-independent approach is particularly attractive in applications like recommendation, trust–distrust networks, and computational biology, where the "true" number of meaningful groups is inherently data-dependent and domain-specific.

In this paper, we introduce *graphC*, a hierarchical *graphC*lustering algorithm for signed graphs that is explicitly designed to address these challenges in both theory and practice. The algorithm is built on a balanced–spectral duality result that connects Harary cuts of balanced states to eigenvectors associated with zero eigenvalues of the corresponding Laplacian. Intuitively, once a signed graph is transformed into a balanced state via sign switching, its Laplacian becomes conjugate to the Laplacian of an all-positive graph with the same topology. In a connected balanced graph, the 0-eigenvector has entries in $\{\pm 1\}$, and its sign pattern directly encodes a Harary bipartition. Thus, spectral clustering on balanced states effectively reconstructs Harary cuts and motivates searching for high-quality Harary splits without explicitly computing a full spectral decomposition. At the same time, our analysis highlights why standard spectral methods become less informative on large, unbalanced signed graphs, strengthening the case for an explicitly balance-driven algorithm.

To measure clustering quality in real signed networks, *graphC* employs a loss function that symmetrically accounts for both positive and negative edges. In many empirical datasets, the number of positive edges is 5–10 times larger than the number of negative edges, as shown in [4]. The imbalance in the number of edges leads classical metrics, such as the unhappy ratio, to favor trivial solutions that place all vertices in a single community. Our formulation instead minimizes the fraction of violating positive edges between communities and violating negative edges within communities, while also controlling the fraction of isolated vertices. The conductance-like objective is normalized by the total number of positive and negative edges, thereby mitigating bias arising from highly skewed sign distributions. As a result, *graphC* produces clusters that more faithfully reflect the underlying signed structure in recommendation graphs, trust–distrust networks, and protein–protein interaction datasets.

From a scalability perspective, *graphC* is designed to operate on large, sparse signed graphs with millions of vertices and edges. Rather than performing a global spectral decomposition, the algorithm iteratively identifies promising connected components, computes candidate Harary cuts using a stable-state search, and accepts splits only when they sufficiently improve the global loss. The hierarchical refinement is accompanied by stopping criteria and size thresholds that prevent over-partitioning components that offer diminishing returns. We analyze the complexity of the main loop in terms of the number of committed and reverted splits and provide empirical evidence that the algorithm scales to real-world graphs at Amazon scale, processing nearly ten million vertices and over twenty million edges within practical time and memory budgets.

Finally, *graphC* is motivated by and evaluated on real applications. In recommendation systems, it groups users with similar preferences and products that are consistently liked or disliked within communities, enabling more personalized and robust recommendations. In trust—distrust networks such as Bitcoin OTC, it isolates malicious or low-trust actors while identifying cohesive, trusted communities, thereby aiding fraud detection and risk management. In computational biology, *graphC* discovers clusters of proteins that function coordinately through activation or inhibition relationships, enabling the discovery of functional modules without

requiring biologists to pre-specify the number of clusters. These application scenarios illustrate that the main contributions of *graphC* are not only algorithmic but also directly beneficial in practical, large-scale signed-network analysis.

2. Related work

Existing approaches to community detection in signed networks span classical spectral methods, balance-theoretic formulations, probabilistic models, and more recent deep learning techniques. Early work on spectral clustering for signed graphs relies on Laplacian variants that extend unsigned formulations to handle positive and negative edges [9]. For example, the signed Laplacian and its symmetric normalization adapt the usual quadratic form to penalize cuts that separate positively connected vertices or merge negatively connected ones. Balanced normalized cut (BNC) and its symmetric variant explicitly incorporate signed weights into the normalized-cut objective [10]. At the same time, SPONGE and related generalized eigenproblem formulations encode positive and negative contributions into a generalized Rayleigh quotient. Bethe-Hessian-based approaches use signed variants of the Hessian to capture assortative and disassortative structure in a spectral framework [11]. These methods have solid theoretical foundations but typically require specifying the number of clusters k and computing several extremal eigenvectors of large sparse matrices, which can be computationally demanding and susceptible to eigenvalue pollution in real large-scale signed networks [4].

Beyond these Laplacian-based methods, a line of work focuses more directly on balance theory and the structure of signed cycles. Harary's classical results on balance, frustration index, and Harary bipartitions [12] inspired algorithms that search for large balanced subgraphs or iteratively trim vertices and edges to improve balance. TIMBAL, for instance, addresses the problem of finding large balanced subgraphs by proposing a two-stage trimming-and-restoring procedure that maximizes balance while maintaining connectivity in the remaining graph[13]. Other heuristics remove edges associated with the smallest Laplacian eigenvalues until a maximum balanced subgraph is obtained [14]. While such methods closely align with the combinatorial notion of balance, they are often designed to extract a single large balanced component rather than to produce hierarchical community structure on massive graphs, and they may still rely on spectral primitives that inherit the aforementioned scalability limitations.

Recent work has also explored local spectral formulations tailored to signed graphs. Xiao et al. introduce a local spectral method for detecting polarized communities, modeling the search for locally biased cuts as a signed eigenproblem centered around a seed set [15]. Tzeng et al. formulate conflicting-group discovery as a maximum discrete Rayleigh quotient (MAX-DRQ) problem and develop two spectral approximation methods with theoretical guarantees [16]. These approaches show that spectral ideas can be adapted to capture polarization and conflict in a principled way. Still, they typically require selecting seeds or solving parameterized eigenproblems and are not primarily designed to generate a k-independent hierarchical clustering of the entire signed network.

A different family of methods leverages graph neural networks (GNNs) and learned embeddings. He et al. combine a signed mixed-path aggregation (SIMPA) scheme with a probabilistic balanced normalized cut loss to train a model that predicts community assignments in signed graphs [17]. Their approach uses triangle-based social heuristics (e.g., "the friend of my friend is my friend") and depends on labels produced by SPONGE for some graphs, thereby coupling its performance to that of spectral baselines. Other works propose Signed Graph Neural Networks (SGNNs) that incorporate balance theory into message passing and loss design [18]. These models initially showed promising results, but their embeddings often misrepresent negative edges in highly imbalanced settings, posing challenges for downstream community discovery and recommendation tasks [19]. Neutrosophic signed graph convolutional networks further extend this line of work by mapping nodes to a neutrosophic feature space, enabling the identification of overlapping communities [20]. While these neural approaches are flexible and can exploit side information such as node features and labels, they typically require training data, careful hyperparameter tuning, and mini-batch optimization, and they do not naturally yield interpretable Harary cuts or explicit balance-based guarantees.

Complementary to these embedding-based methods, matrix-function and augmentation techniques have been proposed to better integrate positive and negative information. Mercado et al. use the signed matrix power means to merge the positive and negative Laplacian components into a unified operator whose eigenvectors drive clustering[21], and the design directly addresses the asymmetry between positive and negative edges at the operator level. Zhang et al. also applied contrastive learning to signed bipartite graphs, thereby improving robustness in representation learning for applications such as social networks, recommendation systems, and peer-review systems [22]. These works strengthen the empirical case for carefully modeling signed structure but largely focus on predictive tasks (e.g., link sign prediction) rather than on unsupervised, k-independent hierarchical clustering.

Several methods explicitly target overlapping or multi-resolution community structure in signed networks. The neutrosophic SGCN mentioned above [20] is one example of a network in which overlapping communities arise naturally from embeddings in a high-dimensional feature space. Other approaches adapt local spectral ideas to extract nested or multi-scale polarized groups [15]. Classical multilevel and hierarchical clustering algorithms for unsigned graphs usually proceed by coarsening the graph, computing a partition at a coarse level, and uncoarsening with local refinements. In contrast, *graphC* operates directly on the original signed graph without an explicit coarsening/uncoarsening pipeline: it repeatedly searches for Harary cuts that reduce a signed loss function and refines connected components in a top-down manner. The distinction is important because *graphC*'s hierarchy emerges from the balance-driven splitting process itself, rather than from an externally imposed multilevel structure. From a dynamical perspective, Altafini analyzes spreading processes on signed networks and their connection to structural balance, showing how sign patterns shape long-term behavior [23]. This line of work underscores the importance of balance-aware modeling in signed networks; *graphC* leverages related balance-theoretic ideas, but focuses on static hierarchical community structure rather than dynamical spreading.

Beyond single-view signed clustering, Liu et al. propose a multiview clustering framework that partitions a signed prototype graph constructed from heterogeneous views of the data [24]. Their method integrates signed relationships across views into a unified prototype graph and then performs clustering on this signed representation. While their focus is on multiview data and embedding-based

clustering, rather than on k-independent hierarchical structure, the work further highlights the usefulness of explicitly modeling signed patterns when defining clustering objectives. Our focus in this paper is on unsupervised, k-independent signed community detection. Still, the balance-theoretic principles and hierarchical design of *graphC* are complementary to these recent graph-attention models and could potentially be combined with them in future work. More recently, Zhang et al. introduce a desired-clustering formulation for signed networks that explicitly encodes target intra- and inter-community relations and then optimizes over cluster assignments to match these desiderata [25]. Their formulation provides a flexible optimization-based view of signed clustering but still assumes a prescribed number of communities and requires solving a global objective, in contrast to *graphC*'s hierarchical Harary-split process, which discovers the number of clusters adaptively. Very recent work at NeurIPS further explores signed graph neural architectures, including signed-graph perspectives on over-smoothing [26] and local search methods for polarized community detection [27]. These methods focus primarily on learned representations and prediction tasks. In contrast, *graphC* remains a purely unsupervised, k-independent clustering algorithm that operates directly on the signed graph structure without training a neural network.

In summary, existing signed graph clustering methods either (i) rely heavily on global spectral decompositions and require a predefined number of clusters k [3], (ii) focus on extracting large balanced subgraphs or local polarized groups without producing a full hierarchical clustering [14], or (iii) depend on supervised or semi-supervised neural architectures that introduce additional modeling and training complexity [17]. The proposed *graphC* algorithm occupies a different point in this design space: it is unsupervised, does not require k as input, avoids global spectral decomposition on the original signed graph, and uses a balance-theoretic loss that treats positive and negative edges symmetrically. At the same time, it is designed to scale to large signed networks encountered in real applications, where both the number of vertices and edges can reach tens of millions.

3. Balanced-spectral duality in signed networks

In this section, we present the terminology and properties of signed graphs and introduce spectral analysis and the duality theory of near-balanced states. Let Σ_i be a signed graph with underlying unsigned topology $G = (V, E)$, as illustrated in Fig. 1. Let Σ_{ij} denote the collection of vertex sets discovered by clustering algorithm j when applied to Σ_i. The *graphC* algorithm minimizes the fraction of positive edges *between* communities and the fraction of negative edges *within* communities when algorithm j is applied to the signed graph Σ_i. We quantify the positive edges outside communities as $pos_{out}(\Sigma_{ij})$ and the negative edges within communities as $neg_{in}(\Sigma_{ij})$ for algorithm j operating on the signed graph Σ_i.

Intuitively, balance theory and spectral theory are linked as follows. If a signed graph can be transformed, via a sequence of vertex switches, into an all-positive graph with the same underlying topology, then its Laplacian matrix is conjugate to the Laplacian of that all-positive graph. In a connected balanced graph, the 0-eigenvector of the Laplacian has entries in $\{\pm 1\}$, and its sign pattern exactly corresponds to a Harary bipartition. Harary bipartition provides an explicit spectral fingerprint of balance. It motivates the *graphC* strategy of searching for high-quality Harary cuts without computing a full spectral decomposition on the original, possibly unbalanced, signed graph.

3.1. Fundamental cycle basis

Definition 3.1. Let $G = (V, E)$ be a graph. A graph $G_i = (V_i, E_i)$ is a **subgraph** of G if $V_i \subseteq V$ and $E_i \subseteq E$, i.e., all vertices and edges of G_i are contained in G.

Definition 3.2. Let $G = (V, E)$ be a graph. A **path** of length m is a sequence of distinct vertices $v_0, v_1, \ldots, v_m$ such that each consecutive pair $\{v_{k-1}, v_k\}$ is an edge in E. A **connected graph** is a graph in which, for every pair of vertices $u, v \in V$, there exists a path between

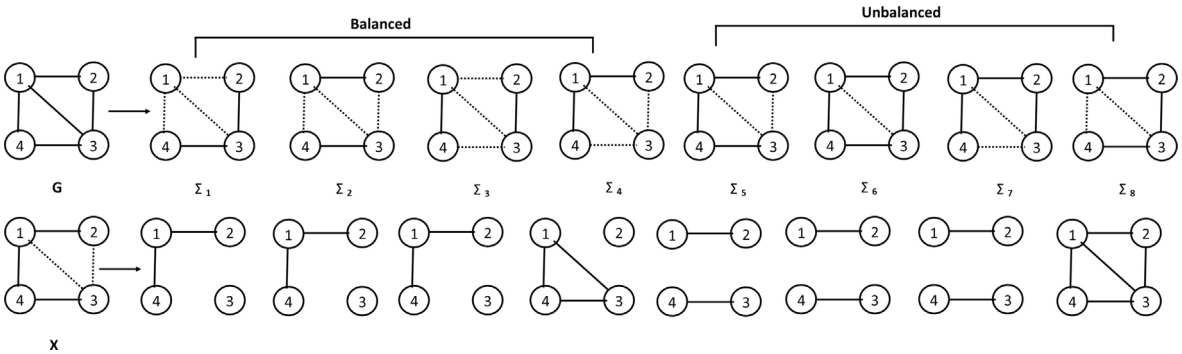

Fig. 1. Dotted lines are negative edges whereas **solid** lines are positive edges. Top: A graph G, a balanced graph Σ_1 obtained by switching (changing the sign of the edges originating from) v_1, a balanced graph Σ_2 obtained by switching v_1 and v_2, a balanced graph Σ_3 by switching v_2 and v_3, and a balanced graph Σ_4 by switching v_4. Σ_5, Σ_6, Σ_7 and Σ_8 are examples of unbalanced states. Bottom: Harary cuts of nearest stable states [28] of Σ_5.

u and v. A **cycle** is a path $v_0, v_1, \ldots, v_m$ with $m \geq 3$ that begins and ends at the same vertex, i.e., $v_0 = v_m$, and has all intermediate vertices distinct. A **cycle basis** of G is a set of simple cycles whose edge sets form a basis of the cycle space of G over $\mathbb{Z}_2$.

Definition 3.3. Let $G = (V, E)$ be a graph and let $T = (V, E_T)$ be a spanning tree of G, so that $E_T \subseteq E$ and $|E_T| = |V| - 1$. Let $e = \{x, y\} \in E \setminus E_T$ be an edge that is *not* in T, with endpoints $x, y \in V$. The **fundamental cycle** determined by e (with respect to T) is the unique simple cycle consisting of the edge e together with the unique path in T between x and y.

Corollary 3.1. *Let $G = (V, E)$ be a connected graph and let T be any spanning tree of G. For each edge $e \in E \setminus E_T$, the corresponding fundamental cycle with respect to T is well defined and simple. The collection of all such fundamental cycles, as e ranges over $E \setminus E_T$, forms a cycle basis of G. Moreover, since $|E_T| = |V| - 1$, the number of fundamental cycles is exactly $|E| - |V| + 1$.*

Proof. Because T is a spanning tree of G, there is a unique simple path in T between any two vertices $x, y \in V$. For each $e = \{x, y\} \in E \setminus E_T$, adding e to T creates exactly one simple cycle: the union of e with the unique path in T between x and y. The cycle is well-defined and contains no repeated vertices except at the endpoints, where $x = y$, so it is simple.

It is a standard result in algebraic graph theory that the dimension of the cycle space of a connected graph G is $|E| - |V| + 1$. Since $|E_T| = |V| - 1$, we have $|E \setminus E_T| = |E| - |V| + 1$, so the number of fundamental cycles is equal to the dimension of the cycle space. Each fundamental cycle contributes an edge $e \in E \setminus E_T$ that does not appear in any other fundamental cycle, so the set of fundamental cycles is linearly independent over $\mathbb{Z}_2$. Because the number of independent cycles matches the dimension of the cycle space, this set of cycles forms a basis. □

3.2. Balanced signed graphs

Definition 3.4. A **signed graph** is a pair $\Sigma = (G, \sigma)$ where $G = (V, E)$ is an underlying unsigned graph and $\sigma : E \to \{+1, -1\}$ is an edge-signing function. An edge $e \in E$ is *positive* if $\sigma(e) = +1$ and *negative* if $\sigma(e) = -1$; we denote these by e^+ and e^-, respectively. For any subgraph $H = (V_H, E_H)$ of G, the **sign** of H is defined as the product of the signs of its edges, i.e., $\prod_{e \in E_H} \sigma(e)$. A **balanced signed graph** is a signed graph in which every cycle has a positive sign. The **frustration** of a signed graph Σ is the minimum number of edges whose signs must be switched to obtain a balanced signed graph on the same underlying graph G. Fig. 1 (top, balanced states) provides an example.

Theorem 3.1 (**[12]**). *Let $\Sigma = (G, \sigma)$ be a signed subgraph. The following are equivalent:*

1. *Σ is balanced (every cycle has a positive sign).*
2. *For every vertex pair (v_i, v_j) in Σ, all (v_i, v_j)-paths have the same sign.*
3. *$Fr(\Sigma) = 0$, i.e., the frustration of Σ is zero.*
4. *The vertex set V can be partitioned into two (possibly empty) sets U and W such that all edges with both endpoints in U or both endpoints in W are positive, and all edges with one endpoint in U and the other in W are negative. The pair (U, W) is called a Harary bipartition.*

A **Harary cut** (or **Harary split**) is formed by deleting the negative edges in each balanced state in the frustration cloud of the signed graph, as illustrated in Fig. 1 (bottom). In that example, a trivially unbalanced signed graph consisting of four vertices and five edges yields eight nearest balanced states. A Harary cut can yield multiple connected components, and the input graph may contain many initially connected components, in which case Harary cuts and balancing are carried out independently on each component. In prior work, we have characterized signed graphs via the frustration cloud [28], a collection of nearest-balanced states, and introduced an improved, parallelizable method for efficiently discovering the nearest-balanced states in large signed graphs [29]. In this paper, the terms **Harary cut** and **Harary split** are used interchangeably to refer to the deletion of negative edges in a balanced state. Table 1 summarizes the notations and their meanings.

3.3. Spectral clustering and balanced states

A signed graph is *balanced* if the graph contains no cycle with an odd number of negative edges. The *switch* operation flips the sign of all edges incident to a specific vertex, as shown in Fig. 1 when switching v_1 in G to obtain Σ_1. Two signed graphs on the same underlying graph G are said to be *switching equivalent* if one can be transformed into the other by a sequence of such vertex switches. In particular, all balanced signed graphs obtained from an all-positive graph on G by switching are switching-equivalent [30].

The balanced signed graphs of the same underlying graph G in Fig. 1 (top) are switching equivalent. For example, Σ_2 and Σ_3 in Fig. 1 (top) are switching equivalent because we can obtain Σ_3 by switching v_1 and v_3 in Σ_2. The key point is that switching operations preserve the graph's balance, as they do not alter the sign of any cycle: every time we switch a vertex, exactly two edges on any given cycle that are incident to that vertex change sign, leaving the product of edge signs on the cycle unchanged [28]. Consequently, the set of balanced states associated with a given underlying graph forms a switching-equivalence class.

Table 2 illustrates how switching affects the Laplacian spectrum. For the connected graph G on four vertices with all positive edges, the Laplacian has a simple 0-eigenvalue with eigenvector $\mathbf{1} = (1, 1, 1, 1)^\top$. The balanced signed graphs Σ_0, Σ_1, and Σ_2 are obtained from G by switching different subsets of vertices, and they are all isospectral: their Laplacians share the same eigenvalues, while their eigenvectors differ only by sign flips in the entries corresponding to switched vertices. In contrast, the unbalanced graph Σ_5 has a different spectrum and different eigenvectors.

Table 1
Summary of notations and their meanings. The terms harary cut and harary split are used interchangeably to refer to the deletion of negative edges in a balanced state.

Notation	Meaning
Σ/G	signed network/its underlying graph.
Σ_{ij}	collection of vertex sets (connected components) discovered by clustering algorithm j for signed network Σ_i.
T/I	spanning tree/number of spanning trees or iterations.
$v/V_x/V/\|V\|$	a node/set of some vertices/set of all vertices/number of nodes.
$E_x/E/\|E\|$	a set of edges/set of all edges in Σ / number of edges.
$e/e^+/e^-$	an edge/positive edge/negative edge.
$Fr(\Sigma)$	frustration index of Σ or number of edge sign switches needed to balance Σ.
(U, W)	Harary bipartition sets, $\|U\| \geq \|W\|$.
neg_{in}/neg_{out}	fraction of negative edges within communities/fraction of negative edges between communities.
pos_{in}/pos_{out}	fraction of positive edges within communities/fraction of positive edges between communities.
CC	a connected component.
C	a clustering assignment.
α	parameter to control the weights between pos_{out} and neg_{in}.
β	parameter to control the weight of the fraction of isolated vertices.
G^F	filtered signed graph obtained by removing certain nodes.
G^{FC}	filtered signed graph obtained by removing certain nodes, followed by applying the best Harary cut.
Γ	threshold parameter indicating the minimum component size to be processed by the algorithm; components smaller than Γ are excluded.
ϵ	lower bound on the improvement needed for a Harary split to be committed and applied permanently during the algorithm's run.
GraphBplus	graphBplus algorithm introduced in [29]; returns the best-balanced signed graph determined by the corresponding Harary cut.

Table 2
Balanced signed graphs Σ_0, Σ_1, and Σ_2 are isospectral: they have identical eigenvalues, but their eigenvectors differ by sign patterns. Σ_5 is an unbalanced graph, so its eigenvalues and eigenvectors differ. We present the eigen-decompositions of Σ_0, Σ_1, and Σ_2 to illustrate the differences between balanced and unbalanced graphs.

	Balanced Graphs												Unbalanced Graph			
Graph	Σ_0				Σ_1				Σ_2				Σ_5			
Eigen values	0	2	4	4	0	2	4	4	0	2	4	4	$2-\sqrt{2}$	$3-\sqrt{3}$	$2+\sqrt{2}$	$3+\sqrt{3}$
	1	0	1	0	1	0	1	0	1	1	1	1	1	1	-1	1
Eigen	1	1	0	1	-1	1	0	1	1	1	0	1	$\sqrt{2}$	0	$\sqrt{2}$	0
vectors	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	$\frac{1-\sqrt{3}}{2}$

Classical spectral clustering yields as many zero eigenvalues as there are connected components in the graph [30]. In a connected unsigned graph G, the 0-eigenspace is one-dimensional and spanned by the all-ones vector $\mathbf{1}$; its entries do not distinguish communities. For balanced signed graphs obtained by switching from G, the Laplacian eigenvalues remain the same, but the corresponding eigenvectors can acquire entries in $\{\pm 1\}$ that reflect the switched vertex set. This observation underlies the balanced–spectral duality result described below.

We now make the connection between switching and the Laplacian explicit. Let $G = (V, E)$ be a connected graph and let $\Sigma = (G, \sigma)$ be a balanced signed graph obtained from the all-positive signing by switching a subset $W \subseteq V$ of vertices. Define the diagonal matrix $\mathbf{I}_W \in \mathbb{R}^{|V|\times|V|}$ by

$$(\mathbf{I}_W)_{vv} = \begin{cases} -1, & v \in W, \\ +1, & v \notin W. \end{cases}$$

Let $\mathbf{L}_G$ denote the (combinatorial) Laplacian of the all-positive graph G, and let $\mathbf{L}_\Sigma$ denote the Laplacian of Σ. Then, as argued earlier, we have

$$\mathbf{L}_\Sigma = \mathbf{I}_W \, \mathbf{L}_G \, \mathbf{I}_W.$$

We next prove that all balanced signed graphs on the same underlying graph are isospectral and, in the connected case, that the 0-eigenvector encodes a Harary cut.

Theorem 3.2. *The Laplacian matrices of balanced signed graphs with the same underlying graph G are isospectral.*

Proof. Let $G = (V, E)$ be a fixed underlying graph and let $\Sigma_1 = (G, \sigma_1)$ and $\Sigma_2 = (G, \sigma_2)$ be two balanced signed graphs on G. Because both Σ_1 and Σ_2 are balanced, each can be obtained from the all-positive signing by a sequence of vertex switches. Therefore, there exist subsets $W_1, W_2 \subseteq V$ and diagonal matrices $\mathbf{I}_{W_1}, \mathbf{I}_{W_2}$ with entries in $\{\pm 1\}$ such that

$$\mathbf{L}_{\Sigma_1} = \mathbf{I}_{W_1} \, \mathbf{L}_G \, \mathbf{I}_{W_1}, \qquad \mathbf{L}_{\Sigma_2} = \mathbf{I}_{W_2} \, \mathbf{L}_G \, \mathbf{I}_{W_2}.$$

Consider the matrix $\mathbf{S} = \mathbf{I}_{W_2}\mathbf{I}_{W_1}$. Since both $\mathbf{I}_{W_1}$ and $\mathbf{I}_{W_2}$ are diagonal with entries ± 1, their product $\mathbf{S}$ is also diagonal with entries ± 1 and satisfies $\mathbf{S}^{-1} = \mathbf{S}$. We compute

$$\mathbf{S}^{-1}\,\mathbf{L}_{\Sigma_1}\,\mathbf{S} = \mathbf{I}_{W_1}\mathbf{I}_{W_2}\,\mathbf{I}_{W_1}\,\mathbf{L}_G\,\mathbf{I}_{W_1}\,\mathbf{I}_{W_2}\mathbf{I}_{W_1} = \mathbf{I}_{W_2}\,\mathbf{L}_G\,\mathbf{I}_{W_2} = \mathbf{L}_{\Sigma_2},$$

where we used $\mathbf{I}_{W_k}^2 = \mathbf{I}$ and commutativity of diagonal matrices. Thus, $\mathbf{L}_{\Sigma_2}$ is similar to $\mathbf{L}_{\Sigma_1}$ via the invertible matrix $\mathbf{S}$. Similar matrices have identical characteristic polynomials and hence the same eigenvalues. Therefore, any two balanced signed graphs on the same underlying graph G have isospectral Laplacian matrices.

An alternative viewpoint, is to express the Laplacian characteristic polynomial in terms of contributions from cycle subgraphs. In that formulation, the signs of cycles determine the coefficients of the characteristic polynomial. Because a balanced signed graph has all cycles positive, every such graph on G produces the same signed cycle structure and hence the same characteristic polynomial. Thus, all balanced signed graphs on G have identical Laplacian spectra. □

Theorem 3.3. *Let $\Sigma = (G, \sigma)$ be a connected balanced signed graph, and let $\mathbf{L}_\Sigma$ denote its Laplacian. Then:*

1. *The dimension of the 0-eigenspace of $\mathbf{L}_\Sigma$ is 1.*
2. *There exists a 0-eigenvector $\mathbf{z}$ whose entries are all in $\{+1, -1\}$.*
3. *Partitioning the vertex set V according to the sign of $\mathbf{z}_v$ yields a Harary bipartition of Σ.*

Proof. Since Σ is balanced, it is switching equivalent to the all-positive graph on G. Let $W \subseteq V$ and $\mathbf{I}_W$ be as above, and let $\mathbf{L}_G$ be the Laplacian of the connected all-positive graph G. Then $\mathbf{L}_\Sigma = \mathbf{I}_W\mathbf{L}_G\mathbf{I}_W$.

Because G is connected, it is well known that $\mathbf{L}_G$ has a simple 0-eigenvalue, and its 0-eigenspace is spanned by the all-ones vector $\mathbf{1}$ [30]. In particular, $\mathbf{L}_G\mathbf{1} = \mathbf{0}$ and any 0-eigenvector is a scalar multiple of $\mathbf{1}$. Consider the vector

$$\mathbf{z} = \mathbf{I}_W\mathbf{1}.$$

Since $\mathbf{I}_W$ has diagonal entries ± 1, the entries of $\mathbf{z}$ are all in $\{+1, -1\}$. We now check that $\mathbf{z}$ is a 0-eigenvector of $\mathbf{L}_\Sigma$:

$$\mathbf{L}_\Sigma\mathbf{z} = \mathbf{I}_W\mathbf{L}_G\mathbf{I}_W\mathbf{I}_W\mathbf{1} = \mathbf{I}_W\mathbf{L}_G\mathbf{1} = \mathbf{I}_W\mathbf{0} = \mathbf{0}.$$

Thus, $\mathbf{z}$ lies in the kernel of $\mathbf{L}_\Sigma$. Because similarity preserves the dimension of eigenspaces, the multiplicity of the 0-eigenvalue of $\mathbf{L}_\Sigma$ is the same as that of $\mathbf{L}_G$, namely 1. Therefore, the 0-eigenspace of $\mathbf{L}_\Sigma$ is one-dimensional and spanned by $\mathbf{z}$, proving items (1) and (2).

To prove item (3), define

$$U = \{v \in V : z_v = +1\}, \qquad W' = \{v \in V : z_v = -1\}.$$

By construction, switching all vertices in W transforms Σ into the all-positive graph on G, and this same set W is encoded in the sign pattern of $\mathbf{z}$. In the all-positive graph, every edge connects two vertices with the same sign in $\mathbf{1}$, and there is no distinction between U and W'. In the original signed graph Σ, however, edges whose endpoints have the same sign in $\mathbf{z}$ correspond to positive edges after switching, and edges whose endpoints have opposite signs correspond to negative edges after switching. Because balance is preserved by switching and all cycles in Σ are positive, it follows that all edges inside U and inside W' are positive in Σ, and all edges between U and W' are negative. Hence (U, W') is a Harary bipartition of Σ in the sense of Theorem 3.1. The partition induced by the sign pattern of the 0-eigenvector $\mathbf{z}$ is therefore exactly a Harary cut of the balanced signed graph Σ. □

The previous theorem shows that, in a connected balanced signed graph, spectral clustering based on the 0-eigenvector recovers a Harary bipartition. In contrast, if a signed graph is unbalanced, some cycles are negative, and the signed cycle structure cannot be made all-positive by switching. In this case, the Laplacian spectrum changes: the 0-eigenvalue may disappear, and the geometry of the eigenvectors reflects a mixture of structure and sentiment. For example, in Table 2, the unbalanced graph Σ_5 has no all-± 1 eigenvector at eigenvalue 0. The example illustrates that spectral methods on unbalanced signed graphs primarily rely on the geometry of eigenvectors rather than directly encoding sentiment; the sign information is embedded in the eigenvector entries rather than in the proximity structure used by algorithms such as k-means.

Thus, both Harary cuts and spectral methods can be viewed as methods for bipartitioning a graph, albeit from different perspectives. Once a signed network is balanced, we can form its balanced Laplacian matrix and perform a spectral decomposition whose 0-eigenvector yields a Harary cut. The *graphC* algorithm leverages this balanced–spectral duality indirectly: instead of computing eigenvectors of the balanced Laplacian, it searches over Harary cuts in the space of near-balanced states and selects those that most improve a signed loss function. Thus, we avoid the cost and instability of global spectral decomposition on large signed graphs while still capturing the essential structure suggested by the duality.

4. Measuring signed graph clustering efficacy

Let $\Sigma = (V, E)$ be a signed graph where each edge $e \in E$ is either positive (e^+) or negative (e^-). For any subset of edges $F \subseteq E$, we define the counting functions (with the exception that pos_{out}, pos_{in}, neg_{in}, and neg_{out} are not counting functions and are defined below

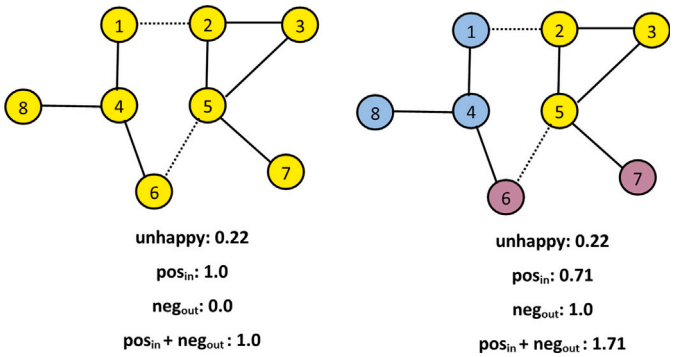

Fig. 2. An illustration emphasizing the advantage of measuring the quality of a clustering assignment in a signed graph using the summation of the fractions of positive edges within communities and the fractions of negative edges between communities. **Dotted** lines are negative edges whereas the **solid**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

using counting functions):

$$pos_F = |\{e \in F \mid e \text{ is positive}\}|, \qquad neg_F = |\{e \in F \mid e \text{ is negative}\}|. \tag{1}$$

Consider two clustering assignments for the network illustrated in Fig. 2. In the absence of ground truth and training data for the community assignment in signed networks, the state-of-the-art uses the *unhappy ratio* to assess clustering quality [17]. The *unhappy ratio* U_i is defined in Eq. (2) for a set of communities produced by method j in the signed graph Σ_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 numbers of positive edges between communities, the numbers of positive edges within communities, the numbers of negative edges within communities, and the numbers of negative edges between communities, respectively.

The *unhappy* ratio for both clustering approaches in Fig. 2 is 0.22. The clustering assignment on the right seems more effective as it avoids disregarding *all* negative edges, successfully segregates two communities (1,4,8,6 and 2,3,5,7), and accurately classifies *most* positive edges. The *unhappy* ratio fails to capture these distinctions. It fails to capture the quality of the clustering in real signed networks because the *unhappy* ratio favors positive edges and trivial clustering cases in networks where the typical number of negative edges is 5–10 times smaller than the number of positive edges [4].

Consider another signed graph with 1000 edges, where 900 are positive, and 100 are negative, with two clustering assignments. In the first clustering, there are 400 positive edges between clusters ($pos_{between}$), 500 positive edges within clusters (pos_{within}), 50 negative edges within clusters (neg_{within}), and 50 negative edges between clusters ($neg_{between}$), yielding an unhappy ratio (U_1) of 0.45. The second clustering has zero positive edges between clusters, 900 positive edges within clusters, 100 negative edges within clusters, and zero negative edges between clusters, resulting in an unhappy ratio (U_2) of 0.1. In this example, the distribution of positive and negative edges is imbalanced. The first clustering can separate some negative edges into different clusters while preserving positive edges within the same clusters. The second clustering is trivial because it places all vertices in a single cluster. Thus, the *unhappy ratio* as defined favors a trivial class when the number of positive edges is much higher than the number of negative edges and vice versa. In the edge-sign imbalance case, the *unhappy ratio* is low for the clustering approach that assigns all vertices to a single community. As a remedy, we change the *unhappy ratio* to the *unhappy score*, which accounts for both positive and negative edges as follows:

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

Intuitively, measuring the sum of the fractions of positive edges within communities (pos_{in}) and the fraction of negative edges between communities (neg_{out}) provides a better assessment of the clustering quality as shown in Fig. 2 [4]. The unhappy ratio for both clustering assignments is the same (0.22), even though the left assignment ignores all negative edges, while the right assignment considers both positive and negative edges. In Fig. 2, the left assignment has an updated $pos_{within} + neg_{between}$ of 1.0. The right assignment has an updated $pos_{within} + neg_{between}$ of 1.71, and the scores more accurately reflect the true clustering quality.

The updated *unhappy score* is the new *graphC* loss measure for a signed graph Σ_i with an unbalanced edge ratio and clustering algorithm j. We rewrite it as a loss function, since our objective in clustering is to minimize $\mathcal{L}(\Sigma_{ij})$ by partitioning the vertices into communities. The quantities pos_{out} and neg_{in} are defined as

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

Thus, we propose to minimize the fraction of violating negative edges $neg_{in}(\Sigma_{ij})$ as well as the fraction of violating positive edges $pos_{out}(\Sigma_{ij})$ simultaneously. In this context, we define the violating edges ($\mathcal{V}$) as the sum of the fraction of positive edges between clusters and the fraction of negative edges inside them.

$$\mathcal{V}_{ij} = \mathcal{L}_{\Sigma_{ij}} = pos_{out} + neg_{in} \tag{5}$$

Symmetrically, we redefine pos_{in} (Eq. 6) and neg_{out} (Eq. 7) measures in terms of fractions to account for the positive and negative edge imbalance.

$$pos_{in} = \frac{pos_{within}}{pos_{between} + pos_{within}} \tag{6}$$

$$neg_{out} = \frac{neg_{between}}{neg_{within} + neg_{between}} \tag{7}$$

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

$$\mathcal{L}_{\Sigma_{ij}}(\alpha, \beta) = \beta(\alpha pos_{out} + (1-\alpha) neg_{in}) + (1-\beta) * \frac{|V_{iso}|}{|V|} \tag{8}$$

The α parameter controls the importance between pos_{out} and neg_{in}. Setting $\alpha = 0.5$ gives equal weight to both. The β parameter controls the fraction of isolated vertices $|V_{iso}|$ produced in the process; $|V|$ is the total number of vertices in the graph. If $\beta = 0$, then the algorithm will completely ignore minimizing the pos_{out} and neg_{in} and will find the cuts that yield the least number of isolated vertices possible. Note that $\mathcal{L}_{\Sigma_{ij}}$ in Eq. (5) is equal to $\mathcal{L}_{\Sigma_{ij}}(0.5, 1)$ in Eq. (8).

5. The *graphC* methodology

As shown in Section 3, the Laplacian matrices of balanced signed graphs with the same underlying graph are isospectral and spectral clustering reconstructs a Harary cut. The proposed *graphC* algorithm searches for the optimal Harary cut using the quality measure defined in Eq. (8), along with a set of stopping criteria to automatically halt the algorithm and return the final clustering labels for each vertex. Fig. 3 outlines the flow of the *graphC* algorithm. It starts by labeling all vertices within the same connected component with the same label, creating an initial clustering assignment set C, as described in Algorithm 1. Isolated vertices receive unique labels, and each isolated vertex is assigned a different label. Specifically, the algorithm first checks whether the connected component is in a set called *processed*. If not, it restores the original signs using the signed network, and then continues. Next, the algorithm checks a time limit and ensures that the connected component's size exceeds the threshold Γ. Once the algorithm identifies the **best Harary cut** based on the loss function in Eq. (8), it evaluates whether the Harary split improves the overall clustering quality, as measured by Eq. (9). Finally, if the improvement exceeds a threshold ϵ, the algorithm commits the split; otherwise, it terminates.

Selecting the Best Harary Cut We use the graphBplus algorithm to compute stable states and perform Harary cuts. To find the best Harary cut, we select the one that yields the smallest loss, as defined in Eq. (8). After selecting the Harary cut, we employ Depth-First Search (DFS) to create the connected components. DFS starts at the root vertex and explores each branch as far as possible before backtracking. The process continues until *graphC* 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 ($|E|$). Algorithm 2 demonstrates the process of selecting the best Harary cut.

Stopping Criteria: After finding the best Harary cut of a particular connected component, the algorithm computes the overall quality of the new clusters of the entire signed network, which is similar to Eq. (8) and works as follows:

$$\mathcal{L}^{t}_{\Sigma_{ij}} = (pos_{out} + neg_{in}) \tag{9}$$

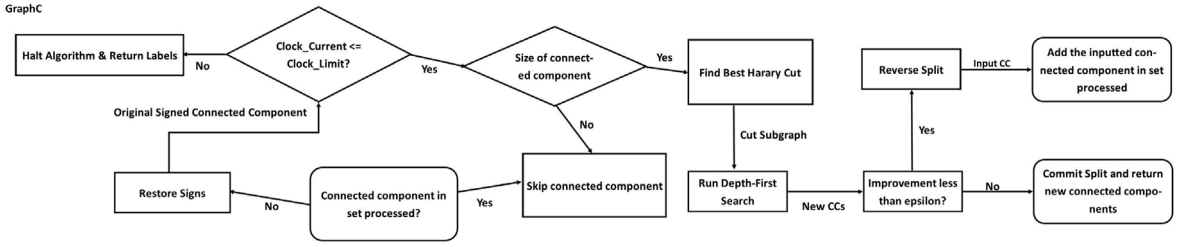

Fig. 3. The *graphC* pipeline: the starting state is the "Connected component in set processed?" block.

Algorithm 1: Split and label components.

Data: Σ_{ij}, label_counter
Result: Clustering label assignment set $C = \{C_v\}, v \in V$

```
label_counter = 0;
for CC, CC ∈ Σ_ij do
    for v, v ∈ CC do
        C_v = label_counter;
    end
    label_counter++;
end
```

Algorithm 2: Best harary cut.

Data: Σ_{ij}, Set R, label_selected, C, α, β, I
Result: G^{FC}
for *every vertex* v *in* Σ_{ij} **do**
  if C_v! = *label_selected* **then**
    R = R + v ;
  end
end
$G^F = G \setminus R$;
Σ = GraphBplus(G^F, I) ;
$G^{FC} = \Sigma|\{\min \mathcal{L}_{\alpha,\beta,G^F}\}$;

Algorithm 3: Choose component to split.

Data: Signed graph Σ_{ij}, spanning tree sampling method M, I, α, β, ϵ, Γ, Time limit t_l
Result: Clustering label assignment set $C = \{C_v\}, v \in V$
label_counter = 0;
Call Algorithm 1 with inputs $G, label\_counter, C$;
set R = $\emptyset$, processed = $\emptyset$;
while *true* **do**
  if $t \geq t_l$ **then**
    break;
  end
  label_selected = -1 ;
  terminate = true ;
  for *label* $l, l \in C$ **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
  $C_t = C$;
  label_counter_temp = label_counter ;
  G^{FC} = Call Alg. 2 (Σ_{ij}, Set R,label_selected, C,α, β, I);
  for $CC \in G^{FC}$ **do**
    for *vertex* $i \in CC$ **do**
      $C_{\geqq}$ = label_counter;
    end
    label_counter++;
  end
  Compute $\mathcal{L}_G^{t+1}$;
  if $\mathcal{L}_G^t - \mathcal{L}_G^{t+1} \leq \epsilon$ **then**
    $C = C_t$ label_counter = label_counter_temp;
    Append label_selected to processed and exit loop;
  end
  Set $\mathcal{L}_G^t = \mathcal{L}_G^{t+1}$;
  $R = \emptyset$;
end

The ϵ parameter is a defined stopping condition for the graph clustering. If the new clusters yield an improvement of less than ϵ, the algorithm undoes the last split of the connected component and adds it to the set *processed*. The *graphC* algorithm ignores any connected components in the *processed* set going forward. Next, *graphC* restores the original signs of the target connected component before invoking the graphBplus algorithm again. If the improvement of the split is greater than ϵ, the split is adopted, and *graphC* proceeds to the next connected component. Algorithm 3 outlines the steps of the proposed community discovery

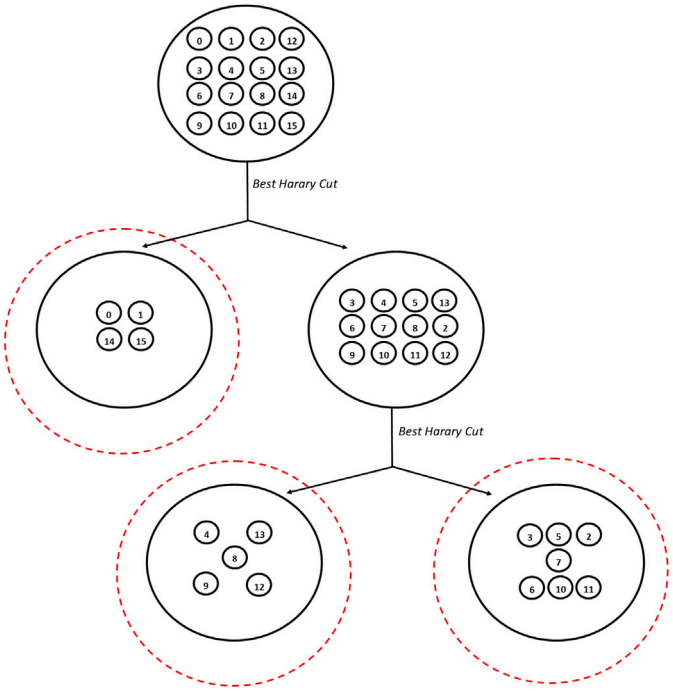

Fig. 4. Execution of the *graphC* algorithm on the highland signed graph comprising 16 vertices and 58 edges.

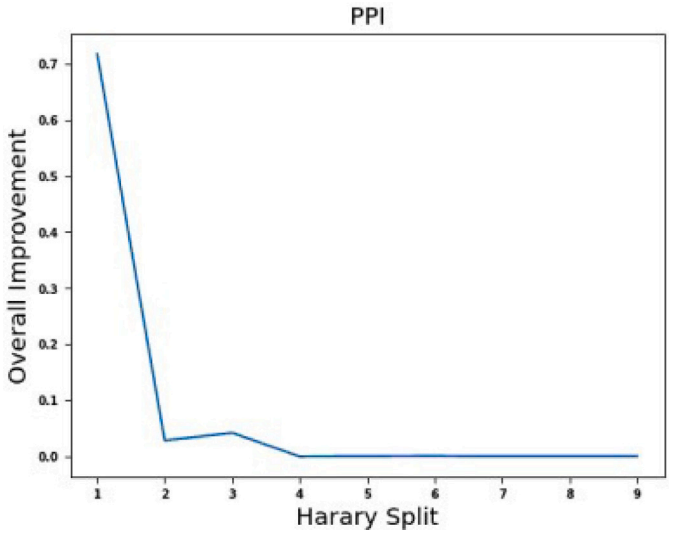

Fig. 5. Example output of the overall improvements with each **committed** harary split of every connected component of our proposed algorithm on the signed PPI graph.

algorithm. The t_l is an optional time limit parameter for the algorithm. If $t_l = -1$, the algorithm will run until it meets the ϵ stopping criterion.

Diminishing Returns (Fostering Scalability): A characteristic of hierarchical clustering is that successive splits yield progressively smaller improvements. Finding the best Harary cut for each connected component is computationally expensive, especially on massive signed graphs with millions of relatively small connected components, yielding only marginal improvements, if any. This observation justifies using the optional Γ parameter to save computation time. It represents a trade-off between efficiency and performance. If the connected component's size is less than Γ, the algorithm ignores it and will not find the best Harary cut even if it is not in the *processed* set. Fig. 5 illustrates a typical execution of the algorithm (on the signed PPI graph) and its gradually decreasing improvement over each Harary split. The overall improvement dropped from around 0.7 to a meager 0.0007 in the 5th Harary cut.

Illustrative Execution on Highland Tribes Graph: Fig. 4 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 whether it has exceeded the time limit t_l; if not, it proceeds. Next, the *graphC* algorithm checks whether the initial component at level 0 is in *processed*. The first pass proceeds

directly to the next step because the *processed* set is empty. Then, *graphC* restores the original signs and finds the best Harary cut for the processed component. In the first iteration, the largest connected component of the graph consists of two components. This step is iteratively repeated for both components. After performing the Harary split, the algorithm stops splitting the current component when the improvement is less than ϵ. Recall that it undoes the last split and commits the connected components found so far, as illustrated in Fig. 4 with red dashed circles. The vertices within that component are given a unique final label. Eventually, the two components at level 1 also satisfy the same stopping criteria, yielding 3 clusters.

6. Complexity analysis

6.1. Space complexity

The *graphC* algorithm stores the signed input graph in Compressed Sparse Row (CSR) format after an initial preprocessing phase that filters out self-loops, inconsistent edges, and nodes belonging to the removed set R. All auxiliary data structures such as arrays used to store edge information, the label map, and the copies of graphs created are linear in the size of the graph. As a result, the overall space complexity of the *graphC* algorithm is $O(|V| + |E|)$.

6.2. Time complexity

6.2.1. GraphBplus(G^F, I)

The time complexity for reading the graph is $O(q\log q + |E|\log|V|)$. The formula takes into account the nature of the underlying data structures used in the C++ code (`std::set` and `std::map`, which use an underlying self-balancing binary tree) after preprocessing and constructing the CSR structure (linear time), where q is the number of edges in the input (before filtering). The following steps run I times: Generating a Breadth-First Search spanning tree takes $O(|V| + |E|)$ time. The core algorithm used for balancing is graphBplus [29], where processing the cycles and balancing takes $O(|E|\log(|V|)t)$ with t being the average degree of the vertices on a cycle. The GraphBplus [29] algorithm then executes a function for computing connected components on the balanced positive edges (Union-Find), constructing a `std::set` of super-edges between components, and determining even/odd hop parity from the largest component via Bellman-Ford on the meta-graph (meta-graph is a graph where each node represents a connected component and each super-edge represents a unique connection between components). The function runs in $O(|E|\log|V| + p|E'|)$ time, where p is the number of components and E' is the set of super-edges. Moreover, the runtime for Harary cuts is $O(|E|)$. The runtime for building the new graph and forming connected components after splitting in our implementation is $O(|E|\log|V| + |V| + |E|)$ because we employ Depth-First Search, and the `std::map` access is $O(\log|V|)$ for traversing the remaining edges. Computing Eq. (8) takes $O(|E|)$, as the algorithm traverses all edges of the graph ($O(|E|)$) and compares the labels of the nodes along each signed edge. If it finds a better clustering assignment with a lower value for Eq. (8), it overwrites the current clustering assignment with the better solution, which takes $O(|V|)$ time. Plus, the code cleanup, such as freeing memory, takes $O(1)$. Therefore, the overall time complexity of GraphBplus(G^F, I) is $O\big(I(|E|\log|V| + p|E'|)\big)$.

6.3. Algorithm 3

The outer loop in Algorithm 3 repeatedly selects the next unprocessed component whose size exceeds Γ. First, in this loop, the time complexity of computing connected component sizes in our code is $O(|V|\log|V|)$ because we use a `std::map` to store the frequency of each label for each node. Next, finding the next connected component to split (size above Γ and unprocessed) also takes $O(|V|\log|V|)$ because we iterate through every node and check if the size of connected component associated with this node's label is less than or equal to Γ to skip it which is done by accessing the label frequency map ($O(\log|V|)$). In addition, we check whether the node's label is already in the set of processed labels ($O(\log w)$), where w is the number of processed connected components. Next, we construct the set R as in Algorithm 2 by adding nodes whose labels differ from that of the selected label (i.e., the connected component to be processed).

The complexity of the algorithm is $O(|V|\log|V|)$ as it iterates through the nodes and inserts every node into `std::set` in every iteration in the worst-case scenario. Next, the GraphBplus(G^F, I) costs $O\big(I(|E|\log|V| + p|E'|)\big)$ as analyzed above, especially during the early splits (large connected components). Afterward, we create a temporary `std::map` containing a copy of the labels in case we need to reverse the split if the overall improvement is not greater than or equal to ϵ. This copy takes $O(|V|\log|V|)$ time. Next, we apply the new labels to the nodes of the newly connected component after the split, which runs in $O(|V|\log|V|)$ time. We then compute the overall quality of the new clusters of the entire signed network as in Eq. (9). This is done by scanning all the edges of the original graph ($O(|E|)$) and comparing nodes' labels along a signed edge ($O(\log|V|)$), which makes the time complexity of this evaluation $O(|E|\log|V|)$. Finally, if the overall improvement is less than ϵ, we reverse the split by copying the labels from the temporary `std::map` into the modified main one, in $O(|V|\log|V|)$. Otherwise, we commit the split ($O(1)$). Overall, the time complexity of each iteration of this outer while-loop is $O\big(I(|E|\log|V| + p|E'|)\big)$ because $I(|E|\log|V| + p|E'|)$ is the dominant term and GraphBplus(G^F, I) is the most expensive computationally.

6.3.1. Overall complexity of graphC

We assume x is the number of committed splits and r is the number of reverted splits. The total time complexity of *graphC* is:

$$O\big((x + r)(I(|E|\log|V| + p|E'|))\big)$$

Table 3
Ten selected leading methods from the literature.

SOTA	Description
Laplacian_none [9]	spectral clustering using the signed graph Laplacian
Laplacian_sym [9]	spectral clustering using the symmetric Laplacian.
BNC_none [10]	balanced normalized cuts.
BNC_sym [10]	symmetric balanced normalized cuts.
SPONGE_none	SPONGE implementation.
SPONGE_sym	symmetric SPONGE implementation.
Hessian [11]	clustering based on signed Bethe Hessian.
A_sym [31]	spectral clustering using symmetric adjacency matrix.
dns [32]	clusters the graph using eigenvectors of the bns Laplacian matrix.
sns [32]	clusters the graph using eigenvectors of the sns Laplacian matrix.

7. Proof of concept implementation

7.1. Implementation and setup

We select 10 established baselines that span classical and state-of-the-art spectral methods for signed graph clustering. These include standard approaches such as signed Laplacian-based spectral clustering [9], balanced normalized cuts [10], and Bethe Hessian-based methods [11], as well as more recent techniques such as SPONGE and its variants that leverage the adjacency matrix and specialized Laplacian matrices [31]. Although newer clustering approaches exist, they often require additional information such as node features or labels, which is unavailable in our evaluation setting. The unsupervised baselines above are therefore the most appropriate for a fair comparison, as they operate under the same constraints and are widely used in the literature for benchmarking signed graph clustering [17]. The implementations of the baseline methods listed in Table 3 are implemented in Python in the *SIGNET* repository [31].

The parameters we used for all Konect signed graphs are $I = 1000$, $\alpha = 0.5$, $\beta = 1$, $\epsilon = 0.00000001$, $\Gamma = 2$, and $t_l = 100{,}000$ except for WikiConflict and WikiPolitics, where we chose $\Gamma = 10$ to reduce the runtime. The parameters used for all Amazon signed graphs are $I = 50$, $\alpha = 0.5$, $\beta = 1$, $\epsilon = 0.00000001$, $\Gamma = 40$, and $t_l = -1$. Next, we ran k-means++ clustering 10 times with different random centroid seeds for all spectral methods to cluster the eigenvectors. For all graphs, the number of communities k was chosen based on two recent papers [17], as outlined in Table 4. If the signed graph does not have a known k in the literature, we assigned a k value equal to the k value of the most similar graph in size. We implemented the *graphC* algorithm in C++ to automatically discover optimal communities without a predefined number of clusters k. The *graphC* algorithm finds and commits the optimal Harary cut for each connected component if it satisfies the following conditions: it is not in the $processed$ set, does not exceed the time limit t_l, its size is larger than Γ, and the Harary cut leads to an improvement greater than ϵ. The code for running the leading ten methods in Table 3 and for *graphC* implementation is available at https://anonymous.4open.science/r/graphC-2046/.

The operating system used for the experiments is Ubuntu Linux 20.04.3. The CPU is an 11th-generation Intel(R) Core(TM) i9-11900 K @ 3.50 GHz 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 an NVIDIA GeForce RTX 3070 with 8GB of memory. The driver version is 495.29.05, and the CUDA version is 11.5.

7.2. Signed graph datasets

All signed graphs are preprocessed to remove self-edges, inconsistent edges, and duplicate edges (keeping only the first occurrence of each). Neutral or missing-sign edges are treated as positive. Table 4 describes the graphs derived from the Konect benchmark [33]. *Highland* is the signed social network of tribes that indicates a Gama alliance structure of the Eastern Central Highlands of New Guinea, originally studied by Kenneth Read [34]. Positive edges represent alliances and negative edges correspond to enmities, making it a canonical example of structural balance in small social networks. *Congress* is a signed network in which vertices are politicians speaking in the United States Congress, and a directed edge denotes that one speaker mentions another; edge signs reflect agreement or disagreement in discourse.

In the *Chess* network each vertex is a chess player and a directed edge represents a game: the white player is the source and the black player is the target where the edge sign encodes the outcome. *BitcoinAlpha* is a user–user trust/distrust network from the Bitcoin Alpha platform for trading Bitcoins, and *BitcoinOTC* is a user–user trust/distrust network from the Bitcoin OTC platform; in both graphs, positive edges represent trust ratings and negative edges represent distrust, which makes them natural testbeds for signed community detection in financial and reputation systems. *WikiElec* is the network of users from the English Wikipedia who voted for and against each other in administrator elections, where edge signs encode support or opposition. The *SlashdotZoo* graph is the reply network of the technology website Slashdot, where vertices represent users and edges represent signed interactions indicating friend/foe relationships.

The edges of *WikiConflict* represent positive and negative conflicts between users of the English Wikipedia, capturing collaborative and adversarial editing behavior. *WikiPolitics* is an undirected signed network that contains interactions between users of the English Wikipedia who have edited pages about politics; each interaction, such as text editing and votes, is valued positively or negatively to reflect supportive versus contentious relations. *Epinions* is the trust and distrust network of the Epinions online product rating

Table 4
Konect + benchmark: konect [33] as well as TwitterReferendum [35], PPI [17], and WikiRfa [17] signed graphs. LCC stands for the largest connected component, k is the predefined number of clusters for the spectral clustering methods, and the *graphC* results list the number of clusters with at least five elements and the number of clusters with fewer than five elements. The last column is the number of splits produced after the algorithm's execution.

	LCC		k	GraphC		
Dataset	# vertices	# edges	# clusters	# clusters≥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	3058	11,860	10	29	887	16
BitcoinAlpha	3775	14,120	10	14	201	17
BitcoinOTC	5875	21,489	20	22	518	10
Chess	7115	55,779	30	46	1014	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	7634	175,787	30	25	1419	24
WikiElec	7066	100,667	30	38	1528	25
WikiConflict	113,123	2,025,910	100	208	66,083	32
WikiPolitics	137,740	715,334	100	875	18,964	89

site, where directed trust and distrust links connect users. Table 4 lists the characteristics of these Konect signed graphs, as well as three additional signed networks: (1) *TwitterRef*, which captures data from Twitter concerning the 2016 Italian Referendum, where different stances between users signify a negative tie and the same stances indicate a positive link [35]; (2) *Sampson25*, which models the sentiment over time between novice monks in a New England monastery, as captured by Sampson [36]; and (3) *PPI*, which models a signed protein–protein interaction network where activating and inhibiting relations are represented as positive and negative edges, respectively [17]. The *WikiRfa* network describes signed voting information for electing Wikipedia administrators [17].

Table 6 outlines the characteristics of the largest connected components for the signed graphs derived from the Amazon ratings and reviews benchmark dataset. The Amazon collection comprises 17 signed graphs constructed from the Amazon rating and review files [37]. The dataset contains product reviews and metadata from May 1996 to July 2014. Each rating score from 1 to 5 is mapped to an edge between a user and a product as follows: $(5,4) \rightarrow m^+$ (positive), $3 \rightarrow m$ (no sign), and $(2,1) \rightarrow m^-$ (negative) [37]. We treat m as neutral and exclude it when forming signed edges. Table 6 reports the number of users, items, edges, and sign imbalance in the largest connected component of each derived graph, which is the portion processed by *graphC* and the baselines.

In recommendation system terms, clustering a user–item bipartite signed graph can reveal groups of users with similar preferences and groups of items that are consistently liked or disliked within the same communities. Such clusters support tasks such as targeted recommendations, catalog organization, and anomaly detection (e.g., identifying users or products with unusual sign patterns). Analogously, clustering a bipartite graph of authors and papers can identify research communities and topics based on signed interactions such as citations, endorsements, or negative evaluations. Across the Konect social/trust networks, TwitterRef, PPI, and the Amazon graphs, this dataset suite therefore covers small, medium, and web-scale signed networks arising from social interaction, political discourse, biological regulation, and large-scale recommendation scenarios, providing a diverse testbed for evaluating both the quality and interpretability of *graphC*'s hierarchical signed clusters.

8. Signed graphs experiment

The *graphC* algorithm records the number of clusters it discovers (including isolated vertices) and the number of Harary split operations executed on each signed network. To understand how hierarchical refinement affects clustering quality, we analyze the impact of successive Harary splits on the global loss in Eq. (9) using the two evaluation metrics pos_{in} (Eq. 6) and neg_{out} (Eq. 7).

8.1. GraphC for konect signed graphs

Fig. 6 illustrates this behavior on the WikiElec signed graph. At the beginning, when all vertices belong to a single cluster, we have $pos_{in} = 1.0$ and $neg_{out} = 0.0$ because every positive edge is internal and every negative edge is also internal. As *graphC* applies Harary splits to the largest connected component, each accepted split aims to reduce the global loss by decreasing pos_{out} and neg_{in}, thereby increasing pos_{in} and neg_{out}.

Fig. 6 shows that reducing pos_{within} (and hence pos_{in}) with additional splits is unavoidable, since cutting a balanced component always removes some positive edges from within clusters. However, during the first few splits, the rate at which $neg_{between}$ (and thus neg_{out}) increases is much higher than the rate at which pos_{within} decreases, leading to a net improvement in the combined quality measure $pos_{in} + neg_{out}$. After a small number of levels, the curve flattens: additional splits produce only marginal gains in neg_{out} while continuing to erode pos_{in}. This plateau behavior reflects diminishing returns in hierarchical refinement. The parameter Γ allows *graphC* to stop splitting components once they become too small to offer meaningful improvements, thereby avoiding the plateau region and saving computation time.

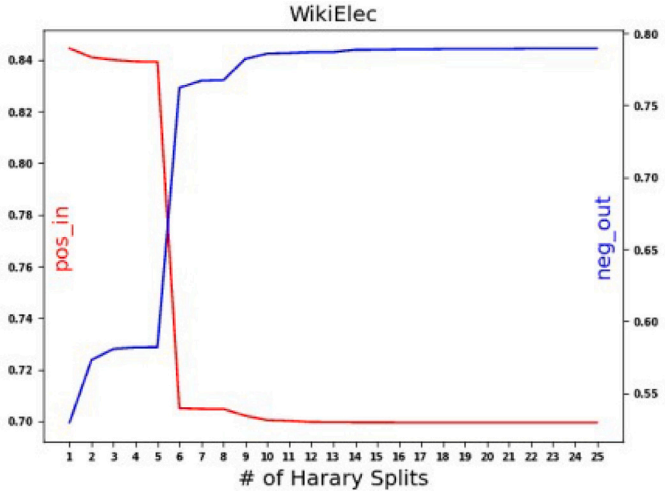

Fig. 6. pos_{in} and neg_{out} for WikiElec with each harary split during algorithm execution.

Table 5
Evaluation results of our proposed algorithm against ten different leading methods on 14 datasets in terms of $pos_{in}(G)$ and $neg_{out}(G)$ (as percentages and in this table, renamed pos and neg respectively), and time in seconds. Empty cells indicate that the method failed to run on the dataset.

Method →	*GraphC*			A_{sym}			L_{none}			L_{sym}			BNC_{none}			BNC_{sym}		
Dataset ↓	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	0.1	93	100	0.1	93	100	0.1	93	100	0.1	93	100	0.1
Sampson25	70	90	0.2	60	91	0.04	59	79	0.1	63	92	0.03	78	65	0.03	62	91	0.04
Congress	93	100	0.77	70	100	0.11	86	97	0.12	24	100	0.11	39	77	0.13	93	97	0.14
PPI	81	98	9	100	2	1	100	0.98	1	93	10	1	100	0.98	1			
BitcoinAlpha	81	86	14	100	0.5	2	30	68	3	72	24	3	55	66	2	100	3	2
BitcoinOTC	82	90	22	71	42	4	21	80	7	61	43	4	86	61	4	100	35	4
Chess	39	84	45	100	0.17	7							65	34	7	100	0.21	7
TwitterRef	89	100	446	65	90	31				67	94	31	23	93	30	100	0.66	31
WikiRfa	61	81	83	79	30	18	100	0.003	19	74	35	18	100	2	18	100	0.01	18
WikiElec	70	79	60	19	65	11	100	0.05	21	100	0.05	11	51	58	11	1	0.11	11

Method →	*GraphC*			Hessian			SPO_{none}			SPO_{sym}			dns			sns		
Dataset ↓	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	0.19	93	100	0.17	93	100	0.2	93	100	0.1
Sampson25	70	90	0.2	63	93	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	84	0.14	67	100	0.12	94	95	0.12	98	33	0.12
PPI	81	98	9	79	35	1	100	0.1	1	100	1	1	100	0.72	1	100	0.77	1
BitcoinAlpha	81	86	14	37	89	2	91	3	3	100	5	3	100	4	2	100	7	2
BitcoinOTC	82	90	22	30	94	4	62	30	8	100	28	4	100	8	4	100	8	4
Chess	39	84	45	40	70	7	100	0	12	100	0.51	8	100	0.2	7	100	0.2	7
TwitterRef	89	100	446	11	98	31	97	4	60	65	94	31	100	0.6	30	100	0.55	31
WikiRfa	61	81	83	54	59	18	9	11	2	91	6	19	100	0.02	18	100	0.01	18
WikiElec	70	79	60	19	92	11	92	12	20	1	0.74	11	1	0.17	11	1	0.16	11

Methods that require a predefined number of clusters k cannot isolate single vertices or very small groups unless doing so happens to align with the chosen k. They typically force isolated vertices into arbitrary clusters to satisfy the k-value constraint, which can artificially inflate pos_{in} while leaving negative edges poorly separated. In contrast, *graphC* naturally produces isolated vertices and small components when justified by the loss: a vertex or subcomponent may split from its parent component as long as the improvement in Eq. (9) exceeds the threshold ϵ. This mechanism enables the algorithm to strike a data-driven balance between preserving dense positive structure within clusters and separating negative ties across clusters, rather than enforcing a rigid k.

On the Konect dataset, *graphC* consistently finds clusterings that achieve a good compromise between $pos_{in}(G)$ and $neg_{out}(G)$, with both metrics taking relatively high values (reported as percentages in the tables). In contrast, many baselines either fail to run on some graphs or predominantly optimize one metric at the expense of the other. Table 4 summarizes, for each Konect signed graph, the number of clusters identified by *graphC* and the number of Harary split operations performed, alongside the corresponding pos_{in} and neg_{out} values; the best results are underlined. The resulting cluster counts highlight how the hierarchy adapts to each graph's structure, rather than being fixed in advance.

We next compare the performance and execution time of *graphC* against ten leading methods from the literature, as summarized in Table 5. Empty cells in the table indicate cases where a baseline either exceeded a 2-day time limit or failed due to non-convergence

Table 6
Amazon ratings and reviews [37] mapped to signed graphs. The *graphC* results include $pos_{in}(G)$ and $neg_{out}(G)$ (as percentages renamed pos and neg respectively in this table) and the resulting number of clusters with at least five elements as well as the number of clusters with fewer 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	9208	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	4456	281,653
Vinyl	1,959,693	3,684,143	74	87	16	16,420	5718	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	1642	1462	205,336
Games	1,489,764	2,142,593	82	22	1662	280,356	8111	272,245
Automoto	950,831	1,239,450	94	79	13	783	8515	202,749
Garden	735,815	939,679	93	85	13	436	4636	153,452
Baby	559,040	892,231	79	91	14	489	2367	94,474
Music	525,522	702,584	87	83	23	571	6124	109,287
Video	433,702	572,834	85	93	14	364	1401	46,481
Instruments	355,507	457,140	90	85	15	281	3536	61,964
Reviews	# vertices	# edges	pos	neg	# splits	time (s)	clusters≥5	# clusters<5
Core Music	9109	64,706	52	84	14	9.57	38	746
Core Video	6815	37,126	57	85	13	5.54	26	737
Core Instrum	2329	10,261	51	92	8	1.90	20	159

of eigenvectors during spectral decomposition. This issue becomes more common on large, sparse, signed graphs with ill-conditioned Laplacians. Across all tested signed graphs, regardless of size or density, *graphC* attains higher or comparable pos_{in} and neg_{out} values than those of the state-of-the-art methods. On some smaller graphs (e.g., WikiElec), our implementation can be slower than certain spectral baselines because their prescribed k is small and the corresponding eigenproblems are relatively cheap to solve. However, as the graph size and k increase, the spectral baselines suffer substantial performance degradation and often fail to complete. In contrast, *graphC* maintains stable behavior by limiting the number of Harary splits via Γ and ϵ and by avoiding global spectral decomposition on the original signed graph.

8.2. GraphC for Amazon ratings and reviews modeling

We map Amazon ratings and reviews [37] to signed bipartite graphs, with negative edges for ratings 1 and 2, neutral (no sign) for rating 3, and positive edges for ratings 4 and 5. We then run *graphC* on the Amazon signed graphs listed in Table 6 to evaluate its scalability on networks with millions of vertices and edges. For example, graphC successfully processes the largest Amazon graph (Books), which contains nearly ten million vertices and over twenty million edges, within approximately 32 hours while achieving high values of pos_{in} and neg_{out}, demonstrating that the method remains effective in highly sparse, large-scale recommendation settings. As Table 6 shows, the Amazon bipartite graphs exhibit a skewed distribution of vertices and edges: most users and items are sparsely connected. In contrast, a relatively small number of highly popular items accumulate many ratings. In the Books signed graph, for instance, about one million clusters discovered by *graphC* contain fewer than five users or books, whereas only about thirty-two thousand clusters contain at least five vertices. This structure is consistent with the “long tail” phenomenon in which a very large number of niche products contribute substantially to overall activity. From a modeling perspective, the Amazon experiments therefore highlight both the scalability of *graphC* and its ability to uncover a fine-grained hierarchy of signed user–item communities that reflects long-tail behavior in real recommendation data.

8.3. Parameter study & analysis

To better understand the parameters I, Γ, and ϵ and their effect on the clustering process, we run *graphC* on the Chess signed graph and perform an ablation study. We first vary the size threshold $\Gamma \in \{2, 1000, 2000, 2500, 3500\}$ while fixing the other parameters at $I = 50$, $\alpha = 0.5$, $\beta = 1$, $\epsilon = 10^{-8}$, and $t_l = -1$. As the four panels of Fig. 7 show, increasing Γ degrades clustering quality (lower pos_{in} and neg_{out}) while reducing the number of Harary splits, the resulting number of communities, and the total execution time. Intuitively, a larger Γ instructs the algorithm to ignore more connected components that are still large enough to admit beneficial splits, which saves computation but leaves some negative edges within communities and some positive edges between communities.

Second, we vary the iteration parameter $I \in \{1000, 800, 500, 50, 10\}$ while keeping $\Gamma = 2$, $\alpha = 0.5$, $\beta = 1$, $\epsilon = 10^{-8}$, and $t_l = -1$ fixed. As summarized in Fig. 8, decreasing I reduces the number of spanning trees sampled when searching for the best Harary split on each component, which gradually lowers clustering quality but significantly reduces execution time. In contrast, the number of committed splits and the final number of clusters remain roughly constant (around 41 splits and 1060 clusters), indicating that I primarily controls how thoroughly we explore the space of candidate Harary cuts rather than how deep the hierarchy becomes.

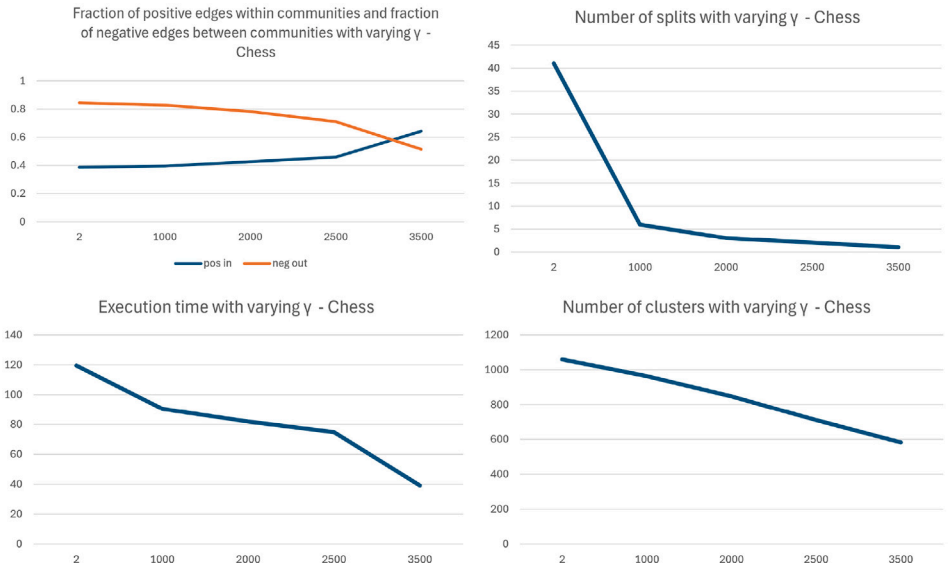

Fig. 7. The effect of varying Γ on the number of clusters, clustering quality, execution time, and harary splits on the chess signed graph. The y-axes show pos_{in} and neg_{out} as in Eqs. (6) and 7, respectively, the number of splits, the execution time, and the number of communities for the four panels (from left to right). The x-axis is the Γ value. The *graphC* algorithm ignores connected components of size less than Γ.

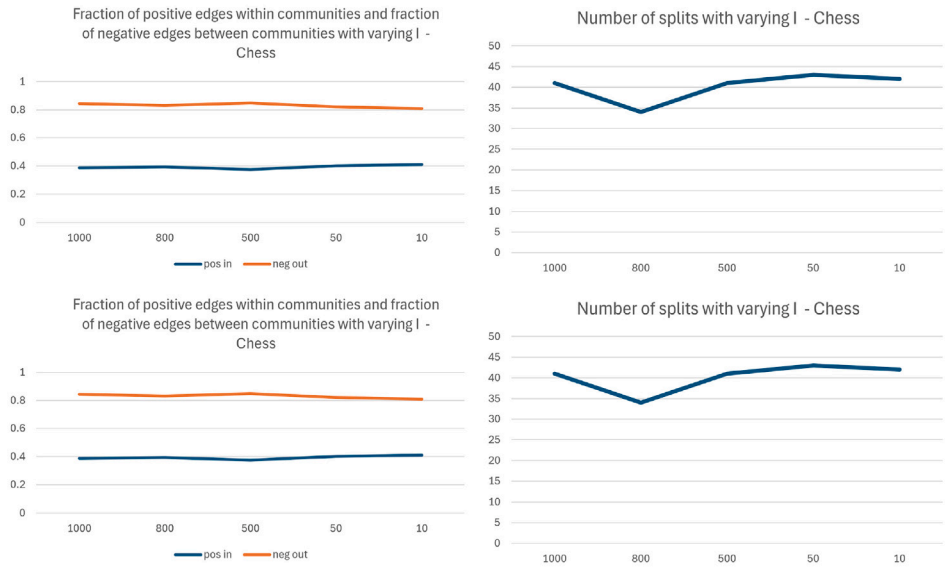

Fig. 8. The effect of varying I (number of iterations per connected component) on the number of clusters, clustering quality, execution time, and harary splits on the chess signed graph. The y-axes show pos_{in} and neg_{out} as in Eqs. (6) and (7), respectively, and the number of splits, the execution time, and the number of communities for the four panels (from left to right). The x-axis is the I value.

Finally, we vary the loss-improvement threshold

$$\epsilon \in \{10^{-9}, 10^{-4}, 10^{-2}, 10^{-1}, 0.5\}$$

while fixing $\Gamma = 2$, $\alpha = 0.5$, $\beta = 1$, $I = 1000$, and $t_l = -1$. The threshold ϵ specifies the minimal global loss decrease required to accept a split. As illustrated in Fig. 9, larger values of ϵ reduce the probability that a candidate Harary cut is committed, which lowers runtime but also leads to worse pos_{in} and neg_{out} because some beneficial (but modest) improvements are rejected. For small to moderate changes in ϵ, the number of clusters stays close to 1060, suggesting that the most informative splits tend to yield relatively large improvements; only when ϵ becomes too large does the hierarchy noticeably become shallower. Overall, the parameter study indicates that high I and low Γ and ϵ values produce the best clustering quality for a given signed graph. In contrast, larger Γ and ϵ (or smaller I) provide principled ways to trade accuracy for runtime.

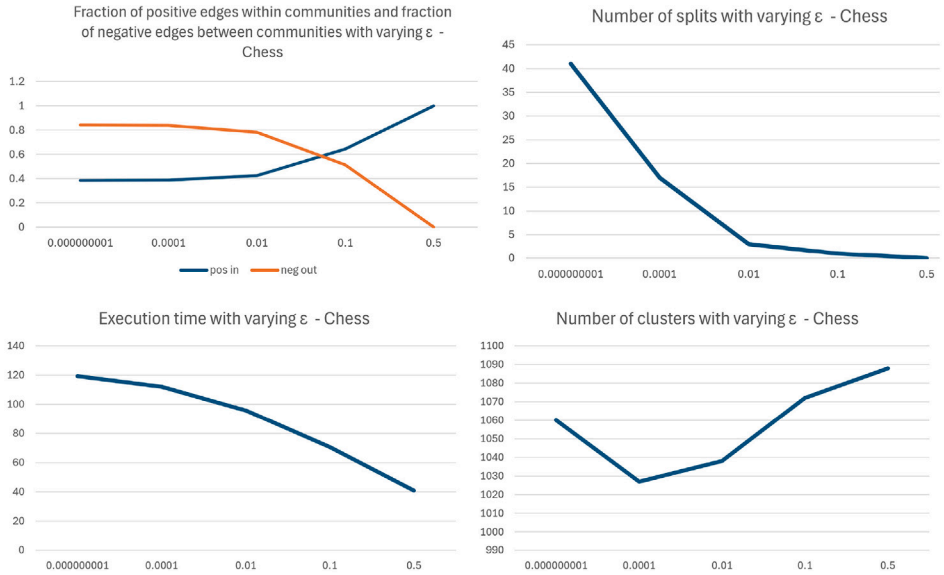

Fig. 9. The threshold ϵ specifies the minimum improvement value for the set split. The image summarizes the effect of varying ϵ on the number of clusters, clustering quality, execution time, and harary splits on the chess signed graph. From left to right, the y-axes are pos_{in} and neg_{out} as in Eqs. (6) and (7), respectively.

8.4. Empirical scalability

We characterize *graphC*'s main components and relate them to the observed empirical scalability. Each iteration of the main loop (Algorithm 3) consists of: (i) selecting a candidate connected component based on its current label and size, (ii) invoking GraphBplus on the induced subgraph G^F to propose a best Harary cut, and (iii) recomputing connected components via DFS and updating the global loss. GraphBplus therefore dominates the cost of each iteration on G^F.

The main source of runtime variability arises from the number of times *graphC* explores and commits Harary splits across different components. This depends on the data in two ways. First, the number of times the algorithm *reverts* previously committed splits is data-dependent: reversals occur only when a prospective refinement fails to improve the loss by more than ϵ, and the frequency of such events is not known a priori. Second, the number of components that satisfy the size threshold Γ and admit loss-improving splits depends on the evolving cluster structure; components that become too small or are added to the *processed* set are no longer refined. In the worst case, the algorithm could attempt many splits on many components, but in practice, the Γ and ϵ thresholds, together with the size of G^F at each step, substantially constrain the number of effective iterations.

Our experiments demonstrate that *graphC* scales to signed graphs with up to nearly ten million vertices and over twenty million edges (Amazon Books), completing in approximately 32 hours on a single machine while maintaining strong pos_{in} and neg_{out} scores. On the Konect and medium-scale signed graphs, *graphC* runs within minutes to hours, and, unlike several spectral baselines, it does not encounter numerical convergence failures on large, sparse signed networks. These empirical results, combined with the parameter study above, provide evidence that *graphC* is practically scalable while offering users explicit knobs (I, Γ, ϵ) to trade clustering quality for computation time.

9. Conclusion and future work

The structure of communities within a network can reveal hidden patterns and insights about relationships and dynamics in complex systems. Community detection is widely used across domains such as biology, social systems, and recommendation platforms to extract such information from large graphs. For signed networks in particular, recent work has highlighted persistent challenges: difficulty in recovering community structure in sparse regimes using spectral methods, sensitivity to the choice of the number of clusters k, eigenvalue contamination in large ill-conditioned matrices, limited scalability to web-scale graphs, and reliance on ground truth labels that are rarely available in practice.

In response to these issues, we proposed *graphC*, a scalable, hierarchical clustering algorithm for signed networks that avoids a predefined k and instead adaptively discovers the number of clusters via Harary splits guided by a balance-theoretic loss. The algorithm is rooted in a balanced–spectral duality: when a signed graph is transformed into a balanced state, its Laplacian becomes conjugate to that of an all-positive graph, and the 0-eigenvector encodes a Harary bipartition. *graphC* leverages this connection indirectly by searching for high-quality Harary cuts without computing global eigenvectors on the original signed graph. We further refined the loss into a conductance-like objective that symmetrically accounts for positive and negative edges, and introduced parameters I, Γ, and ϵ that control the thoroughness of the Harary-cut search, the minimum component size, and the minimal loss improvement required to accept splits.

Our experiments on Konect, Wikipedia, Amazon, and PPI signed graphs show that *graphC* consistently achieves high pos_{in} and neg_{out} values, often outperforming classical and modern spectral baselines that either fail to converge or do not scale to large, sparse signed networks. On Amazon-scale bipartite graphs with nearly ten million vertices and over twenty million edges, *graphC* remains practically feasible, demonstrating that a balance-theoretic, k-independent approach can operate at web scale without requiring node features or labels. The parameter study further clarifies how I, Γ, and ϵ trade off clustering quality, hierarchy depth, and runtime, providing practitioners with explicit knobs to adapt *graphC* to different applications and resource settings.

There are several promising directions for future work. First, we plan to parallelize graphC and its GraphBplus subroutine, exploiting multi-core and distributed architectures to reduce runtime on very large signed graphs, and to perform a more detailed analysis of memory usage on web-scale datasets. Second, we aim to integrate consensus-based features, such as those proposed in prior work on frustration clouds, to stabilize cluster assignments and further improve the robustness of the hierarchical structure. Third, realistic synthetic signed network generators, such as those studied by Shin et al. [38], offer a complementary avenue for evaluation: by combining *graphC* with such generators, we can systematically probe performance under controlled variations in balance, sparsity, sign imbalance, and planted community structure. Fourth, an interesting line of research is to couple *graphC* with recent signed graph neural and attention-based models, using *graphC*'s balance-driven clusters as an interpretable structure for pretraining, regularization, or downstream tasks such as link sign prediction and anomaly detection in large-scale signed networks. Fifth, future extensions will also include evaluation using independent external metrics such as Normalized Mutual Information (NMI), purity, and signed modularity, particularly on benchmark signed graphs with available ground–truth community labels. Finally, we plan to extend *graphC* to dynamic signed graphs by incorporating temporal information into the balance–driven clustering process.

CRediT authorship contribution statement

Muhieddine Shebaro: Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Lucas Rusnak:** Writing – original draft, Methodology, Conceptualization. **Martin Burtscher:** Writing – review & editing, Methodology. **Jelena Tešić:** Supervision, Project administration, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The datasets we used are cited in our manuscript and they belong to others.

References

[1] V.A. Traag, L. Šubelj, Large network community detection by fast label propagation, Sci. Rep. 13 (1) (2023). ISSN 2045-2322.
[2] X. Wang, P. Wang, A.M.-C. So, Exact community recovery over signed graphs, in: G. Camps-Valls, F.J.R. Ruiz, I. Valera (Eds.), Proceedings of the 25th International Conference on Artificial Intelligence and Statistics, Volume 151 of Proceedings of Machine Learning Research, PMLR, Virtual, 28–30 Mar 2022, pp. 9686–9710, https://proceedings.mlr.press/v151/wang22i.html.
[3] M. Cucuringu, A.V. Singh, D. Sulem, H. Tyagi, Regularized spectral methods for clustering signed networks, J. Mach. Learn. Res. 22 (264) (2021) 1–79, http://jmlr.org/papers/v22/20-1289.html.
[4] M. Tomasso, L. Rusnak, J. Tešić, Advances in scaling community discovery methods for signed graph networks, J. Complex Netw. 10 (3) (2022) 06, https://doi.org/10.1093/comnet/cnac013
[5] R.P. Abelson, M.J. Rosenberg, Symbolic psycho-logic: a model of attitudinal cognition, Behav. Sci. 3 (1) (1958) 1–13.
[6] F. Harary, D. Cartwright, On the coloring of signed graphs, Elem. Math. 23 (1968) 85–89, http://eudml.org/doc/140892.
[7] T. Derr, Z. Wang, J. Dacon, J. Tang, Link and interaction polarity predictions in signed networks, Soc. Netw. Anal. Min. 10 (1) (2020) 1–14.
[8] L. Martino, R.S. Millán-Castillo, E. Morgado, Spectral information criterion for automatic elbow detection, Expert Syst. Appl. 231 (2023) 120705. ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.120705
[9] A.V. Knyazev, Signed laplacian for spectral clustering revisited, arXiv preprint arXiv:1701.01394, Jan 2017.
[10] K.-Y. Chiang, J.J. Whang, I.S. Dhillon, Scalable clustering of signed networks using balance normalized cut, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, 2012, pp. 615–624.
[11] A. Saade, F. Krzakala, L. Zdeborová, Spectral clustering of graphs with the bethe Hessian, ArXiv arxiv:1406.1880, 2014, https://api.semanticscholar.org/CorpusID:919316.
[12] D. Cartwright, F. Harary, Structural balance: a generalization of heider's theory, Psychological Rev. 63 (1956) 277–293.
[13] B. Ordozgoiti, A. Matakos, A. Gionis, Finding large balanced subgraphs in signed networks, in: Proceedings of the Web Conference 2020, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1378–1388, https://doi.org/10.1145/3366423.3380212. ISBN 9781450370233.
[14] K. Sharma, I.A. Gillani, S. Medya, S. Ranu, A. Bagchi, Balance Maximization in Signed Networks via Edge Deletions, Association for Computing Machinery, New York, NY, USA, 2021, pp. 752–760. https://doi.org/10.1145/3437963.3441778. ISBN 9781450382977.
[15] C. Xiao, X. Jinhai, D.-C. Luo, Local spectral method for finding polarized communities in signed networks, in: Proceedings of the 15th ACM International Conference on Web Search and Data Mining, 2022, pp. 1013–1021, https://doi.org/10.1145/3366423.3380121
[16] J. Tzeng, D.F. Gleich, Spectral methods for conflicting group detection in signed networks, in: Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020, pp. 15810–15821.
[17] H. Yixuan, G. Reinert, S. Wang, M. Cucuringu, Sssnet: semi-supervised signed network clustering, 2022, https://arxiv.org/abs/2110.06623.
[18] Y.-C. Lee, N. Seo, K. Han, S.-W. Kim, Asine: adversarial signed network embedding, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 609–618, https://doi.org/10.1145/3397271.3401079 9781450380164.
[19] T. Derr, M. Yao, J. Tang, Signed graph convolutional network, *CoRR* arxiv:1808.06354, 2018, http://arxiv.org/abs/1808.06354.
[20] M. Gholami, A. Sheikhahmadi, K. Khamforoosh, M. Jalili, F. Veisi, A new approach to finding overlapping community structure in signed networks based on neutrosophic theory, J. Complex Netw. 12 (1): cnad051, 01 2024. ISSN 2051-1329, https://doi.org/10.1093/comnet/cnad051
[21] P. Mercado, F. Tudisco, M. Hein, Spectral clustering of signed graphs via matrix power means, *CoRR* arxiv:1905.06230, 2019, http://arxiv.org/abs/1905.06230.

[22] Z. Zhang, J. Liu, K. Zhao, S. Yang, X. Zheng, Y. Wang, Contrastive learning for signed bipartite graphs, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 1629–1638, https://doi.org/10.1145/3539618.3591655 9781450394086.
[23] C. Altafini, Spreading and structural balance on signed networks, SIAM J. Appl. Dyn. Syst. 23 (1) (2024) 50–80, https://doi.org/10.1137/23M1545678
[24] Y. Liu, X. Wang, J. Zhang, L. Ming, Y. Yanfeng, Multiview clustering via partitioning the signed prototype graph, IEEE Trans. Neural Netw. Learn. Syst. 35 (11) (2024) 14659–14670, https://doi.org/10.1109/TNNLS.2023.3280924
[25] W. Zhang, L. Xiaoming, J. Chen, Desired clustering in signed networks, IEEE Trans. Netw. Sci. Eng. (2025), https://doi.org/10.1109/TNSE.2025.10442814. Early Access.
[26] J. Wang, W. Xinyi, J. Cheng, Y. Wang, A signed graph approach to understanding and mitigating over-smoothing, In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, (2025).
[27] L. Aronsson, M.H. Chehreghani, An efficient local search approach for polarized community detection in signed networks, In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, (2025).
[28] L. Rusnak, J. Tešić, Characterizing attitudinal network graphs through frustration cloud, Data Min. Knowl. Discov. 6 (Nov 2021), https://doi.org/10.1007/s10618-021-00795-z
[29] G. Alabandi, J. Tešić, L. Rusnak, 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, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN https://doi.org/10.1145/3458817.3476153 9781450384421.
[30] A. Marsden, Eigenvalues of the laplacian and their relationship to the connectedness, 2013, https://api.semanticscholar.org/CorpusID:17239810.
[31] M. Cucuringu, P. Davies, A. Glielmo, H. Tyagi, SigNet (2019).
[32] Q. Zheng, D.B. Skillicorn, Spectral embedding of signed networks, in: Proceedings of the 2015 SIAM International Conference on Data Mining (SDM), Vancouver, British Columbia, Canada, 2015. SIAM. 55–63, https://doi.org/10.1137/1.9781611974010.7
[33] J. Kunegis, KONECT: the Koblenz network collection, in: Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 1343–1350.
[34] K. Read, Cultures of the central highlands, new Guinea, Southwest. J. Anthropol. 10 (1) (1954) 1–43.
[35] M. Lai, V. Patti, G. Ruffo, P. Rosso, Stance evolution and twitter interactions in an Italian political debate, In M. Silberztein, F. Atigui, E. Kornyshova, E. Métais, F. Meziane (Eds.), Natural Language Processing and Information Systems, Springer International Publishing, Cham, 2018, pp. 15–27, ISBN 978-3-319-91947-8.
[36] S. Sampson, A Novitiate in a Period of Change: An Experimental and Case Study of Relationships (Ph.D. thesis), Cornell University, 1968.
[37] H. Ruining, 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.
[38] K. Shin, F.D. Papadopoulos, A. Vahdat, Realistic synthetic signed network generation and analysis, in: Proceedings of the 2024 ACM Web Conference, 2024, pp. 1–12, https://doi.org/10.1145/3627673.3680274