ABCD: Algorithm for Balanced Component Discovery in Signed Networks

Muhieddine Shebaro, *Student Member, IEEE* and Jelena Tešić, *Member, IEEE*

Abstract—The most significant balanced element in signed graphs plays a vital role in helping researchers understand the fundamental structure of the graph, as it reveals valuable information about the complex relationships between vertices in the network. The challenge is an NP-hard problem; there is no current baseline to evaluate state-of-the-art signed graphs derived from real networks. In this paper, we propose a scalable state-of-the-art approach for the maximum balanced sub-graph detection in the network of *any* size. However, it is still bounded by computational capability. The proposed approach builds on the graph characteristics and a scalable fundamental cycle discovery method to minimize the number of vertices discarded. We evaluate the proposed approach against state-of-the-art and demonstrate over two times higher graph size regarding the number of vertices selected of the discovered subset on an extensive signed network with millions of vertices and edges over the state-of-art in the same time frame.

Index Terms—balanced subgraph, frustration index, balanced states, and signed graphs.

1 Introduction

Signed networks allow for negative weights, representing antagonistic relationships or conflicting opinions [1]. This is because unstructured data requires a rich graph representation. Balance theory represents a theory of changes in attitudes [2]: people's attitudes evolve in networks so that friends of a friend will likely become friends, and so will enemies of an enemy [2]. Heider established the foundation for social balance theory [3], and Harary established the mathematical foundation for signed graphs and introduced the k-way balance [4], [5]. Balance theory concepts have been used to predict edge sentiment, to recommend content and products, or to identify unusual trends [6], [7], [8], [9]. The task of the largest balanced sub-graph discovery has applications in portfolio system's economic risk management [10], computational and operational research [11], community analysis and structure [12], computational biology to model balanced interactions between genes and proteins [13] and social network analysis [14]. The vertices that are part of the maximum balanced sub-graph Σ' of Σ may not necessarily have a high degree of centrality between them. Still, they are essential for understanding how the system behaves. Moreover, by locating the maximum balanced sub-graphs, we can simplify the system into sub-systems with balanced interactions and eliminate inconsistencies regarding unbalanced cycles. This is a well-known NP-hard problem [15], and existing solutions do not scale to real-world graphs [1]. We consider Σ a structure-free signed graph derived from real-life networks with millions of vertices and vertices, e.g., [16]. Signed graph balancing is defined in Section 3, and we propose a solution based on the scalable graph cycle-basis computation of the underlying unsigned graph G of Σ. We use the edge sign switching technique using a fundamental cycle basis discovery method to *search* for the maximum balanced subgraph. The proposed approach finds the largest balanced subgraph Σ' of *any* Σ is $O(K*(n*m))$ where n is the number of vertices, m is the number of edges in Σ and the algorithm considers only the top K balanced states with the lowest frustration index. We use the state-of-art method proposed in [17] for baseline comparisons in Section 5. This selection is because the authors of this recent algorithm proposed in [17] achieved the highest vertex cardinality (number of nodes in the largest balanced subgraph) across all signed graphs among other baselines in the literature. Note that the algorithm proposed in [18] is not the same problem we are targeting. Our problem is a *searching* problem, whereas the algorithm proposed by [18] modifies the graph after finding an initial maximum balanced subgraph using TIMBAL. The problem definition of finding the largest balanced component $\Sigma', |\Sigma'| = n'$ in *any* size signed graph $\Sigma, |\Sigma| = n$ is in Eq. 1.

$$\Sigma' \subseteq \Sigma \wedge Fr(\Sigma') = 0 \wedge \underset{n' \leq n}{\arg\max}\, \Sigma' \implies \Sigma' \tag{1}$$

Note that $Fr(\Sigma')$ represents the Frustration of balanced subgraph Σ'. Hence, the goal is to find a subgraph in a signed network with an even number of negative edges along each fundamental Cycle and its size is as large as possible. The size is usually expressed in terms of vertex cardinality (number of nodes in a subgraph). For definitions and corollaries, see Section 3.

2 Related Work

Finding the optimally balanced sub-graph in a signed graph is known to be an NP-hard problem. Gülpinar et al. proposed the GGMZ algorithm, which begins by computing the input graph's minimum spanning tree. Next, a subset of nodes is selected, and all the edges crossing that subset are inverted to create positive edges. This step is then applied to the entire graph, identifying a set of nodes disconnected by negative edges. This set of nodes is returned as the

M. Shebaro and J. Tešić are with the Department of Computer Science, Texas State University, San Marcos, TX, USA 78666.
E-mail: m.shebaro, jtesic@txstate.edu

Manuscript received Dec 21, 2023

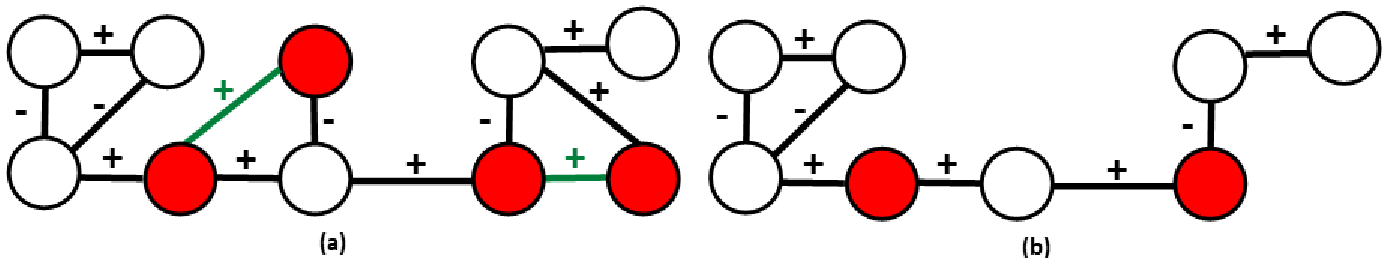

Fig. 1. (a): The unbalanced signed network. Green edges are the candidate edges causing imbalance, and red vertices are the candidate vertices. (b): The maximum balanced signed sub-graph obtained after deleting one candidate node along each edge.

final output of the "algorithm. The system's overall complexity is $O(|N|^3)$ if N is a set of nodes [19]. Poljak and Daniel Turzík show that any signed graph that has |N| nodes and |M| edges contains a balanced sub-graph with at least 0.5|M| + 0.25(|N|- 1) edges [20]. Crowston et al. propose a discovery of a balanced sub-graph of size 0.5|M| + 0.25(|N|- 1+k), k is a parameter, of the complete system's overall complexity is $O(|N|^3)$ if N is a set of nodes [19]. The data reduction is based on finding small separators and a novel gadget construction scheme. The fixed-parameter algorithm is based on iterative compression with a very effective heuristic speedup. Figueiredo et al. first introduced a polyhedral-based branch-and-cut algorithm to find an optimal sub-graph [21], followed by preprocessing routines and initial heuristic improvements in [11]. The proposed GRASP algorithm randomly selects a subset of vertices. It then greedily adds nodes that maximize the number of edges connecting them to the current subset while keeping the size of the subset balanced [11]. The EIGEN algorithm [22] works by first computing the eigenvectors of the Laplacian matrix of the graph. Using the dominant eigenvector of the adjacency matrix, it then partitions the graph into two disjoint sets. The partition is made by setting a threshold value for the eigenvector and assigning each vertex to one of the two sets based on whether its value in the eigenvector is above or below the threshold. The algorithm then recursively this partitioning process on each of the two sets until the desired level of balance is achieved. Sharma et al. proposed a heuristic that deletes edges from the graph associated with the smallest eigenvalues in the Laplacian matrix of the graph until a maximum balanced sub-graph is obtained [18]. Ordozgoiti et al. introduced the most scalable version of the algorithm to date. TIMBAL is an acronym for *trimming iteratively to maximize balance* two-stage method approach where the first stage removes nodes and the second one restores them as long as it does not cause imbalance [17]. Both algorithms rely on signed spectral theory. The approaches do not scale to the large signed graphs as they rely on the costly eigenvalue computation ($O(|N|^2)$), and its performance decreases due to the spectral pollution in eigenvalue computation [23]. TIMBAL proposes a novel bound for perturbations of the graph Laplacian preprocessing techniques to scale the processing for large graphs. The algorithm randomly samples sub-graphs and runs TIMBAL. The nodes deleted from at least one of these sub-graphs are then deleted from the original graph. They evaluate the scalability of the proposed work on graphs over 30 million edges by artificially implanting balanced sub-graphs of a specific size and recovering them [17].

This paper proposes an algorithm for balanced component discovery (ABCD) in signed graphs, and we show that it discovers larger signed sub-graphs faster than TIMBAL. The approach builds on the scalable discovery of fundamental cycles in [24] and utilizes the graph's node density distribution and near-optimal balanced states to minimize the number of vertices removed from the balanced sub-graph. The paper is organized as follows: in Section 1, we formally describe the objective and the problem related to finding the maximum balanced subgraph in the signed network; in Section 3, we present related definitions and corollaries that lead to the proposed solution. n Section 4, we introduce the novel ABCD algorithm and the implementation details; in Section 5, we present proof of concept; and in Section 8, we summarize our findings.

3 Definitions and Corollaries

First, we define the fundamental cycle basis and relevant signed graph network terms.

Definition 3.1. **Signed graph** $\Sigma = (G, \sigma)$ consists of underlying unsigned graph G and an edge signing function $\sigma : m \rightarrow \{+1, -1\}$. The edge m can be positive m^+ or negative m^-. **Sign** of a sub-graph is *product* of the edges signs. **Balanced Signed graph** is a signed graph where every Cycle is positive. **Frustration** of a signed graph is defined as the number of candidate edges whose sign needs to be switched for the graph to reach the balanced state.

Definition 3.2. Graph Σ' is a **subgraph** of a graph Σ if **all** edges and vertices of Σ' are contained in Σ.

Definition 3.3. **Path** is a sequence of distinct edges m that connect a sequence of distinct vertices n in a graph. **Connected graph** has a path that joins any two vertices. **Cycle** is a path that begins and ends at the same node. **Cycle Basis** is a set of simple cycles that forms a basis of the cycle space.

Definition 3.4. For the underlying graph G, let T be the spanning tree of G, and let an edge m be an edge in G between vertices x and y that is *NOT* in the spanning tree T. Since the spanning tree spans all vertices, a unique path in T between vertices x and y does not include m. **The fundamental cycle** is any cycle that is built using path in T plus edge m in graph G.

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

Definition 3.5. The balanced states are **optimal** if and only if it requires a minimum number of edge sign switches in the original graph to reach a balanced state.

Theorem 3.1. If a signed subgraph Σ' is balanced, the following are equivalent [4]:

1) Σ' is balanced. (All circles are positive.)
2) For every vertex pair $(n_i, n_j) in \Sigma'$, all (n_i, n_j)-paths have the same sign.
3) $Fr(\Sigma') = 0$.
4) There exists a bipartition of the vertex set into sets U and W such that an edge is negative if, and only if, it has one vertex in U and one in W. The bipartition (U,W) is called the *Harary-bipartition*.

4 Methodology

Balancing a signed network via edge sign switching identifies a set of candidate edges to have their sign switched. Removing such edges yields the removal of the unbalanced fundamental cycles and produces a unique partitioning of input graphs to balanced subgraphs. Optimal balancing states require minimum edge signed switched, and thus, we consider only the candidates to reach such states. Note that optimal balanced states do not necessarily have the same amount of candidate edges that need to be switched [25]. Their frustrations are different. First, we propose to obtain the variety of optimal balanced state candidates by sampling the graph multiple times and using the fundamental cycle basis discovery method as a base [24], [25]. Next, we propose to process the edges that cause imbalance by deleting one vertex along each candidate edge. The proposed approach removes *one* of the vertices along these candidate edges to minimize the number of vertices simultaneously lost and obtain the largest possible balanced sub-graph. The criteria for choosing the vertices to purge is that if the candidate edge is positive, we delete the one that "carries" fewer vertices (degree/sum of neighborhood degree). f the edge is negative, we exploit the concept of Harary bipartition [4] and remove the vertex in the smaller partition because it would be connected to a smaller number of vertices. Thus, the loss of vertices in the process is minimized to reach a balanced sub-graph and simultaneously maximize the vertex cardinality of that sub-graph. Ongoing research is still investigating the potential adverse effects of selecting this criterion. However, this handling criteria can be modified and improved to yield a greater vertex cardinality. Figure 1 demonstrates an example execution of our algorithm where the candidate edges causing imbalance are identified using [25], proving that the spanning tree-based approach can discover fundamental cycles and balance the graph. Then, one of the vertices (denoted in red) along these candidate edges is removed based on the above-mentioned criteria. In this paper, we introduce the **Algorithm for the Balanced Component Discovery** (ABCD) as a scalable solution for the discovery of the largest balanced subgraph in Alg. 1. The approach produces different optimal balanced states of Σ, as defined in 3.5. We have proposed an efficient data structure and algorithm to discover fundamental cycles if given the spanning tree T [24]. We demonstrated that the discovery and analysis of the fundamental cycles can be computed with linear time complexity and only require a linear amount of memory. The algorithm outline in 1 consists of three steps, as illustrated in Figure 2.

Algorithm 1 ABCD Phase 1

```
1: Fetch signed graph $\Sigma$, number of iterations $I$, and integer $K$ that determines the top optimal balanced states with the lowest frustration index to keep
2: Generate set $\mathcal{T}_i$ of $I$ spanning trees of $\Sigma$
3: Counter to keep only the top optimal balanced states with the lowest frustration index $i = 0; m_K = m;$
4: for i =0; i++; i < I do
5:     for edges $m, m \in \Sigma \setminus T_i$ do
6:         if fundamental cycle $T \cup m$ is negative then
7:             add edge $m$ to $m_i$
8:         end if
9:     end for
10:    if $|m_i| < m_K$ then
11:        $\mathcal{M}_\Sigma \cup = m_k$
12:    end if
13:    if $|\mathcal{M}_\Sigma| > K$ then
14:        Remove the largest set and update $m_K$
15:    end if
16: end for
17: for i =0; i++; i < K do
18:    Create zero vector $H_k$ of dimension $n$
19:    for edge $m \in m_i$ do
20:        switch edge sign in $\Sigma_k$: $m^- \to m^+; m^+ \to m^-$
21:    end for
22:    Cut all the negative edges to create Harary bipartitions $A$ and $B$ so that $|A| > |B|$
23:    for $node$ in $n$ do
24:        if $node \in A, H_k(node) = 1$
25:    end for
26: end for
27: return $\mathcal{M}_\Sigma = m_k, \mathcal{H}_\Sigma = H_k, k \in [1, K]$
```

ABCD phase 1 creates a candidate list of fundamental cycle bases with minimal unbalanced cycles. This algorithm is the backbone of our proposed algorithm in Alg. 1. I is the number of iterations we run the algorithm and the upper bound on how many optimal balanced states we discover in the process. The steps are:
1.1. Discover the fundamental cycle bases for each of the I spanning trees (Alg. 1).
1.2. For each of the cycles in the basis, count the number of cycles that contain the odd number of negative edges (Alg. 1).
1.3. Keep only the $K, K << I$ fundamental cycle basis out of I accessed that have the smallest number of fundamental cycles with an odd number of negative edges (imbalanced Cycle). his translates into lowest cardinality $|m_k|, |m_k| < m - n + 1$ in Alg. 1.

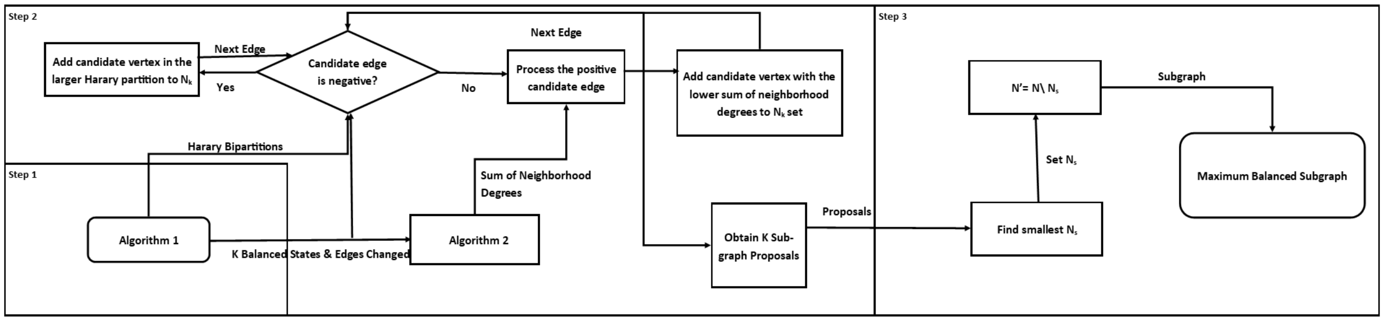
Step 2
Add candidate vertex in the larger Harary partition to N_k
Next Edge
Yes
Candidate edge is negative?
No
Process the positive candidate edge
Next Edge
Add candidate vertex with the lower sum of neighborhood degrees to N_k set
Harary Bipartitions
Step 1
Algorithm 1
K Balanced States & Edges Changed
Algorithm 2
Sum of Neighborhood Degrees
Obtain K Sub-graph Proposals
Proposals
Step 3
N'= N\ N_s
Subgraph
Set N_s
Find smallest N_s
Maximum Balanced Subgraph

Fig. 2. Algorithm for Balanced Component Discovery (ABCD): graph processing flow and illustration of Algorithm 1, Algorithm 2, and phase 3 of the algorithm.

Algorithm 2 ABCD Phase 2

```
1: Harary bipartition array for each node $H_k$, edges of each top-$K$ optimal balanced state with the lowest frustration $m_k$
2: Initialize empty set $n_k = \emptyset$ for all values of k
3: for i =0; i++; i < K do
4:     for edges $m$, $m \in m_k$ do
5:         if $m+$ then
6:             Append the vertex $x$ along $m+$ that has a lower sum of neighborhood degrees to set $n_k$
7:         else
8:             Append the vertex $x$ along $m-$ where $H_k(x) = 0$ to set $n_k$
9:         end if
10:    end for
11: end for
12: return $n_k$ which is the entire set of vertices of the $k$th graph to keep when reconstructing the original graph
```

ABCD phase 2 employs a smart edge deletion approach for all K discovered balanced states as outlined in Alg. 2. The illustrative example is outlined in Figure 3. Minimizing the number of vertices removed from the graph increases the cardinality of the largest balanced sub-graph. *Harary bipartition* separates the vertices of the balanced graph into two sets such that the vertices of both sets internally agree with each other but disagree with the vertices of the other set [4]. The H_k set is created as a labeling vector in Alg. 1. We repeat the following steps for all K identified balanced states for a signed graph Σ, and the heuristic on how we remove the unbalanced fundamental cycles out of possible $m - n + 1$ cycles for the balanced state $k, k \in [1, K]$.
For the m_k list of edges that should have a different sign for the entire graph to be balanced, we initialize an empty set of vertices n_k. or every edge in m_k, Alg. 2 repeats the following steps:
2.1. If the edge is positive, it will be negative in the balanced state. If either vertices is already in n_k, move on to the next edge. Else, add the edge-defining node $node$ to n_k so that $H_k(n) = 0$. If they are both 0 or both 1, move this edge to the end of the m_k set and revisit. The remaining node is in the largest Harary partition for a fully balanced graph, so it will be connected to more vertices than the node ending up in the smaller Harary set after partitioning.
2.2. If the edge is negative and marked for switching to positive, if either vertex is already in n_k, nothing needs to be done; move on to the next edge. e add the node with the lower neighborhood degree to n_k. The computation of the neighborhood degree is demonstrated in Alg 3. e observe that iterating 3 times where in each iteration, we set degree[] to be equal to neighborhood_degree[] before computing an empty neighborhood_degree[] using the new degree[] to improve the results generally.

Algorithm 3 Computation of the sum of neighborhood degree

```
1: Input Signed graph $\Sigma$
2: Declare and initialize array neighborhood_degree = []
3: for v = 0; v++; v < $|n|$ of $\Sigma$ do
4:     Declare and initialize integer sum = 0
5:     for every neighbor $nei$ connected to v via an edge do
6:         sum += degree[$nei$]
7:     end for
8:     neighborhood_degree[v]=sum
9: end for
10: return array neighborhood_degree
```

Fig. 3 illustrates how we compute the neighborhood degree for an exemplar signed graph. Two red vertices in the image indicate the balancing algorithm labeled the edge and that its sign needs to be switched for the graph to achieve a complete balancing state. The degree of the left node is 3, and the right node is 4. The neighborhood degree of the left node is 10 (neighbors of a neighbor), and the neighborhood degree of the right node is 7. We chose the node on the right to delete and the node on the left to keep. If both have the same sum of neighborhood degrees, choose the one connected to another edge in m_k set of edges marked to form an unbalanced fundamental cycle. Note that the *neighborhood* degree is computed for all vertices in the signed graph once and re-used for computation.

ABCD phase 3 finds the index of the smallest sized n_k set among all $n_k, k \in [1, K]$ sets. Let it be index $s : |n_s| \leq |n_k|, k \in [1, K]$. The resulting maximized balanced sub-graph proposal n'_s is finally obtained by removing specific vertices as $n' = n \setminus n_s$. The remaining subset is balanced as all fundamental cycles in the graph are balanced.

Illustrative Example of the ABCD algorithm The full ABCD algorithmic flow is illustrated in Figure 2. Figure 4 illustrates the step-by-step ABCD method on the sampled

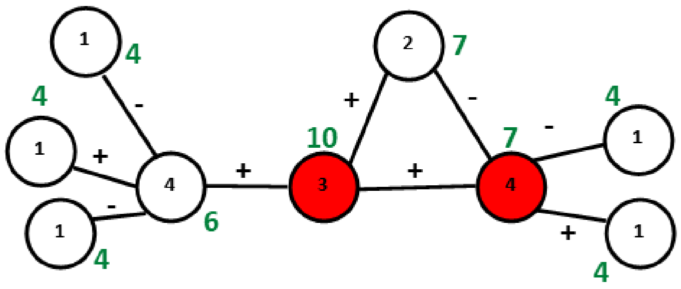

Fig. 3. Degree (black, in node) vs. Sum of Neighborhood Degrees (green, next to the node) computation. The sum of neighborhood degrees labels the red vertices connected by a positive link that will be deleted.

signed graph. The signed graph with seven vertices and ten edges is introduced in the top row. Balancing occurs in ABCD phase 1 (Alg.1), and green text on the edges indicates the changed signs. Green numbers beside vertices are the sum of neighborhood degrees. Orange edges are the edges of the unbalanced fundamental cycles. Green edges are the candidate edges. Red vertices are the candidates for deletion in ABCD phase 2 (Alg.2). The final candidate selection and comparison is in ABCD phase 3, and we found that the remaining subgraph is balanced by removing node $v2$. The red dotted oval over the graph in Step 3 signifies that the final output n'_s of ABCD has a maximal cardinality.

5 Proof Of Concept

All real-world benchmark graphs have one large connected component. The **implementation** of the algorithms is in C++. The algorithm identifies the largest connected component (LCC). It applies the ABCD algorithm to LCC. The implementation treats edges without signs as positive edges in the fundamental Cycle. If the graph has more equal connected components, the implementation accommodates that scenario. ABCD phase 1 (Alg. 1) implementation builds on [24], [25]. [26] have recently shown that the breadth-first search sampling of the spanning trees captures the diversity of the optimally balanced states, and we adopt the breath-first search method for sampling spanning trees in the Algorithm 1. ABCD phase 2 is implemented as listed in Algorithm 2. or the ABCD phase 3, n'_s is constructed by the algorithm re-reading the original graph and skipping the entries with vertices in n_s. **ABCD** algorithm parameters: $I = 5000$ for all, $K = 4000$ for $n < 100,000$, $K = 100$ for $100,000 < n < 300,000$, and $K = 20$ for $300,000 < n$ vertices. **ABCD_Fast** is a faster version of **ABCD** and the parameters are: $I = 1000$ for all, $K = 700$ for $n < 100,000$, $K = 100$ for $100,000 < n < 300,000$, and $K = 20$ for $300,000 < n$ vertices. For this faster version, we can also study the effect of decreasing the number of iterations on the overall speed and performance. We experiment with this version solely on the Konect dataset as in Figure 5 and Figure 6.

Baseline for the proof of concept is TIMBAL [17]. The TIMBAL approach has reached the highest cardinality of the sub-graphs in various datasets and is a defacto state-of-the-art [17]. The input parameters of TIMBAL [17] are set as follows for all subsample_flag=False, samples=4 based on the paper implementation. The parameter max_removals=1 is set for small graphs (under 1000 vertices) and to max_removals=100 for the rest of the signed networks. e set avg_size=20 for datasets of several vertices less than 80,000, and subsample_flag=True, samples = 1000, avg_size = 200 max_removals=100 for datasets with the number of vertices greater than 80,000. TIMBAL is a non-deterministic algorithm, and we run it 5 and 10 times for Konect data to get the maximum node cardinality.

Setup ABCD is run on the same graphs as TIMBAL, and the results are compared side-by-side for 14 Konect and 17 Amazon datasets in terms of runtime in seconds and the size of the produced sub-graph. We verify the balanced state of the discovered subgraph for both methods. The operating system used for the experiments is Linux Ubuntu 20.04.3, running on the 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz with 16 physical cores. t has one socket, two threads per core, and eight cores per socket. The architecture is X86_x64. Its driver version is 495.29.05, and the CUDA version is 11.5. The cache configuration is L1d : 384 KiB, L1i : 256 KiB, L2 : 4 MiB, L3 : 16 MiB. The CPU op is 32-bit and 64-bit.

6 Konect Benchmark Evaluation

Konect signed graphs and their characteristics are described in Table 1. *Highland* is the signed social network of tribes of the GahukuGama alliance structure of the Eastern Central Highlands of New Guinea, from Kenneth Read [27]. *Crisis-InCloister* is a directed network that contains ratings between monks related to a crisis in an abbey (or monastery) in New England (USA), which led to the departure of several of the monks [27]. *ProLeague* are results of football games in Belgium from the Pro League in 2016/2017 in the form of a directed signed graph. Vertices are teams; each directed edge from A to B denotes that team A played at home against team B. The edge weights are the goal difference, and thus positive if the home team wins, negative if the away team wins, and zero for a draw [27]. *DutchCollege* is a directed network that contains friendship ratings between 32 first-year university students who mostly did not know each other before starting university. Each student was asked to rate the other at seven different time points. A node represents a student, and an edge between two students shows that the left rated the right. The edge weights show how good their friendship is in the eye of the left node. The weight ranges from -1 for the risk of conflict to +3 for best friend [27]. *Congress* is a signed network where vertices are politicians speaking in the United States Congress, and a directed edge denotes that a speaker mentions another speaker [27]. In the *Chess* network, each node is a chess player, and a directed edge represents a game with the white player having an outgoing edge and the black player having an ingoing edge. The weight of the edge represents the outcome [27]. *BitcoinAlpha* is a user-user trust/distrust network from the Bitcoin Alpha platform on which Bitcoins are traded [27].*BitcoinOTC* is a user-user trust/distrust network, from the Bitcoin OTC platform, on which Bitcoins are traded [27]. *TwitterReferendum* captures data from Twitter concerning the 2016 Italian Referendum. Different stances

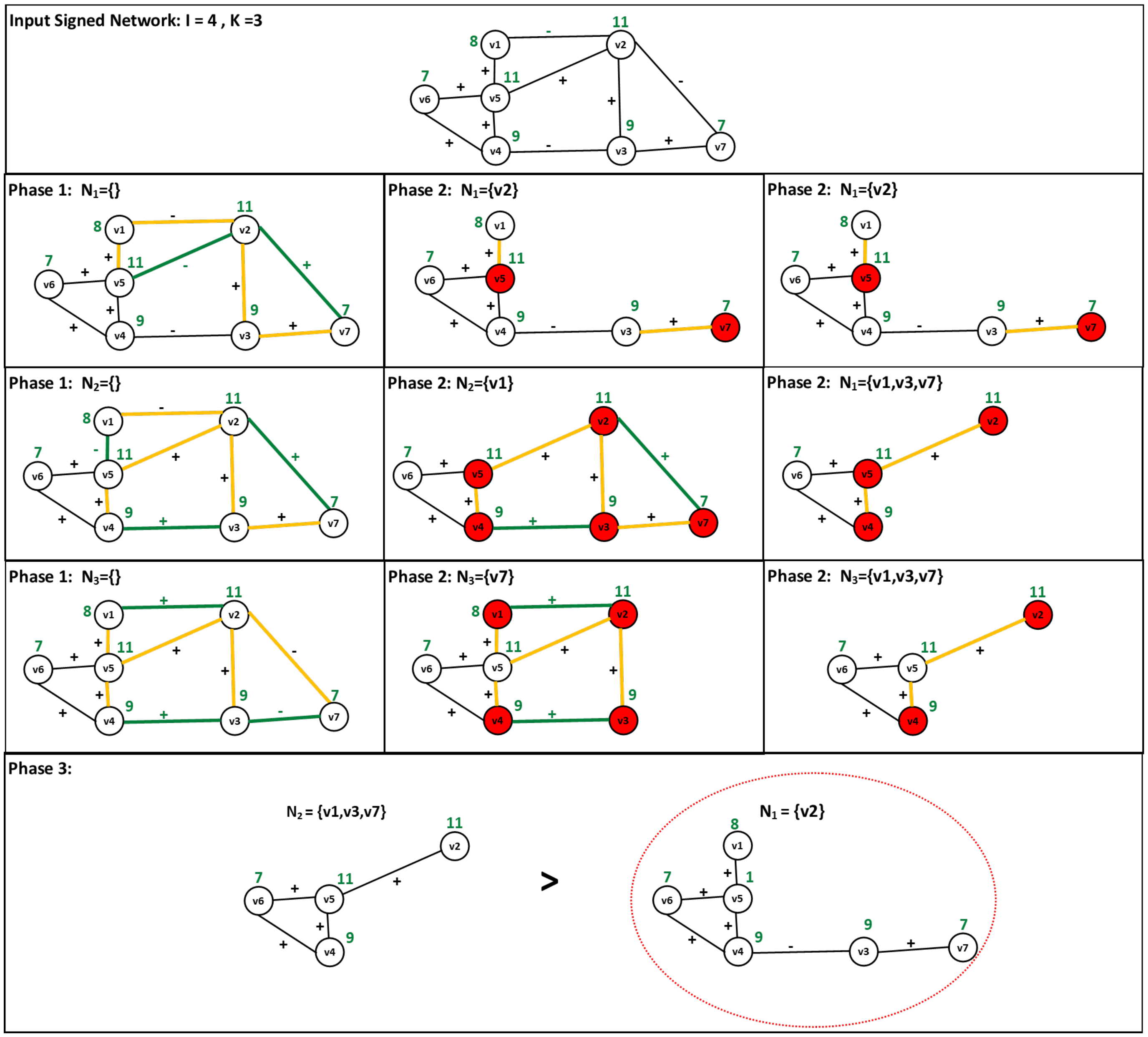

Fig. 4. The ABCD algorithm applied to a sample signed graph with seven vertices and ten edges. Balancing occurs in ABCD phase 1 (Alg.1), and green text on the edges indicates the changed signs. Green numbers beside vertices are the sum of neighborhood degrees. Orange edges are the edges of the unbalanced fundamental cycles. Green edges are the candidate edges. Red vertices are the candidates for deletion in ABCD phase 2 (Alg.2).

between users signify a negative tie, while the same stances indicate a positive link [28]. *WikiElec* is the network of users from the English Wikipedia that voted for and against each other in admin elections [27]. *SlashdotZoo* is the reply network of the technology website Slashdot. Vertices are users, and edges are replies [27]. The edges of *WikiConflict* represent positive and negative conflicts between users of the English Wikipedia [27]. *WikiPolitics* is an undirected signed network that contains interactions between the users of the English Wikipedia that have edited pages about politics. Each interaction, such as text editing and votes, is given a positive or negative value [27]. *Epinions* is the trust and distrust network of Epinions, an online product rating site. It incorporates individual users connected by directed trust and distrust links [27]. *PPI* models the protein-protein interaction network [29].

The first benchmark consists of 14 signed graphs from the Konect repository [27] used in [17] TIMBAL benchmark evaluations. The supplemental PDF document describes Konect signed graphs and their characteristics in great detail. ABCD and TIMBAL performance are outlined in Figure 5 and Figure 6. ABCD matches TIMBAL performance in the smallest three networks. ABCD algorithm finds a more significant subset for 11 Konect datasets. TIMBAL performs better on the three Konect Wiki data sets. TIMBAL is faster than ABCD on smaller networks. For the most extensive graph in the collection, Epinions, ABCD takes double the time to recover the largest balanced sub-graph. We recorded

TABLE 1
Konect sets plus TwitterReferendum and PPI signed graphs attributes: n is the total number of vertices; m is the total number of edges in the graph; The number of edges of the entire input signed network is the reported numbers from the Konect site. The number of edges of LCC is computed locally in our machine with preprocessing as removing duplicate and inconsistent edges is applied.

Konect	**Entire Input Signed Network**		**Largest Connected Component Graph**		
Dataset	# vertices	# edges	# vertices	# edges	**# cycles**
Highland	16	58	16	58	43
CrisisInCloister	18	126	18	126	145
ProLeague	16	120	16	120	105
DutchCollege	32	3,062	32	422	391
Congress	219	764	219	521	303
PPI	3,058	11,860	3,058	11,860	8,803
BitcoinAlpha	3,783	24,186	3,775	14,120	10,346
BitcoinOTC	5,881	35,592	5,875	21,489	15,615
Chess	7,301	65,053	7,115	55,779	48,665
TwitterReferendum	10,884	251,406	10,864	251,396	240,533
SlashdotZoo	79,120	515,397	79,116	467,731	388,616
Epinions	131,828	841,372	119,130	704,267	585,138
WikiElec	7,118	103,675	7,066	100,667	93,602
WikiConflict	118,100	2,917,785	113,123	2,025,910	1,912,788
WikiPolitics	138,592	740,397	137,740	715,334	577,595

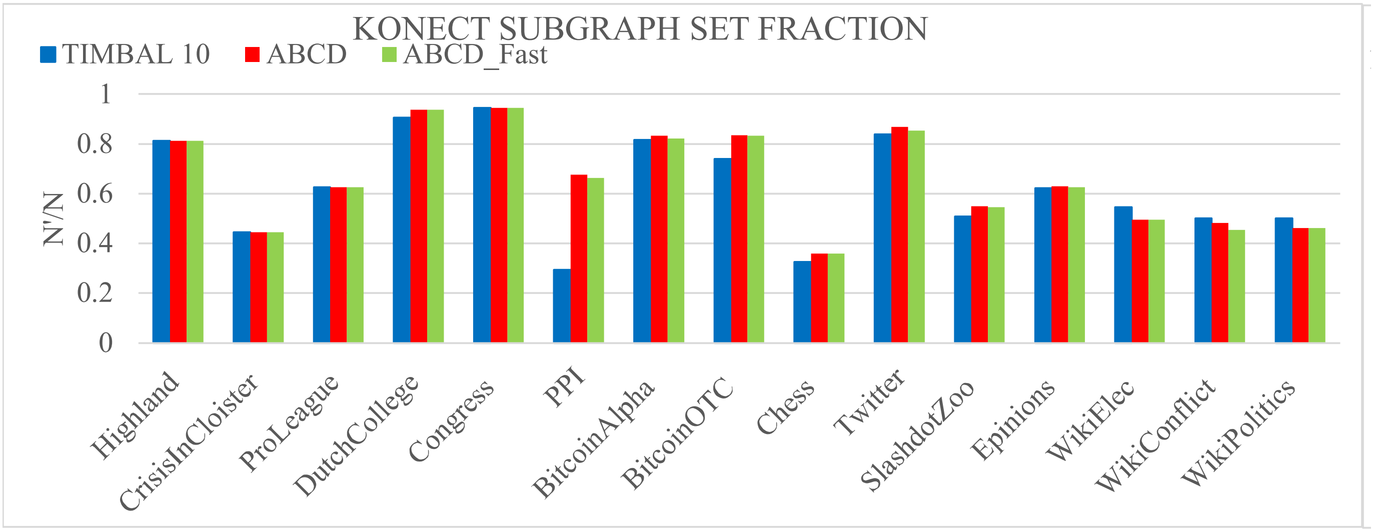

Fig. 5. ABCD and TIMBAL performance comparison for Konect benchmark in terms of subset graph fractions.

the maximum number of vertices obtained after 5 and 10 runs for TIMBAL, and only for one dataset did the repeated runs discover a more significant subset. For ABCD_Fast, we can observe that the performance is consistently better than TIMBAL for the exact runtime for the majority of the graphs.

7 Amazon Benchmark Evaluation

Amazon benchmark consists of 17 signed graphs derived from the Amazon rating and review files [16]. The dataset contains product reviews and metadata from Amazon, spanning May 1996 to July 2014. Rating score is mapped into an edge between the user and the product as follows $(5, 4) \rightarrow m^{+}$, $3 \rightarrow m$ (no sign), and $(2, 1) \rightarrow m^{-}$ [16]. The characteristics of the most significant connected component are outlined in Table 3.

Figure 7 and Table 4 illustrate the sub-graph size TIMBAL recovers (blue box) and the sub-graph ABCD algorithm recovers (red box). Amazon data is extensive. For millions of vertices, the ABCD algorithm performs much better than TIMBAL. One iteration of TIMBAL (blue line) takes as long as the entire ABCD algorithm (red line) for larger graphs. We detail the timing and the experiment in the supplemental PDF tables. In this experiment, the ABCD algorithm has a superior runtime and performance regarding the graph size it discovers, as illustrated in Figure 8. TIMBAL's performance degrades with the graph size, and the discovered sub-graphs are much smaller than what ABCD finds, as illustrated in Figure 7.

7.1 Graph Size vs. Runtime Experiment

evaluation considers runtime for TIMBAL and total runtime for ABCD as a function of the f the number of edges m vertices n in the graph. We use all 31 signed graphs for this evaluation. Details on the graph characteristics are listed in Table 1 and Table 3. Figure 9 illustrates a single TIMBAL run time and ABCD run time as a function of the graph size for all Konect and Amazon graphs. Both algorithms

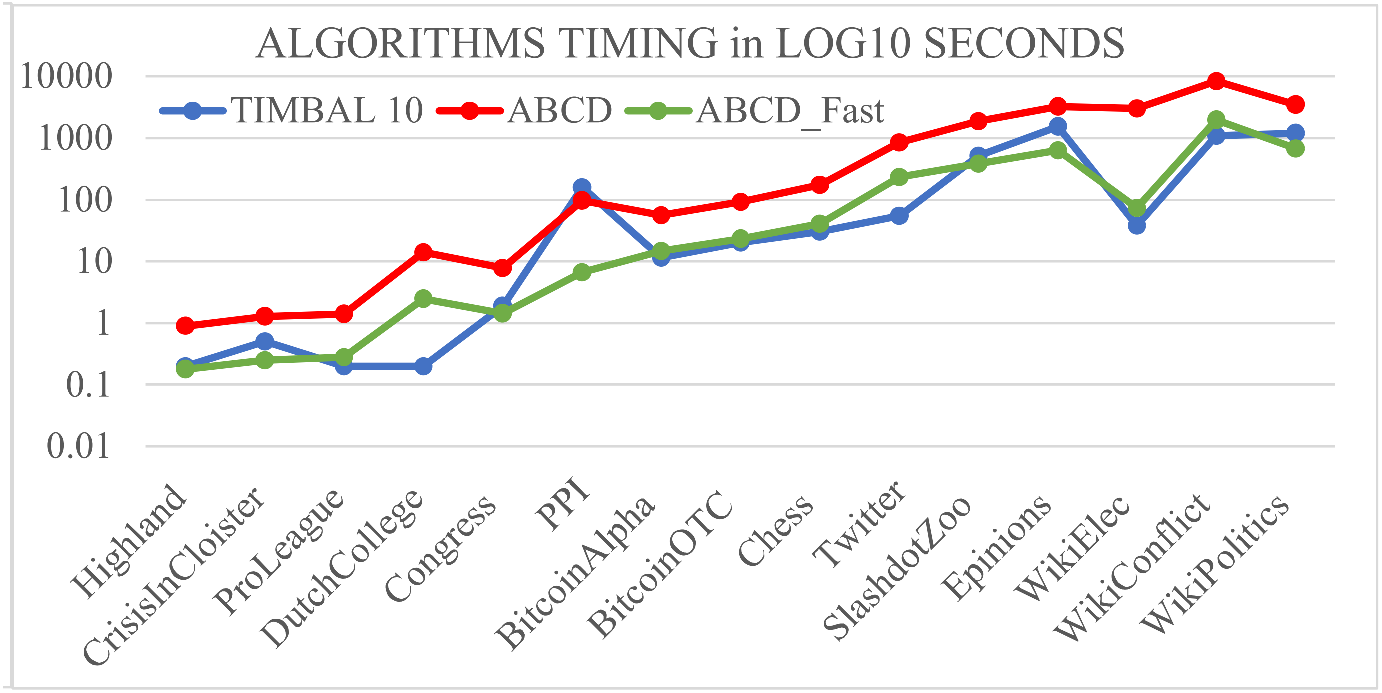

Fig. 6. ABCD and TIMBAL performance comparison for Konect benchmark in terms of algorithmic timing.

TABLE 2
[27] Konect (except TwitterReferendum and PPI, which are not from the Konect site) graph performance comparisons: The largest sub-graph regarding the number of vertices discovered for TIMBAL 5 runs, TIMBAL 10 runs, and ABCD 5000 iterations approach in a given runtime in seconds.

Konect Results	TIMBAL 5 runs		TIMBAL 10 runs		ABCD	
	sub-graph # vertices	Run time (s)	sub-graph # vertices	Run time (s)	sub-graph # vertices	Run time (s)
Highland	13	0.1	13	0.2	13	0.9
CrisisInCloister	8	0.25	8	0.5	8	1.28
ProLeague	10	0.1	10	0.2	10	1.40
DutchCollege	29	0.1	29	0.2	**30**	14
Congress	207	0.85	207	1.9	207	7.79
PPI	900	78.45	900	156.9	**2,072**	97.59
BitcoinAlpha	3,014	5.57	3,081	11.4	**3,146**	55.68
BitcoinOTC	4,250	10.15	4,349	20.3	**4,910**	92.29
Chess	2,230	15.25	2,320	30.5	**2,551**	174.67
TwitterReferendum	9,021	27,6	9,110	55.2	**9,438**	851.94
SlashdotZoo	39,905	259.4	40,123	518.8	**43,544**	1870.20
Epinions	73,433	774.9	74,106	1549.8	**74,843**	3272.87
WikiElec	**3,758**	18.95	**3,856**	37.9	3,506	3033.08
WikiConflict	**56,768**	539.25	56,768	1078.5	54,476	8332.22
WikiPolitics	67,009	602.65	**69,050**	1205.3	63,584	3478.08

have approximate run times that are linear with the size of the graph. ABCD's performance in the most extensive sub-graph discovery is superior to TIMBAL. TIMBAL performance significantly degrades in terms of balanced graph recovery for all graphs over 350,000 vertices.

8 Conclusion

Finding maximum balanced sub-graphs is a fundamental problem in graph theory with significant practical applications. While the situation is computationally challenging, the existing heuristic algorithms have made considerable progress in solving it efficiently for many signed networks and propose a novel scalable algorithm for balance component discovery (ABCD). We capture the information on the unbalanced fundamental cycles and the Harary bipartition labeling for the top unique total cycle bases with the lowest number of unstable cycles. A balanced state with the lowest frustration index for a specific signed network does not necessarily yield a maximum balanced sub-graph. A balanced state with a high frustration index skyrockets the number of vertices discarded due to the increase in the number of candidate vertices and edges to be processed. We introduce a novel set of conditions (neighborhood degree, bi-cut) to remove the vertices from the graph. The output of the ABCD algorithm is guaranteed to be balanced. ABCD eliminates the unbalanced cycle bases by removing the edges. Thus, the Cycle turns into an open path. The resulting subgraph has the most significant size regarding the number of vertices; it

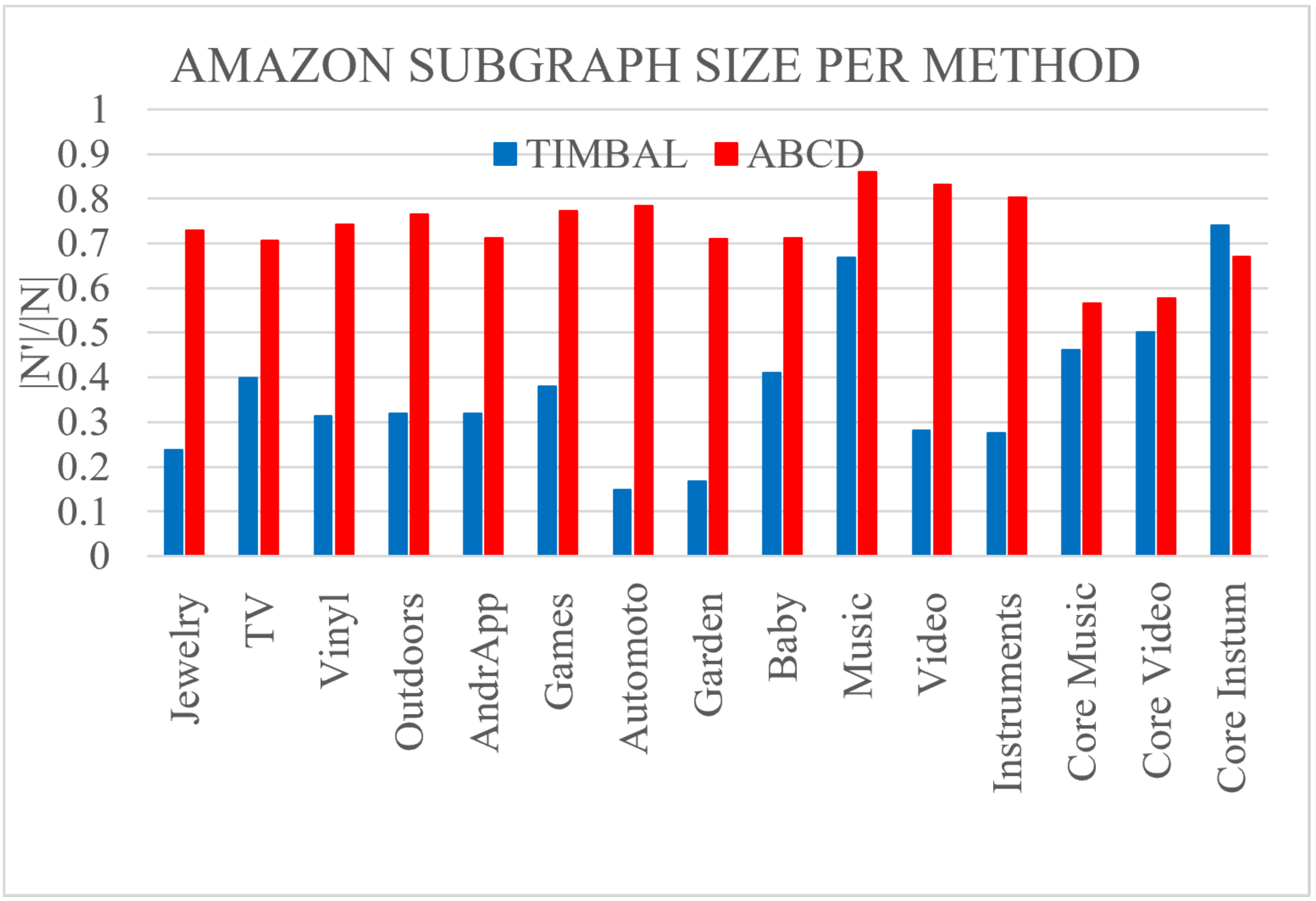

Fig. 7. ABCD and TIMBAL performance comparison for Amazon data.

TABLE 3
Amazon ratings and reviews [16] (a subset of Amazon ratings constructed from core5 reviews) [16] mapped to signed graphs. The number of vertices, edges, and cycles reflect the number in the largest connected component of each dataset.

Amazon	**Input graph**	**Largest Connected Component**		
Ratings	**# ratings**	**# vertices**	**# edges**	**# cycles**
Books	22,507,155	9,973,735	22,268,630	12,294,896
Electronics	7,824,482	4,523,296	7,734,582	3,211,287
Jewelry	5,748,920	3,796,967	5,484,633	1,687,667
TV	4,607,047	2,236,744	4,573,784	2,337,041
Vinyl	3,749,004	1,959,693	3,684,143	1,724,451
Outdoors	3,268,695	2,147,848	3,075,419	927,572
AndrApp	2,638,172	1,373,018	2,631,009	1,257,992
Games	2,252,771	1,489,764	2,142,593	652,830
Automoto	1,373,768	950,831	1,239,450	288,620
Garden	993,490	735,815	939,679	203,865
Baby	915,446	559,040	892,231	333,192
Music	836,006	525,522	702,584	177,063
Video	583,993	433,702	572,834	139,133
Instruments	500,176	355,507	457,140	101,634
Reviews	**# reviews**	**# vertices**	**# edges**	**# cycles**
Core Music	64,706	9,109	64,706	55,598
Core Video	37,126	6,815	37,126	30,312
Core Instrum	10,621	2,329	10,261	7,933

TABLE 4
Amazon ratings and reviews graph results. The time in seconds for ABCD includes 5000 iterations. The time in seconds for TIMBAL is the total time. For graphs over a million vertices, we had only data on one run as it takes over 100 minutes per run for TIMBAL. N/A indicates that a method does not terminate within two days.

Amazon	**TIMBAL**		**ABCD**	
Ratings	# vertices	time s	# vertices	time s
Book	N/A	N/A	**7,085,285**	116897
Electronics	N/A	N/A	**3,104,399**	37677
Jewelry	530,363	47046.34	**2,769,431**	21468
TV	891,106	11379.43	**1,579,760**	17146
Vinyl	612,700	11529.94	**1,452,496**	13011
Outdoors	683,846	12717.01	**1,640,544**	11295
AndrApp	437,740	5052.82	**977,536**	12254
Games	565,301	6251	**1,150,782**	7617.5
Automoto	140,711	12989	**744,474**	4157
Garden	122,844	5204	**522,340**	3200
Baby	229,545	3591	**397,940**	2986
Music	351,124	3223	**451,320**	2203
Video	121,694	4280	**360,665**	2176
Instruments	97,486	1785	**285,233**	1464.8
Reviews	# vertices	time s	# vertices	time s
Core Music 5	4,193	30.3	**5,143**	200.4
Core Video 5	3,419	23.7	**3,934**	128.3
Core Instrum 5	**1,725**	19.1	1,559	36.9

is balanced as it has no unbalanced cycles, and it is a subgraph as the algorithm removes the *vertices*. ABCD recovers significantly balanced subgraphs, over two times larger than

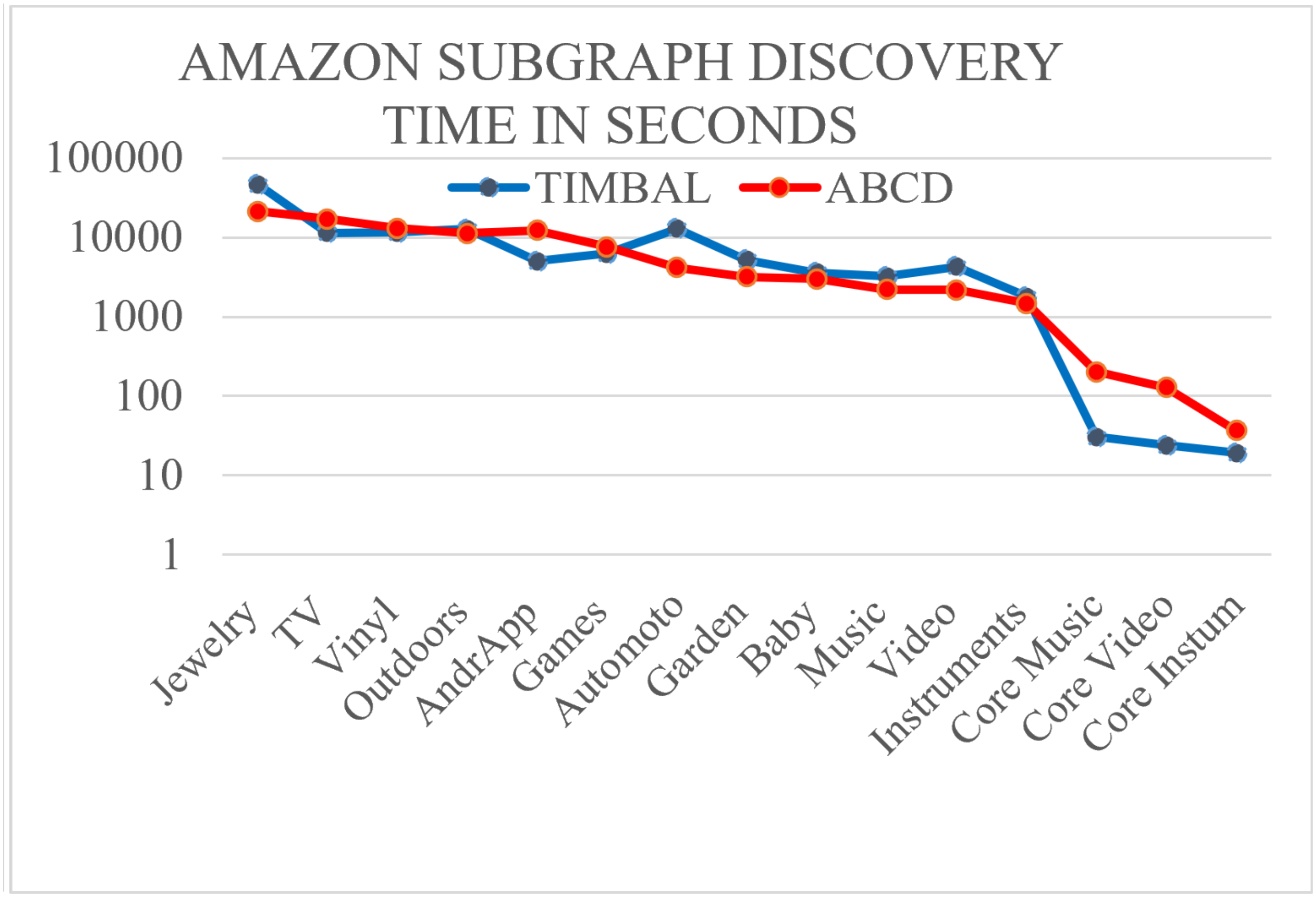

Fig. 8. ABCD and TIMBAL running time comparison for Amazon data.

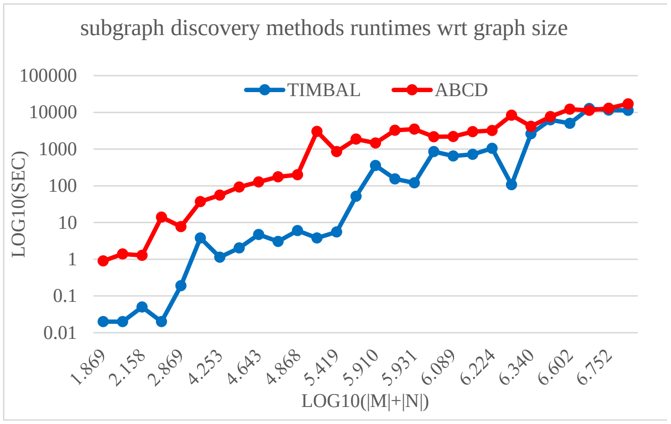

Fig. 9. ABCD and TIMBAL algorithm is linear with the graph size m + n for 28 signed graphs tested. TIMBAL failed to complete the largest two graphs.

the state of the art, while keeping the processing time linear with the size of the graph.

Recently, [30] proposed faster O(m) heuristic and efficient implementation for balancing a graph and for typically obtaining a lower number of flips to reach consensus. Their paper suggests that edges, regardless of whether they belong to the spanning tree, can be chosen to be switched for balance. We plan to investigate this next, as multiple unbalanced fundamental cycles can share an edge in the spanning edge. Changing the sign of a tree edge might cause processing instabilities. Future work also includes integrating the OpenMP and GPU code accelerations. [24] has shown that GPU code takes less than 15 minutes to find 1000 fundamental cycle bases for 10M vertices and 22M edges. Since the runtime is roughly proportional to the input size, the ABCD parallel implementation can balance ten times larger inputs in a few seconds per sample, making it tractable to analyze graphs with 100s of millions of vertices and edges.

References

[1] Y. Wu, D. Meng, and Z.-G. Wu, "Disagreement and antagonism in signed networks: A survey." *IEEE/CAA Journal of Automatica Sinica, Automatica Sinica, IEEE/CAA Journal of, IEEE/CAA J. Autom. Sinica*, vol. 9, no. 7, pp. 1166 – 1187, 2022.

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

[3] F. Heider, "Attitudes and cognitive organization," *J. Psychology*, vol. 21, pp. 107–112, 1946.

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

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

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

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

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

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

[10] F. Harary, M.-H. Lim, and D. C. Wunsch, "Signed graphs for portfolio analysis in risk management," *IMA Journal of Management Mathematics*, vol. 13, no. 3, pp. 201–210, 2002.

[11] R. Figueiredo and Y. Frota, "The maximum balanced subgraph of a signed graph: Applications and solution approaches," *European Journal of Operational Research*, vol. 236, no. 2, pp. 473–487, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221713010205

[12] K. T. Macon, P. J. Mucha, and M. A. Porter, "Community structure in the united nations general assembly," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 1, pp. 343–361, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437111004778

[13] C. Liu, Y. Dai, K. Yu, and Z. Zhang, "Enhancing cancer driver gene prediction by protein-protein interaction network." *IEEE/ACM Transactions on Computational Biology and Bioinformatics, Computational Biology and Bioinformatics, IEEE/ACM Transactions on, IEEE/ACM Trans. Comput. Biol. and Bioinf*, vol. 19, no. 4, pp. 2231 – 2240, 2022.

[14] C. Chen, Y. Wu, R. Sun, and X. Wang, "Maximum signed θ-clique identification in large signed graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1791–1802, 2023.

[15] T. Zaslavsky, "A mathematical bibliography of signed and gain graphs and allied areas." *ELECTRONIC JOURNAL OF COMBINATORICS*, 2012. [Online]. Available: https://libproxy.txstate.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=000209504400001&site=eds-live&scope=site

[16] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. ACM, 2016, pp. 507–517.

[17] B. Ordozgoiti, A. Matakos, and A. Gionis, "Finding large balanced subgraphs in signed networks," in *Proceedings of The Web Conference 2020*, ser. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1378–1388. [Online]. Available: https://doi.org/10.1145/3366423.3380212

[18] K. Sharma, I. A. Gillani, S. Medya, S. Ranu, and A. Bagchi, *Balance Maximization in Signed Networks via Edge Deletions*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 752–760. [Online]. Available: https://doi.org/10.1145/3437963.3441778

[19] N. Gülpinar, G. Gutin, G. Mitra, and A. Zverovitch, "Extracting pure network submatrices in linear programs using signed graphs," *Discrete Applied Mathematics*, vol. 137, no. 3, pp. 359–372, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166218X03003615

[20] S. Poljak and D. Turzík, "A polynomial-time heuristic for certain subgraph optimization problems with guaranteed worst-case bound," *Discrete Mathematics*, vol. 58, no. 1, pp. 99–104, 1986. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0012365X86901925

[21] R. M. Figueiredo, M. Labbé, and C. C. de Souza, "An exact approach to the the problem of extracting an embedded network matrix," *Computers and Operations Research*, vol. 38, no. 11, pp. 1483 – 1492, 2011.

[22] F. Bonchi, E. Galimberti, A. Gionis, B. Ordozgoiti, and G. Ruffo, "Discovering polarized communities in signed networks," 2019.

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

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

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

[26] *Muhieddine Shebaro* and **Jelena Tešić**, "Identifying stable states of large signed graphs," in *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion)*, 2023.

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

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

[29] Y. He, G. Reinert, S. Wang, and M. Cucuringu, "SSSNET: semi-supervised signed network clustering," in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, 2021, pp. 244–252.

[30] S. Kundu and A. A. Nanavati, "A more powerful heuristic for balancing an unbalanced graph," in *Complex Networks and Their Applications XI*, H. Cherifi, R. N. Mantegna, L. M. Rocha, C. Cherifi, and S. Micciche, Eds. Cham: Springer International Publishing, 2023, pp. 31–42.

Muhieddine Shebaro is a Computer Science Ph.D. student at Texas State University. He received his B.S.c degree in Computer Science from Beirut Arab University, Lebanon, in 2021. His research interests include network science, machine learning, and data science.

Jelena Tešić, Ph.D. is an Assistant Professor at Texas State University. Before that, she was a research scientist at Mayachitra (CA) and IBM Watson Research Center (NY). She received her Ph.D. (2004) and M.Sc. (1999) in Electrical and Computer Engineering from the University of California Santa Barbara, CA, USA, and Dipl. Ing. (1998) in Electrical Engineering from the University of Belgrade, Serbia. Dr. Tešić served as Area Chair for ACM Multimedia 2019-present and IEEE ICIP and ICME conferences; she has served as Guest Editor for IEEE Multimedia Magazine for the September 2008 issue and as a reviewer for numerous IEEE and ACM Journals. She has authored over 40 peer-reviewed scientific papers and holds six US patents. Her research focuses on advancing the analytic application of EO remote sensing, namely object localization and identification at scale.