ABCD: Algorithm for Balanced Component Discovery in Signed Networks

Anonymous Author(s)

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 nodes 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 nodes discarded. We evaluate the proposed approach against state-of-the-art and demonstrate over two times higher graph size regarding the number of nodes selected of the discovered subset on an extensive signed network with millions of nodes and edges over the state-of-art in the same time frame.

KEYWORDS

Signed Network, balance theory, balanced sub-graph

ACM Reference Format:

Anonymous Author(s). 2023. ABCD: Algorithm for Balanced Component Discovery in Signed Networks. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graphs are widely used to model complex systems in various fields, such as computer science, physics, biology, and social sciences. In contrast to unsigned networks, where all edges are assumed to have positive weights, signed networks allow for negative weights, which may represent antagonistic relationships or conflicting opinions [26]. The largest balanced element in signed graphs plays a crucial role in aiding researchers to comprehend the fundamental structure of the graph as it reveals valuable information about the complex relationships between nodes in the network. For instance, the nodes that are part of the maximum balanced sub-graphs are the key players in the system. These nodes may not necessarily have a high degree of centrality between them, but they are essential for understanding how the system behaves. We can gain insight into the mechanisms that drive the system's behavior by identifying these nodes. In addition, it cannot be easy to understand the complex system's behavior. By locating the maximum balanced sub-graphs, we can simplify the system into sub-systems with balanced interactions and eliminate inconsistencies regarding unbalanced cycles. The task of the largest balanced sub-graph discovery has applications in portfolio system's economic risk management [10], computational and operational research [7], community analysis and structure [17], computational biology to model balanced interactions between genes and proteins [16] and social network analysis [6]. **Graph** G is an (N,M) set tuple, where N is a set of all nodes of size $|N$ and M is a set of all edges of size $|M|$.

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 final output of the "algorithm. The system's overall complexity is $O(|N|^3)$ if N is a set of nodes [9]. 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[9]. 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 [8], followed by pre-processing routines and initial heuristic improvements in [7]. 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 [7]. The EIGEN algorithm [2] 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 [23]. 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 [19]. Both algorithms rely on signed spectral theory. The approaches do not scale to the large signed graphs as they rely on

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

the costly eigenvalue computation ($O(|N|^2)$), and its performance decreases due to the spectral pollution in eigenvalue computation [3]. TIMBAL proposes a novel bound for perturbations of the graph Laplacian pre-processing and sub-sampling 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 [19].

This paper proposes an algorithm for balanced consensus 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 [1] and utilizes the graph's node density distribution and near-optimal balanced states to minimize the number of nodes removed from the balanced sub-graph. The paper is organized as follows: in Section 2, we present preliminary findings; in Section 3, we introduce the ABCD algorithm; in Section 4, we elaborate on the implementation details of the proposed algorithm and evaluation; Section 5, we conduct experiments to compute the number of nodes of the maximum balanced sub-graph of 12 different real-world datasets for our algorithm and compare it to the baseline; in Section 6 we summarize our findings.

2 FUNDAMENTAL CYCLE BASIS

In this section, we introduce the concept of a fundamental cycle and present an efficient way to compute fundamental cycles in any graph (computational capabilities still limit it). *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. In a signed graph Σ, a *path* is a sequence of distinct edges that connect a sequence of distinct vertices. *Cycle* is a path that begins and ends at the same node. *Cycle basis* is a minimal set of cycles that allows every even-degree sub-graph to be expressed as a symmetric difference of basis cycles. *Connected graph* is a graph in which a path joins two vertices. *sub-graph* is a graph with all edges and nodes in a larger graph. *Balanced signed graph* is a signed graph where every cycle is positive. *Harary bipartition* separates the nodes of the balanced graph into two sets such that the nodes of both sets internally agree with each other but disagree with the nodes of the other set[5].

Fundamental cycle is a type of cycle sub-graph in a graph created in a specific manner. For the underlying graph G, let T be the spanning tree of G, and let an edge m be an edge in G between nodes x and y that is *NOT* in the spanning tree T. Since the spanning tree spans all vertices, a unique path in T between nodes x and y does not include m. Therefore, that path in T plus m for a *cycle* in graph G. A cycle built this way is called a *fundamental cycle*. A fundamental cycle basis may be formed from a spanning tree or spanning forest of the given graph by selecting the cycles formed by combining a path in the tree and a single edge outside the tree. For the graph G with $|N|$ nodes and $|M|$ edges, there are exactly $|M| - |N| + 1$ fundamental cycles. It has been shown that using this spanning tree-based approach to discover fundamental cycles and balance the graph requires a minimum number of edge sign switches to produce a balanced graph from the signed graph Σ [22]. The efficient way to discover fundamental cycles is given the spanning tree T [1]. The main benefits of the approach are that labels can be computed with linear time complexity, only require a linear amount of memory, and that the running time for balancing a cycle is linear in the length of the cycle times the vertex degrees but *independent* of the size of the graph.

3 ABCD: ALGORITHM FOR BALANCED COMPONENT DISCOVERY

Algorithm 1: Fundamental Cycle Bases with the smallest number of imbalanced fundamental cycles

Data: Signed graph Σ; I K
Result: $\mathcal{M}_\Sigma = M_k, \mathcal{H}_\Sigma = H_k, k \in [1, K]$
Generate set $\mathcal{T}_i$ of I spanning trees of Σ;
Counter to keep only the top nearest balanced states
  $i = 0; m_K = m;$;
for *i =0; i++; i < I* **do**
  forall *edges m,* $m \in \Sigma \setminus T_i$ **do**
    if *fundamental cycle* $T \cup m$ *is negative* **then**
      add edge m to M_i
  end
  if $|M_i| < m_K$ **then**
    $\mathcal{M}_\Sigma \cup = M_k$
  if $|\mathcal{M}_\Sigma| > K$ **then**
    Remove the largest set and update m_K ;
end
for *i =0; i++; i < K* **do**
  Create zero vector H_k of dimension n;
  for *edge* $m \in M_i$ **do**
    switch edge sign in Σ_k: $m^- \rightarrow m^+; m^+ \rightarrow m^-$;
  end
  Cut all the negative edges to create Harary bi-partitions A and B so that $|A| > |B|$;
  for *n in N* **do**
    if $n \in A, H_k(n) = 1$
  end
end

Goal is to identify a sub-graph Σ', $\Sigma' \subset \Sigma$ induced by a set of nodes N', $N' \subset N$. N is a set of all nodes in graph Σ, Σ' graph is balanced and the size of N' set is largest possible. This section introduces the Algorithm for the Balanced Component Discovery (ABCD) as a scalable solution for this task, as explained in the following three steps. We discover the fundamental cycle bases out of I sampled bases in the first step. We keep only the K fundamental cycle basis with the smallest number of imbalanced fundamental cycles, as outlined in Alg. 1. The backbone of the proposed algorithm is the fast discovery of fundamental cycles, as introduced in [1] and published in [4]. We integrate the algorithm to identify the fundamental cycle basis for a given spanning tree T. Note that even though we ran the algorithm $I >> K$ times as outlined in

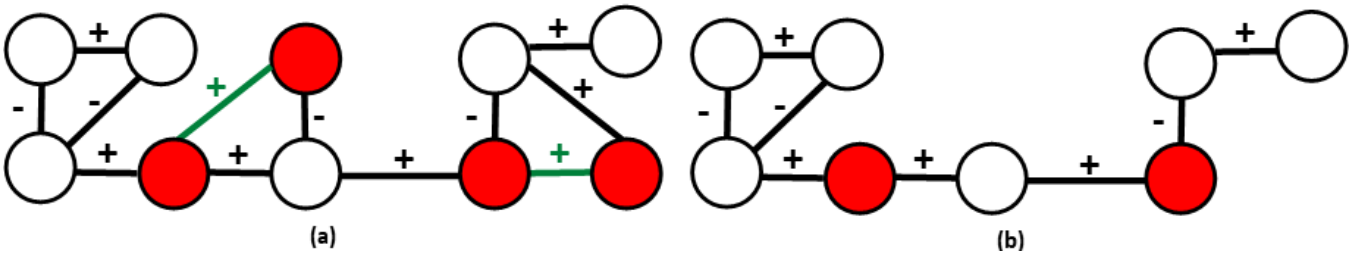

Figure 1: (a): The unbalanced signed network. Green edges are the candidate edges causing imbalance, and red nodes are the candidate vertices. (b): The maximum balanced signed sub-graph obtained after deleting one candidate node along each edge.

[22], we only save the information on the unbalanced fundamental cycles and the Harary bipartition labeling for the top K unique fundamental cycle bases with the lowest cardinality $|M_k|, |M_k| < |M| - |N| + 1$, as outlined in Alg. 1.

Algorithm 2: Computation of the sum of neighborhood degree

```
Data: Signed graph Σ;
Result: Array neighborhood_degree
Declare and initialize array neighborhood_degree = [];
for v = 0; v++; v < |N| of Σ do
    Declare and initialize integer sum = 0;
    for every neighbor nei connected to v via an edge do
        sum += degree[nei];
    end
    neighborhood_degree[v]=sum;
end
```

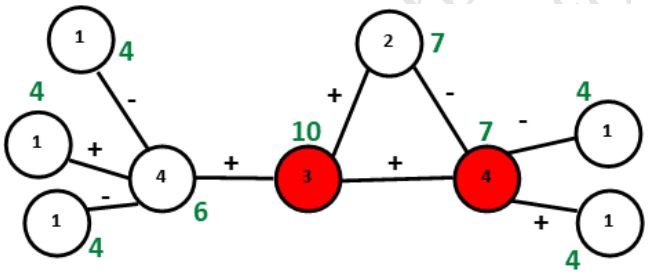

Figure 2: Degree (black, in node) vs. Sum of Neighborhood Degrees (green, next to the node) computation. The sum of neighborhood degrees labels the red nodes connected by a positive link that will be deleted.

In the **second step**, the ABCD algorithm employs a smart edge deletion heuristic for each of the one of K discovered balanced states as it removes all unbalanced fundamental cycles. Our heuristic relies on the notion that minimizing the number of nodes removed from the graph increases the chance that the largest balanced sub-graph has its node cardinality maximized. We have a list of M_k edges that should have a different sign for the entire graph to be balanced, and we try to minimize the number of edges removed from the graph, as outlined below. 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 $k, k \in [1, K]$ is as follows. First, initialize an empty set of nodes N_k. Second, for every edge m in M_k created in Alg. 1 repeat the following:

(1) If the edge m is positive, it will be negative in the balanced state. If either of the nodes is already in N_k, move on to the next edge. Else, add the edge-defining node n 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. If no resolution is offered by N_k by the end of the process, randomly pick a node. *Harary bipartition* separates the nodes of the balanced graph into two sets such that the nodes of both sets internally agree with each other but disagree with the nodes of the other set[5]. As a labeling vector, the H_k set is created in Alg. 1. The logic behind it is that the remaining node is in the largest Harary partition for a fully balanced graph so that it will be connected to more nodes than the node ending up in the smaller Harary set after partitioning.

(2) If the edge m is negative and marked for switching to positive: if either of the nodes is already in N_k, nothing needs to be done; move on to the next edge. We add the node with the lower neighborhood degree to N_k.

The example of how we compute the neighborhood degree for signed vector Σ is illustrated in Fig. 2. Two red nodes 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. However, the neighborhood degree of the left node is 10 (neighbors of a neighbor), and the neighborhood degree of the right node is 7. Thus, 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 nodes in Σ once and re-used in this step. The algorithm for the computation of the sum of neighborhood degrees is highlighted in Algorithm 2.

In the **third step**, we find the index of the smallest sized N_k set among all $N_k, k \in [1, K]$ sets. Let it be index $s : |N_s| \leq |N_k|, k \in$

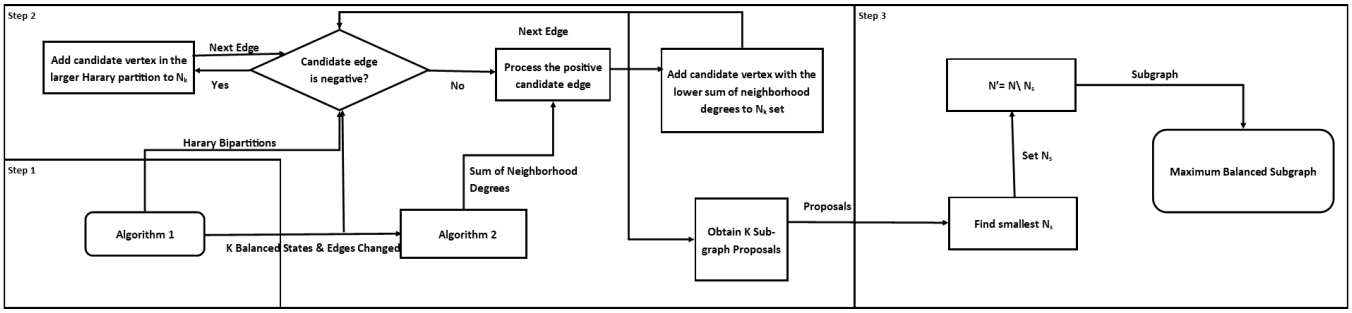

Figure 3: The ABCD pipeline.

$[1, K]$. The resulting maximized balanced sub-graph proposal N'_s is finally obtained by removing specific nodes as $N' = N \setminus N_s$. We know the remaining subset is balanced as all fundamental cycles in the remaining graph are balanced.

The following is a set of observations that have helped refine and shape our method as well as establish an underpinning on which our algorithm stands:

Observation 1: A balanced state with the lowest frustration index for a specific signed network does not necessarily yield a maximum balanced sub-graph using the set of conditions we laid out for processing positive and negative candidate edges (relying on Harary bipartitions for positive edges and the sum of neighborhood degrees for negative edges for the decision-making for vertex deletion). However, 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. Hence, we only capture the information on the unbalanced fundamental cycles and the Harary bipartition labeling for the top K unique fundamental cycle bases with the lowest cardinality.

Observation 2: Among the tree-sampling techniques such as depth-first search, randomized depth-first search, and breadth-first search, the latter is the most suitable for this kind of task because it relatively reduces the amount of identified candidate edges compared to others.

Observation 3: The largest connected component (LCC) produced at the end is *guaranteed* to be balanced. This algorithm eliminates the unbalanced cycle bases by leaving a "hole," which prevents any consensus conflict. Therefore, this component is then called the maximum (it has the largest size in terms of the number of vertices) balanced (it has no unbalanced cycles) sub-graph (the number of nodes in this component is less than or equal to that of the entire signed network input) where edge sign switching is absent.

4 METHODOLOGY

Implementation The entire code is implemented in C++. The implementation first identifies the largest connected component (LCC) in the supplied network and only applies the ABCD algorithm to that component. All benchmark graphs have one large connected component, so the findings do not take away from the generalizability of the method. The implementation treats edges without signs as positive edges in the fundamental cycle. If the graph has more equal connected components, the implementation can be easily extended to accommodate that scenario. Step one of the code implementation of the algorithm presented in Alg. 1 builds on fundamental cycle discovery code release [4]. Recent findings have shown that the breadth-first search approach for sampling the spanning tree from the graph was diverse enough. Its implementation was faster than the adapted version for other samplers of a given signed graph Σ [24], and we adopt the breath-first search method for sampling spanning trees in the Algorithm 1. Iterations are set to 5000 when running ABCD on all signed networks. For small signed networks (Highland, CrisisInCloister, ProLeague, DutchCollege, and Congress), K is set to be high $K = 4000$ as processing is fast. We set $K = 100$ for larger Konect and Amazon datasets with under 300,000 nodes. For large signed graphs derived from Amazon ratings, K is set to 20. We implement the rest of the algorithm as outlined in Section 3. In the third step, N'_s is constructed by the algorithm re-reading the original graph and skipping the entries with nodes in N_s. The code and data are released in the anonymous GitHub repository for review [21].

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 pre-processing techniques such as removing duplicate and inconsistent edges are 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

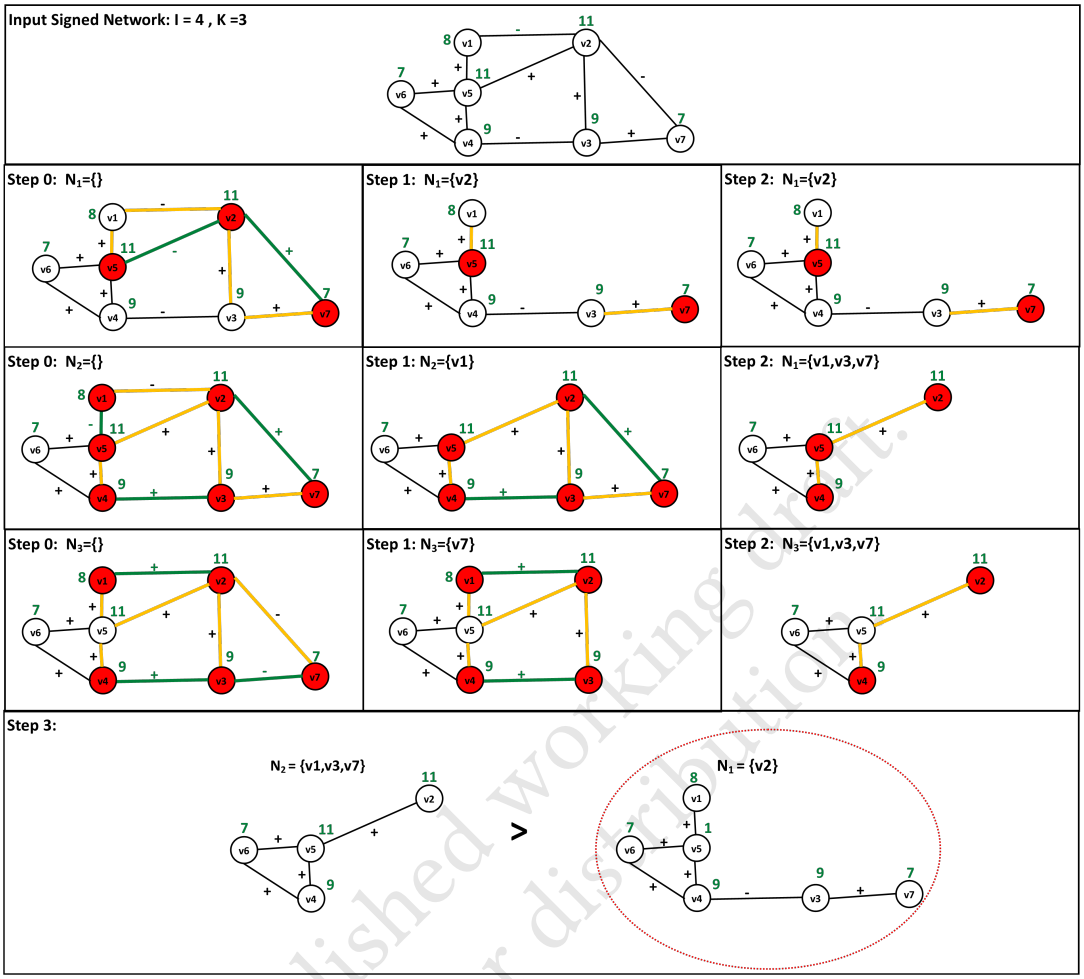

Figure 4: The ABCD algorithm. Balancing occurs in Step 0 with green text on the edges with the changed signs. Green numbers beside nodes are the sum of neighborhood degrees. Orange edges are the edges of the unbalanced fundamental cycles. Green edges are the candidate edges. Red nodes are the candidate nodes for deletion. The red dotted oval over the graph in Step 3 signifies that the final output N_s' of ABCD has a maximal cardinality.

Datasets We evaluate the proposed ABCD algorithm on two signed graph benchmarks. The first set comprises 14 signed graphs from the Konect repository [14] used in [19] (except TwitterReferendum and PPI, which are not from the Konect site). The second set is the set of 17 Amazon rating files [11] also used in [1]. **Konect Signed Graphs** 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 [14]. *CrisisInCloister* is a directed network that contains ratings between monks related to a crisis in a cloister (or monastery) in New England (USA) which led to the departure of several of the monks [14]. *ProLeague* are results of football games in Belgium from the Pro League in 2016/2017 in the form of a directed signed graph. Nodes 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 [14]. *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 [14]. *Congress* is a signed network where nodes are politicians speaking in the United States Congress, and a directed edge denotes that a speaker mentions another speaker [14]. 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 [14]. *BitcoinAlpha* is a user-user trust/distrust network from the Bitcoin Alpha platform on which Bitcoins are traded [14].*BitcoinOTC* is a user-user trust/distrust network, from the Bitcoin OTC platform,

Table 2: Amazon ratings and reviews (a subset of Amazon ratings constructed from core5 reviews) [11] 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**	**# nodes**	**# 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
Clothing, Shoes, and Jewelry	5,748,920	3,796,967	5,484,633	1,687,667
Movies and TV	4,607,047	2,236,744	4,573,784	2,337,041
CDs and Vinyl	3,749,004	1,959,693	3,684,143	1,724,451
Sports and Outdoors	3,268,695	2,147,848	3,075,419	927,572
Android App	2,638,172	1,373,018	2,631,009	1,257,992
Toys and Games	2,252,771	1,489,764	2,142,593	652,830
Automotive	1,373,768	950,831	1,239,450	288,620
Patio, Lawn, and Garden	993,490	735,815	939,679	203,865
Baby	915,446	559,040	892,231	333,192
Digital Music	836,006	525,522	702,584	177,063
Instant Video	583,993	433,702	572,834	139,133
Musical Instruments	500,176	355,507	457,140	101,634
Reviews	**# reviews**	**# nodes**	**# edges**	**# cycles**
Digital Music core5	64,706	9,109	64,706	55,598
Instant Video core5	37,126	6,815	37,126	30,312
Musical Instruments core5	10,621	2,329	10,261	7,933

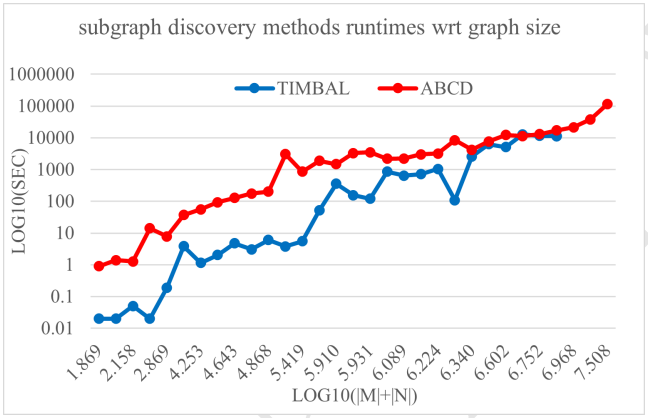

Figure 5: TIMBAL algorithm and ABCD algorithm are linear with the graph size |M| + |N| for all signed graphs tested.

on which Bitcoins are traded [14]. *TwitterReferendum* captures data from Twitter concerning the 2016 Italian Referendum. Different stances between users signify a negative tie, while the same stances indicate a positive link [15]. *WikiElec* is the network of users from the English Wikipedia that voted for and against each other in admin elections [14]. *Slashdot* is the reply network of the technology website Slashdot. Nodes are users, and edges are replies [14]. The edges of *WikiConflict* represent positive and negative conflicts between users of the English Wikipedia [14]. *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 [14]. *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 [14].[12]. The characteristics of the signed graphs are listed in Table 1.

5 RESULTS

Amazon Signed Graphs benchmark consists of 17 signed graphs derived from the Amazon rating and review files [11]. The dataset contains product reviews and metadata from Amazon, spanning May 1996 - July 2014. We use the Amazon rating dataset as (user, item, rating, timestamp) tuples for these experiments. We also use a smaller version of Digital Music (64,706 reviews), Musical Instruments (10,261 reviews), and Amazon Instant Video (37,126 reviews) ratings, where there are five or more reviews provided per product. 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 $(2, 1) \rightarrow m^-$ [11]. The summary of the datasets is outlined in Table 2.

Setup TIMBAL [19] approach has been shown to reach the highest cardinality of the sub-graphs in various datasets and is a de-facto state of the art [19]. 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. The input parameters of TIMBAL [18] 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 such as Highland, CrisisInCloister, ProLeague, DutchLeague, and Congress, and it is set to max_removals=100 for the rest of the signed networks. We set avg_size=20 for datasets of several nodes less than 80,000, and subsample_flag=True, samples = 1000, avg_size = 200 max_removals=100 for datasets with the number of nodes greater than 80,000. TIMBAL is a non-deterministic

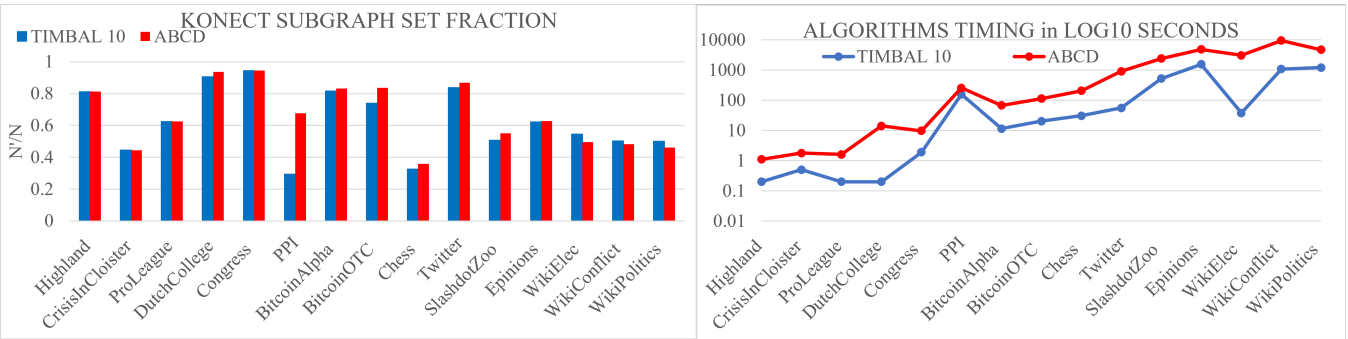

Figure 6: ABCD and TIMBAL performance comparison from Table 3). The largest connected component of Konect graphs is close to the size of the graph, and both methods retrieve similarly size portions of the graphs. ABCD performs better for seven graphs, TIMBAL for three wiki graphs, and they score equally on three smaller graphs.

Table 3: [14] Konect (except TwitterReferendum and PPI, which are not from the Konect site) graph performance comparisons: The largest sub-graph regarding the number of nodes discovered for TIMBAL 5 runs, TIMBAL 10 runs, and ABCD 5000 iterations approach in a given runtime in seconds.

Konect	TIMBAL 5 runs		TIMBAL 10 runs		ABCD	
Datasets	sub-graph	Run	sub-graph	Run	sub-graph	Run
	# vertices	time (s)	# vertices	time (s)	# vertices	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

algorithm, and we run it 5 and 10 times for Konect data to get the maximum node cardinality. All ABCD parameters are fixed, as described in the implementation section. The operating system used for the experiments is Linux Ubuntu 20.04.3, running on the 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz with 16 physical cores. It has one socket, two threads per core, and eight cores per socket. The architecture is X86_x64. GPU is Nvidia GeForce having 8GB memory. Its driver version is 495.29.05, and the CUDA version is 11.5. The cache configuration is L1d : 384 KiB, L1i : 256 KiB, L2 : 4 MiB, L3 : 16 MiB. The CPU op is 32-bit and 64-bit.

Konect benchmark consists of 14 Konect datasets, and TIMBAL and ABCD performance are outlined in Table 3 ABCD matches TIMBAL performance in the smallest three networks and *Congress* data and takes a slightly longer time. ABCD algorithm finds a more significant subset for relatively small *DutchCollege*, BitcoinAlpha, BitcoinOTC, Chess, TwitterReferendum [15], SlashdotZoo, PPI [25], and Epinions [14]. TIMBAL performs better on the three Konect Wiki data sets. While Timbal is significantly faster than ABCD on smaller networks, it is only double the speed for the Epinions network of 119,000+ nodes, and ABCD recovers the largest balanced sub-graph. The comparison graphs are outlined in Figure 6(right). We recorded the maximum number of vertices obtained after 5 and 10 runs for TIMBAL, and only for one dataset did the repeated runs discover a larger subset.

A single TIMBAL run takes as long as ABCD for bipartite Amazon graphs. Moreover, ABCD has a superior runtime and performance regarding the graph size it discovers, as illustrated in Table 4. TIMBAL's performance degrades with the graph size, and the discovered sub-graphs are much smaller than what ABCD finds; see Figure 7(left) for comparison. Figure 5 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 have approximately

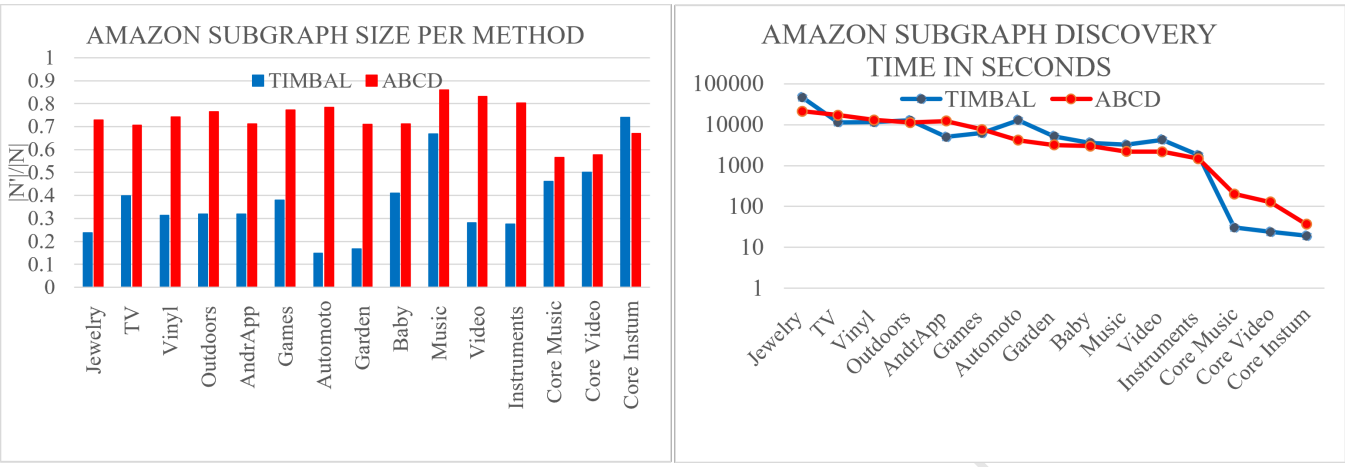

Figure 7: ABCD and TIMBAL performance comparison from Table 4). Left: the sub-graph size TIMBAL recovers (blue box) is over two times smaller than the sub-graph ABCD algorithm recovers (red box). Right: one iteration of TIMBAL (blue line) takes as long as the entire ABCD algorithm (red line) for larger graphs.

Table 4: Amazon ratings and reviews graph characteristics [11]. The number of vertices, edges, and cycles reflect the number in the largest connected component of each dataset. The time in seconds for TIMBAL is the total time. For graphs over million nodes, we had only data on one run as it takes over 100 minutes per run for TIMBAL. The time in seconds for ABCD includes 5000 iterations. N/A indicates that a method does not terminate within two days.

Amazon			**TIMBAL**			**ABCD**	
Ratings	|N|	|M|	# nodes	time s	#runs	# nodes	time s
Books	9,973,735	22,268,630	N/A	N/A	1	**7,085,285**	116897
Electronics	4,523,296	7,734,582	N/A	N/A	1	**3,104,399**	37677
Jewelry	3,796,967	5,484,633	530,363	47046.34	1	**2,769,431**	21468
TV	2,236,744	4,573,784	891,106	11379.43	1	**1,579,760**	17146
Vinyl	1,959,693	3,684,143	612,700	11529.94	1	**1,452,496**	13011
Outdoors	2,147,848	3,075,419	683,846	12717.01	1	**1,640,544**	11295
Android App	1,373,018	2,631,009	437,740	5052.82	1	**977,536**	12254
Games	1,489,764	2,142,593	565,301	6251	1	**1,150,782**	7617.5
Automotive	950,831	1,239,450	140,711	12989	5	**744,474**	4157
Garden	735,815	939,679	122,844	5204	5	**522,340**	3200
Baby	559,040	892,231	229,545	3591	5	**397,940**	2986
Digital Music	525,522	702,584	351,124	3223	5	**451,320**	2203
Instant Video	433,702	572,834	121,694	4280	5	**360,665**	2176
Musical Instruments	355,507	457,140	97,486	1785	5	**285,233**	1464.8
Reviews	|N|	|M|	# nodes	time s	#runs	# nodes	time s
Digital Music	9,109	64,706	4,193	30.3	5	**5,143**	200.4
Instant Video	6,815	37,126	3,419	23.7	5	**3,934**	128.3
Musical Instruments	2,329	10,261	**1,725**	19.1	5	1,559	36.9

run times 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 nodes.

6 CONCLUSION AND FUTURE WORK

We propose a novel scalable algorithm for balance component discovery (ABCD). We demonstrate that it recovers much more significant (over two times) balanced sub-graphs than state-of-art for large signed graph networks while keeping the processing time linear with the size of the graph. 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. In this paper, we proposed a novel framework for this problem to improve the results of the latest baseline on 90% real-world signed networks we have tested in terms of runtime scale and the size of sub-graphs in terms of the number of nodes included. Recently, Kundu et al. proposed faster O(|M|) heuristic and efficient implementation for balancing a graph and for typically obtaining a lower number of flips to reach consensus [13] than [1]. 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. Next, even on the most significant input with 10M nodes and 22M edges, the GPU code takes less than 15 minutes to compute 1000 nearest balanced states, i.e., under 1 second per sample [1]. Since the runtime is roughly proportional to the input size, the ABCD implementation should be able to balance ten times larger inputs in a few seconds per sample, making it tractable to analyze graphs with 100s of millions of nodes and edges. Future work includes integrating the OpenMP and GPU code accelerations.

REFERENCES

[1] Ghadeer Alabandi, Jelena Tešić, Lucas Rusnak, and Martin Burtscher. 2021. Discovering and Balancing Fundamental Cycles in Large Signed Graphs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 68, 17 pages. https://doi.org/10.1145/3458817.3476153

[2] Francesco Bonchi, Edoardo Galimberti, Aristides Gionis, Bruno Ordozgoiti, and Giancarlo Ruffo. 2019. Discovering Polarized Communities in Signed Networks. arXiv:1910.02438 [cs.DS]

[3] Lyonell Boulton. 2016. Spectral pollution and eigenvalue bounds. *Applied Numerical Mathematics* 99 (2016), 1–23. https://doi.org/10.1016/j.apnum.2015.08.007

[4] Martin Burtscher. 2021. graphB+: Balancing Algorithm for Large Graphs. https://userweb.cs.txstate.edu/~burtscher/research/graphB/.

[5] Dorwin Cartwright and Frank Harary. 1956. Structural balance: a generalization of Heider's theory. *Psychological Rev.* 63 (1956), 277–293.

[6] Chen Chen, Yanping Wu, Renjie Sun, and Xiaoyang Wang. 2023. Maximum Signed θ-Clique Identification in Large Signed Graphs. *IEEE Transactions on Knowledge and Data Engineering* 35, 2 (2023), 1791–1802. https://doi.org/10.1109/TKDE.2021.3098423

[7] Rosa Figueiredo and Yuri Frota. 2014. The maximum balanced subgraph of a signed graph: Applications and solution approaches. *European Journal of Operational Research* 236, 2 (2014), 473–487. https://doi.org/10.1016/j.ejor.2013.12.036

[8] Rosa M.V. Figueiredo, Martine LabbÃ©, and Cid C. de Souza. 2011. An exact approach to the problem of extracting an embedded network matrix. *Computers and Operations Research* 38, 11 (2011), 1483 – 1492.

[9] N Gülpinar, G Gutin, G Mitra, and A Zverovitch. 2004. Extracting pure network submatrices in linear programs using signed graphs. *Discrete Applied Mathematics* 137, 3 (2004), 359–372. https://doi.org/10.1016/S0166-218X(03)00361-5

[10] F Harary, M-H Lim, and D C Wunsch. 2002. Signed graphs for portfolio analysis in risk management. *IMA Journal of Management Mathematics* 13, 3 (2002), 201–210. https://doi.org/10.1093/imaman/13.3.201

[11] Ruining He and Julian McAuley. 2016. 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 (WWW '16)*. ACM, 507–517.

[12] Yixuan He, Gesine Reinert, Songchao Wang, and Mihai Cucuringu. 2021. SSSNET: Semi-Supervised Signed Network Clustering. *CoRR* abs/2110.06623 (2021). arXiv:2110.06623 https://arxiv.org/abs/2110.06623

[13] Sukhamay Kundu and Amit A. Nanavati. 2023. A More Powerful Heuristic forÂ Balancing anÂ Unbalanced Graph. In *Complex Networks and Their Applications XI*, Hocine Cherifi, Rosario Nunzio Mantegna, Luis M. Rocha, Chantal Cherifi, and Salvatore Micciche (Eds.). Springer International Publishing, Cham, 31–42.

[14] Jérôme Kunegis. 2013. KONECT – The Koblenz Network Collection. In *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*. ACM, 1343–1350. http://dl.acm.org/citation.cfm?id=2488173

[15] Mirko Lai, Viviana Patti, Giancarlo Ruffo, and Paolo Rosso. 2018. Stance Evolution and Twitter Interactions in an Italian Political Debate. In *Natural Language Processing and Information Systems*, Max Silberztein, Faten Atigui, Elena Kornyshova, Elisabeth Métais, and Farid Meziane (Eds.). Springer International Publishing, Cham, 15–27.

[16] C. Liu, Y. Dai, K. Yu, and Z. Zhang. 2022. 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* 19, 4 (2022), 2231 – 2240.

[17] Kevin T. Macon, Peter J. Mucha, and Mason A. Porter. 2012. Community structure in the United Nations General Assembly. *Physica A: Statistical Mechanics and its Applications* 391, 1 (2012), 343–361. https://doi.org/10.1016/j.physa.2011.06.030

[18] Bruno Ordozgoiti. 2020. Finding large balanced subgraphs in signed networks. https://github.com/justbruno/finding-balanced-subgraphs.

[19] Bruno Ordozgoiti, Antonis Matakos, and Aristides Gionis. 2020. Finding Large Balanced Subgraphs in Signed Networks. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 1378–1388. https://doi.org/10.1145/3366423.3380212

[20] Svatopluk Poljak and Daniel Turzík. 1986. A polynomial time heuristic for certain subgraph optimization problems with guaranteed worst case bound. *Discrete Mathematics* 58, 1 (1986), 99–104. https://doi.org/10.1016/0012-365X(86)90192-5

[21] Anonymous Code Release. 2023. ABCD. https://anonymous.4open.science/r/ABCD-2E9C.

[22] Lucas Rusnak and Jelena Tešić. 2021. Characterizing Attitudinal Network Graphs through Frustration Cloud. *Data Mining and Knowledge Discovery* 6 (November 2021). https://doi.org/10.1007/s10618-021-00795-z

[23] Kartik Sharma, Iqra Altaf Gillani, Sourav Medya, Sayan Ranu, and Amitabha Bagchi. 2021. *Balance Maximization in Signed Networks via Edge Deletions*. Association for Computing Machinery, New York, NY, USA, 752–760. https://doi.org/10.1145/3437963.3441778

[24] Muhieddine Shebaro and Jelena Tešić. 2023. Identifying Stable States of Large Signed Graphs. In *Proceedings of the 32nd International Conference Companion on World Wide Web (WWW '23)*. ACM. https://doi.org/10.1145/3543873.3587544

[25] Arunachalam Vinayagam, Jonathan Zirin, Charles Roesel, Yanhui Hu, Bahar Yilmazel, Anastasia A Samsonova, Ralph A NeumÃ¼ller, Stephanie E Mohr, and Norbert Perrimon. 2014. Integrating protein-protein interaction networks with phenotypes reveals signs of interactions. *Nature Methods* 11, 1 (2014), 94 – 99. https://libproxy.txstate.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=94354747&site=eds-live&scope=site

[26] Yuxin Wu, Deyuan Meng, and Zheng-Guang Wu. 2022. 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* 9, 7 (2022), 1166 – 1187. https://libproxy.txstate.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.9812528&site=eds-live&scope=site