Scalable Algorithm for Finding Balanced Subgraphs with Tolerance in Signed Networks

Jingbang Chen*
j293chen@uwaterloo.ca
David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada

Qiuyang Mang*
qiuyangmang@link.cuhk.edu.cn
School of Data Science, The Chinese University of Hong Kong, Shenzhen
Shenzhen, China

Hangrui Zhou*
zhouhr23@mails.tsinghua.edu.cn
Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University
Beijing, China

Richard Peng
yangp@cs.cmu.edu
Computer Science Department, Carnegie Mellon University
Pittsburgh, USA

Yu Gao
ygao2606@gmail.com
Independent
Beijing, China

Chenhao Ma†
machenhao@cuhk.edu.cn
School of Data Science, The Chinese University of Hong Kong, Shenzhen
Shenzhen, China

ABSTRACT

Signed networks, characterized by edges labeled as either positive or negative, offer nuanced insights into interaction dynamics beyond the capabilities of unsigned graphs. Central to this is the task of identifying the maximum balanced subgraph, crucial for applications like polarized community detection in social networks and portfolio analysis in finance. Traditional models, however, are limited by an assumption of perfect partitioning, which fails to mirror the complexities of real-world data. Addressing this gap, we introduce an innovative generalized balanced subgraph model that incorporates tolerance for imbalance. Our proposed region-based heuristic algorithm, tailored for this **NP**-hard problem, strikes a balance between low time complexity and high-quality outcomes. Comparative experiments validate its superior performance against leading solutions, delivering enhanced effectiveness (notably larger subgraph sizes) and efficiency (achieving up to 100× speedup) in both traditional and generalized contexts.

KEYWORDS

graph mining, signed graph, dense subgraph, community detection

ACM Reference Format:
Jingbang Chen, Qiuyang Mang, Hangrui Zhou, Richard Peng, Yu Gao, and Chenhao Ma. 2024. Scalable Algorithm for Finding Balanced Subgraphs with Tolerance in Signed Networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3637528.3671674

1 INTRODUCTION

Social media platforms, integral to our digital connectivity, transform interactions into analyzable social networks. By deploying graph algorithms, we discern network properties like community detection [20, 36] and partitioning [11], informing user experience improvements and recommendation systems [35]. Yet, these platforms can also engender echo chambers that reinforce divisive ideologies, challenging democratic health. Consequently, detecting and countering polarization in social networks is a critical area of research [9, 31], pivotal for developing defenses against misinformation [8, 14].

A classical model that applies to social networks to deal with polarization is the signed graphs. The signed graph model overcomes the limitation that normal graphs cannot capture users' dispositions. Generally speaking, while the vertex set represents users, there are two kinds of edges between vertices indicating agreement or disagreement. We often refer to them as the positive and the negative edges. The signed graph model was first introduced by Harary in 1953 [22] to study the concept of *balance*. The concept of *balance* is important in signed graphs. Generally speaking, a signed graph is balanced if it can be decomposed into two disjoint sets such that positive edges are between vertices in the same set while negative edges are between vertices from different sets. Such a concept has many practical applications, especially in the polarization study. In a social network, a balanced graph suggests two communities exist with contrasting relationships while maintaining inner cohesion.

There are two lines of work when studying social network polarization with signed graphs. Since most graphs coming from real scenarios are not balanced, if we can find the maximum balanced subgraph (MBS) instead, it usually reveals the largest polarized communities along with several important properties.

On the other line of work, instead of extracting the maximal subgraph, it focuses on removing edges to guarantee the balance of the original graph. The minimum number of edges whose deletion makes all connected components balanced is called the *Frustration Index* of the given signed graph.

However, such a notion of balance is strict in that no edge can disobey the condition. In reality, such strictness does not usually appear. For example, though being dominated, there usually exists some voices of disagreement on the majority idea, even in the most extreme community. Besides, two individuals in different political parties might reach a consensus on certain issues. In other words, there usually exist some *imbalanced* edges in signed graphs extracted from the community, which are against the strict notion of balance. Failing to handle these imbalanced edges might prevent us from identifying, extracting, and characterizing the community

*The first three authors contributed equally to this research.
†Chenhao Ma is the corresponding author.

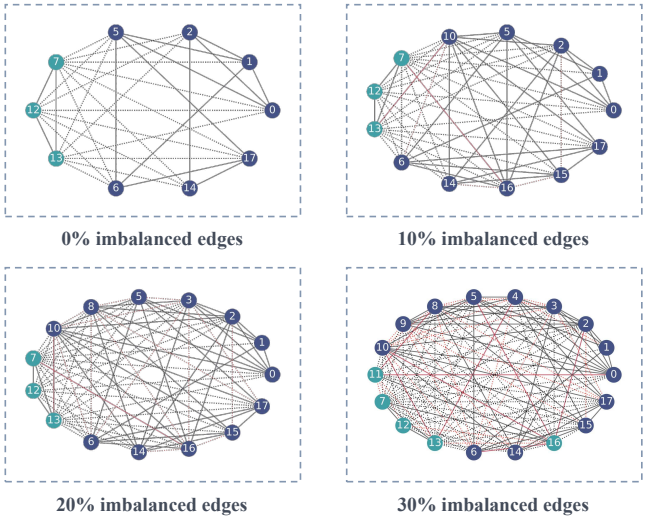

Figure 1: Balanced subgraphs are found in Cloister with different tolerance, where solid edges are positive, dashed edges are negative, black edges are balanced, and red edges are imbalanced.

properly in real scenarios. We provide an example in Figure 1[1]: As we increase the limitation for imbalanced edges, the community is getting significantly larger and denser. Therefore, we may be unable to capture the actual community if we prohibit imbalanced edges when computing.

Many questions arise here due to the many limitations:

- Can these two lines of work be unified?
- Can we develop algorithms finding communities that are non-strictly balanced?

In this paper, we answer these questions affirmatively. We design a tailored function to measure the tolerance of the imbalance. To properly describe such tolerance, we adapt the frustration index as part of the tolerance function. In this way, we can either allow no tolerance to extract the MBS in the network or a loose tolerance to extract communities that might contain voices of disagreement. We further raise new problems based on this tolerance model and present a new region-based heuristic algorithm that computes maximal balanced subgraphs under the tolerance setting. Our new algorithm is versatile, delivering high-quality solutions across a range of problems based on the tolerance model, and it's also efficient, with an expected output time proportional to the size of the result. By setting different tolerances, we utilize our algorithm to handle different tasks and compare it with the performance of state-of-the-art algorithms[2] on 8 real-world datasets.

Our contributions are summarized as follows:

- We introduce a novel, generalized, and practical model for identifying balanced subgraphs with tolerance in signed graphs.
- We have developed an efficient region-based heuristic randomized algorithm, characterized by an expected time complexity proportional to the size of the output, and coupled with a guarantee of result quality.
- Extensive experiments show that our algorithm consistently outperforms the baselines in terms of the quality of the returned subgraphs and achieves up to 100× speedup in terms of running time.
- The effectiveness of our algorithm is also evident in its application to polarized community detection, as detailed in Section 5.5.

Outline. The rest of the paper is organized as follows. We review the related work in Section 2, and introduce our generalized maximum balanced subgraph model with tolerance in Section 3. Section 4 presents our region-based heuristic algorithm. Experimental results are given in Section 5, and we conclude in Section 6.

2 RELATED WORK

Signed Graphs. Signed graphs were first introduced by Harary in 1956 to study the notion of *balance* [22]. Cartwright and Harary generalized Heider's theory of balance onto signed graphs [12]. Harary also developed an algorithm to detect balance in signed graphs [24]. There are other works on studying the minimum number of sign changes to make the graph balance [3]. Spectral properties have also been studied recently. Hou et al. have studied the smallest eigenvalue in signed graphs' Laplacian [26, 27].

Many works focus on community detection or partition in signed graphs. Anchuri et al. give a spectral method that partitions the signed graph into non-overlapping balanced communities [4]. Doreian and Mrvar propose an algorithm for partitioning a directed signed graph and minimizing a measure of *imbalance* [18]. Yang et al. give a random-walk-based method to partition into cohesively polarized communities [37]. The more recent work is by Niu et al [32]. They leverage the balanced triangles, which model cohesion and polarization simultaneously, to design a good heuristic.

Maximum Balanced Subgraphs. The Maximum Balanced Subgraphs (MBS) problem has two variants: maximizing the number of vertices (MBS-V) and edges (MBS-E). Poljak et al. give a tight lower bound in 1986 [34] on the number of edges and vertices of the balanced subgraph. The MBS-E problem is in fact **NP**-hard since it can be formulated as a generalization of the standard MaxCut problem. To find the exact solution, there are some algorithms in the context of *fixed-parameter tractability* (FPT) are developed [16, 28]. More studies are on extracting large balanced subgraphs. DasGupta et al. have developed an algorithm based on semidefinite programming relaxation (SDP) [17]. For the MBS-V problem, Figueiredo and Frota propose several heuristic methods in 2014 [19].

In 2020, Ordozgoiti et al. proposed a new algorithm named Timbal [33] that extracts large balanced subgraphs regarding both vertices and edges. Timbal relies on signed spectral theory and the bound for perturbations of the graph Laplacian. However, Timbal is not stable and the balanced subgraph it found is sometimes small and unsatisfying as shown in our experiments.

Frustration Index. The frustration index was first introduced in 1950s [1, 23]. Computing the frustration index is related to the EdgeBipartization problem, which requires minimization of the number of edges whose deletion makes the graph bipartite. Since EdgeBipartization is **NP**-hard, computing the frustration index is

[1] konect.cc/networks/moreno_sampson
[2] Here, the SOTA algorithms are adapted to fit our new tasks under various tolerances.

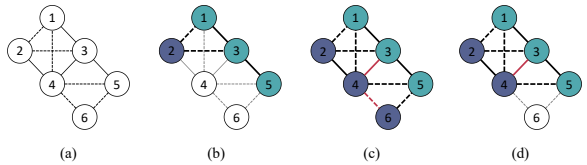

Figure 2: A signed graph (a), its MBS-V and MBS-E (b), its TMBS-V and TMBS-E (c), and its β-TMBS (d) with $\beta = \frac{1}{3}$.

also **NP**-hard. The MaxCut is also a special case of the frustration index problem. Assuming Khot's unique games conjecture [30], it is still **NP**-hard to approximate within any constant factor. For the non-constant factor case, there are works that produce a solution approximated to a factor of $O(\sqrt{\log n})$ [2] or $O(\log k)$ [7] where n is the number of vertices and k is the frustration index. Coleman et al. have given a review on different approximation algorithms [15]. Hüffner et al. show that the frustration index is *fixed parameter tractable* and can be computed in $O(2^k m^2)$, where m is the number of edges and k is the fixed parameter (the frustration index) [29]. There are also algorithms using binary programming models to compute the exact frustration index [5, 6].

3 PROBLEM SPECIFICATION

A signed graph is an undirected simple graph $G = (V, E^+, E^-)$ where V is the vertex set and E^+, E^- represent the positive and negative signed edge sets. We first give the formal definition of balanced graphs as follows, which is the same as the previous works [32, 33]. Note that we and these works require the graph to be connected for the community detection proposal.

Definition 3.1 (Balanced Graph). *Given a signed graph $G = (V, E^+, E^-)$, G is balanced if G is connected and there exists a partition $V = V_1 \cup V_2, V_1 \cap V_2 = \emptyset$ such that for each edge $(i, j) \in E^+$, vertices i and j belong to the same set within V_1 and V_2, while for each edge $(i, j) \in E^-$, they belong to the different sets.*

Graphs are usually not balanced, especially when they are from practical scenarios. Therefore, people turn to study to find the maximum balanced subgraph (MBS) from the given graph. There are usually two variants of problems: maximizing the number of vertices [33] and edges [17].

Problem 1 (MBS-V). *Given a signed graph $G = (V, E^+, E^-)$, find the graph $G' = (V', E'^+, E'^-)$ induced by $V' \subseteq V$ such that G' is balanced and $|V'|$ is maximized.*

Problem 2 (MBS-E). *Given a signed graph $G = (V, E^+, E^-)$, find the graph $G' = (V', E'^+, E'^-)$ induced by $V' \subseteq V$ such that G' is balanced and $|E'^+ \cup E'^-|$ is maximized.*

We illustrate the concepts of MBS-V and MBS-E with an example in Figure 2, along with other problems to be discussed below. Here, we give a graph with 6 vertices numbered from 1 to 6 where solid edges are positive and dashed edges are negative, presented in (a). When solving Problem 1 or 2 on this graph, the optimal subgraph is constructed by vertex 1, 2, 3, 5, where $V_1 = \{1, 3, 5\}$ and $V_2 = \{2\}$. They are painted in two different colors in (b).

Another well-known problem in this topic is computing the minimum number of edges whose deletion makes all connected components balanced. This minimum amount of the edge removal is called the *Frustration Index* of the given signed graph G, denoted as $L(G)$.

Now, we are ready to introduce the concept of *Balance with β-tolerance* in signed networks, which is central to our paper.

Definition 3.2 (Balanced Graph under β-Tolerance). *Given a signed graph $G = (V, E^+, E^-)$ and a tolerance parameter $\beta \in (0, 1]$, G is balanced under β-Tolerance if G is connected and $\frac{L(G)}{|E^+ \cup E^-|} \leq \beta$.*

In other words, when a graph is balanced under β-tolerance, it implies that by removing at most $\beta|E^+ \cup E^-|$ edges from it, we can ensure that all connected components of the remaining graph become strictly balanced. Expanding upon the original definition of balance, this concept allows for the tolerance of a maximum of $\beta|E^+ \cup E^-|$ edges that disobey the partitioning of polarized communities.

Under such β-tolerance restriction, finding the maximum balanced subgraph is generalized into the following two problems.

Problem 3 (TMBS-V). *Given a signed graph $G = (V, E^+, E^-)$ and a tolerance parameter $\beta \in (0, 1]$, find the graph $G' = (V', E'^+, E'^-)$ induced by $V' \subseteq V$ such that G' is balanced under β-Tolerance and $|V'|$ is maximized.*

Problem 4 (TMBS-E). *Given a signed graph $G = (V, E^+, E^-)$ and a tolerance parameter $\beta \in (0, 1]$, find the graph $G' = (V', E'^+, E'^-)$ induced by $V' \subseteq V$ such that G' is balanced under β-Tolerance and $|E'^+ \cup E'^-|$ is maximized.*

When β becomes any value between $(0, \frac{1}{|E^+ \cup E^-|})$, Problem 3 and Problem 4 are equivalent to the Problem 1 and Problem 2 respectively. However, unlike the strictly balanced graph, even deciding whether a signed graph satisfies β-tolerance is hard. Formally, we have the following lemma. The proof is deferred to Appendix A.1.

Lemma 3.1. *Given a signed graph $G = (V, E^+, E^-)$, deciding whether G is balanced under β-tolerance cannot be done within polynomial time for tolerance parameter $\beta > \frac{1}{|E^+ \cup E^-|}$, unless **P = NP**.*

Therefore, it is difficult to find large balanced graphs with tolerance and we cannot adapt any previous algorithms on MBS or other related problems. Meanwhile, there is another limitation that is worth mentioning. If $\beta \geq \frac{L(G)}{|E^+ \cup E^-|}$, Problem 3 and Problem 4 has a trivial optimal solution: the largest connected component. That is to say, in such loose tolerance restriction, the optimal solution to these two problems might not reflect the polarized community. The previous showcase given in Figure 2 (c) is a typical example: When we solve Problem 3 and 4 with $\beta = \frac{1}{3}$ in such graph, we get the whole graph. However, vertex 6 is in fact a noise that only has negative edges connected to both V_1 and V_2. Therefore, it should not be included in any meaningful subgraph solution.

To encounter the challenges and limitations, we first propose Tolerant Balance Index, a novel metric to evaluate signed graphs under the given tolerance for the balance.

Definition 3.3 (Tolerant Balance Index (TBI)). *Given a signed graph $G = (V, E^+, E^-)$ and a tolerance parameter $\beta \in (0, 1]$, define its Tolerant Balance Index $\Phi(G, \beta)$ as the value of $|E^+ \cup E^-| - \frac{L(G)}{\beta}$.*

Correspondingly, we propose the following problem as the main task to solve throughout the paper.

Problem 5 (β-TMBS). *Given a signed graph G and a tolerance parameter $\beta \in (0, 1]$, find the graph G' induced by $V' \subseteq V$ such that G' is connected and $\Phi(G', \beta)$ is maximized.*

It is easy to show, for any balanced graph G under β-tolerance, $\Phi(G, \beta) \geq 0$. To discuss the usage of solving Problem 5, we also want to address that maximizing the number of edges (Problem 2 and 4) approximates the solution for maximizing the number of vertices (Problem 1 and 3). That is to say, these two variants of MBS problems are related.

Lemma 3.2. *Given a signed graph G, the solutions to Problem 2 and Problem 4 are $\frac{1}{\Delta}$-approximations for the Problem 1 and Problem 3 respectively, where Δ is the maximum degree of vertices in G.*

We defer the proof of Lemma 3.2 to Appendix A.1. Considering the aforementioned interrelationship between two variants, we propose that finding the large connected subgraph with maximal $\Phi(G, \beta)$ (Problem 5) can be considered as a good approximation of the solutions to Problem 3 and Problem 4 simultaneously. In addition, since Problem 5 takes both the size and the polarity into account, it mitigates the limitation of large β. When we solve Problem 5 with $\beta = \frac{1}{3}$ on Figure 2 (a), the optimal subgraph will be vertices excluded vertex 6, where $V_1 = \{1, 3, 5\}$ and $V_2 = \{2, 4\}$, presented in (d). As it is shown, the noise vertex 6 is not mistakenly selected into the optimal subgraph we try to find. Therefore, the polarity is being preserved.

In Section 4, we will present an efficient and effective algorithm for Problem 5. Then in Section 5, we will demonstrate the experimental results to support our proposal.

4 ALGORITHM

We propose *Region-based Heuristic (RH)*, a new algorithm that searches for large balanced subgraphs with the β-tolerance restriction (Problem 5). Our algorithm runs in a given signed graph $G = (V, E^+, E^-)$, where E^+, E^- represent the positive and negative signed edge sets. The tolerance parameter β is also given as input.

4.1 Relaxation

It is not easy to compute $\Phi(G, \beta)$ directly. Instead, we propose an alternative method to approximate it from the lower end while guaranteeing the tolerance condition ($\Phi(G, \beta) \geq 0$) is not violated.

For each vertex in G, we assign a color in $\{0, 1\}$. Only vertices of the same color can be grouped in the same community. We denote any color assignment of G as $\mathcal{X}$. Such definition is aligned with the previous work [6]. Now, we are ready to give the formal definition of our newly proposed *Tolerant Balanced Count*.

Definition 4.1 (Tolerant Balance Count (TBC)). *Given a signed graph $G = (V, E^+, E^-)$ under coloring $\mathcal{X}$ and a tolerance parameter $\beta \in (0, 1]$, define its Tolerant Balance Count as*

$$\hat{\Phi}(G, \beta, \mathcal{X}) = |E^+ \cup E^-| - \frac{1}{\beta} \sum_{(i,j) \in E^+} \mathbb{1}\{x_i \neq x_j\} - \frac{1}{\beta} \sum_{(i,j) \in E^-} \mathbb{1}\{x_i = x_j\}$$

The following lemma states that $\Phi(G, \beta)$ is, in fact, the upper bound of $\hat{\Phi}(G, \beta, \mathcal{X})$ with respect to different coloring $\mathcal{X}$. The proof is deferred to Appendix A.2. We also have the following corollary that ensures the β-tolerance requirement is not violated.

Lemma 4.1. *Given a signed graph $G = (V, E^+, E^-)$ and a tolerance parameter $\beta \in (0, 1]$, we have $\Phi(G, \beta) = \max_{\mathcal{X}} \hat{\Phi}(G, \beta, \mathcal{X})$.*

Corollary 4.2. *Given a signed graph G and a tolerance parameter β, G is balanced under β-tolerance if there exists a coloring $\mathcal{X}$ such that $\hat{\Phi}(G, \beta, \mathcal{X}) \geq 0$.*

In this way, although we cannot compute $\Phi(G, \beta)$ directly, we can approximate the actual value by accumulating the non-negative $\hat{\Phi}(G, \beta, \mathcal{X})$ values of various coloring. Additionally, by Corollary 4.2, we can guarantee the β-tolerance during the whole computation. Such relaxation plays an important role in our algorithm, and the experimental results in Section 5 also validate its effectiveness.

4.2 Local Search

Our search process in Algorithm 1 starts from an initial vertex s instead of the whole vertices set. We will discuss how to choose s wisely in the later Section 4.3.

Search Operations. As discussed in Section 4.1, we will search for a connected subgraph G' and a coloring $\mathcal{X}$ such that $\hat{\Phi}(G', \beta, \mathcal{X})$ is as large as possible. In the following, if not specified, we use $\hat{\Phi}$ to denote the tolerant balanced count of the subgraph. Throughout the search process, we use a set S to store the selected vertices and the coloring simultaneously. Specifically, S stores selected tuples (x, c), where (x, c) represents a vertex x colored as c. There are three basic operations with S that will insert, delete, or change the color of a vertex x respectively. Note that these three operations well define the neighborhood of any solution we find.

- `VertexInsert`(x, c): Execute $S \leftarrow S \cup (x, c)$.
- `VertexDelete`(x): Let c be the color of x. Execute $S \leftarrow S \setminus (x, c)$.
- `ColorFlip`(x): Let c be the color of x and $\overline{c}$ be c's opposite color. Execute `VertexDelete`(x) and `VertexInsert`$(x, \overline{c})$ in order.

Since our search begins from s, we initialize S to be $\{(s, 0)\}$ (line 1). We use $\hat{\Phi}_{opt}$ and $\hat{\Phi}_{cur}$ to store the maximum $\hat{\Phi}$ that we have found and the current $\hat{\Phi}$ respectively. Initially, they are initialized to be 0 (line 1). During our process, if we choose to execute `VertexInsert`(x, c) or `ColorFlip`(x) for some vertex x, we will greedily execute one with the maximal increment of $\hat{\Phi}$ that it can contribute. Therefore, we use two max-heaps MH_I, MH_F to assist. After being initialized (line 2), since s is selected, for each neighbor v of s, we calculate the corresponding contribution to $\hat{\Phi}$ when `VertexInsert`$(v, 0)$ or `VertexInsert`$(v, 1)$ are executed and insert them to MH_I (line 4 to 5). For MH_F, since only s is selected, we calculate the contribution for `ColorFlip`(x) and insert it into MH_F (line 3).

Here, we do not use any structure to store the contribution when deleting any vertex from S. Instead, whenever we want to find the optimal deletion, we can compute for every vertex in S altogether with a total cost of $O(n)$, for which we may need to use the well-known Tarjan algorithm [25] to guarantee the subgraph stays connected after the deletion. We denote such process as `DelEval`(S).

Our search method is to execute one of the three operations repeatedly. If we only consider inserting vertices from S's neighbors, the total number of `VertexInsert` is bounded by the size of the graph. However, since we also have the other two operations, which are *non-incremental*, if we do not design a proper termination strategy, the time complexity might become exponential. Specifically, we have two parameters that help to define the termination strategy of the algorithm: A float number $p \in (0, 1)$ denotes the *non-incremental probability* and an integer T to limit the number of potentially wasted operations.

Operation Selection. We first discuss how to select an operation each time. Here, we will use the non-incremental probability p to help determine. We use op to denote the operation we select and $\Delta_{\hat{\Phi}}$ to denote the corresponding contribution value to $\hat{\Phi}$. These two variables are initialized by acquiring the best `VertexInsert` operation from MH_I (line 7), indicating choosing an insertion. Then, we generate a random float number z from uniform distribution $U(0, 1)$. If $z < p$, we also consider using a `ColorFlip` operation: We acquire the best `ColorFlip` operation from MH_F and update the two variables if the corresponding $\Delta_{\hat{\Phi}}$ is larger than the current one (line 8 to 9). Similarly, we regenerate z and if $z < \frac{p \log |S|}{|S|}$, we try choosing a `VertexDelete` operation: After calling `DelEval`(S) to recalculate for every possible vertex deletion, we choose the best among them and try updating (line 10 to 12). In this way, we choose an `VertexInsert` operation by default, and with some probability, we check if the current optimal `ColorFlip` and `VertexDelete` can contribute more. The two probabilities are by design and help to balance between accuracy and efficiency.

We execute the current op operation after the selection phase and also update the current $\hat{\Phi}_{cur}$. Since S is updated after the execution, we must also update MH_I and MH_F correspondingly. This is done in a similar way as the initialization (line 13 to 17).

Search Termination. When the current selected operation enlarges the current $\hat{\Phi}_{cur}$, we denote such operation as a *progressive* one. Otherwise, it is *non-progressive*. We use a parameter T to prevent too many *non-progressive* operations. Specifically, our search will terminate when the number of *non-progressive* operations exceeds T times the number of *progressive* operations. We implement such a strategy with a counter t. After we select and execute an operation (line 7 to 17), we compare the current $\hat{\Phi}_{cur}$ with $\hat{\Phi}_{opt}$. If $\hat{\Phi}_{cur}$ is smaller, we decrease t by 1. Otherwise, we update $\hat{\Phi}_{opt}$ and increase t by T (line 18 to 19). In this way, if $t < 0$, the search should terminate (line 6). In addition, we also terminate the search when S contains all vertices in G since no further insertion can be executed (line 6). After the search, we undo the last few non-progressive operations to retract the optimal subgraph we have found (line 20). We return with this subgraph as $\overline{H}$ and its corresponding $\hat{\Phi}$ (line 21).

Algorithm 1: RH: Search

Input: Signed graph G; Tolerance parameter β; Non-incremental probability p; Initial vertex s; Early stop turn limit T.
Output: $\overline{H} \subseteq G$: the found balanced subgraph with β-tolerance; $\hat{\Phi}$: the lower bound of $\overline{H}$'s balance value we achieved.

1 $S \leftarrow \{(s, 0)\}, \hat{\Phi}_{opt} \leftarrow 0, \hat{\Phi}_{cur} \leftarrow 0, t \leftarrow T$;
2 Initialize max-heaps MH_I, MH_F;
3 Insert s into MH_F;
4 **for** $v \in N(s)$ **do**
5 Insert $(v, 0)$ and $(v, 1)$ into MH_I;
6 **while** $t \geq 0$ *and* $|S| < |V(G)|$ **do**
7 $\{op, \Delta_{\hat{\Phi}}\} =$ `GetTopNode`(MH_I); `// ordered by` $\Delta_{\hat{\Phi}}$
8 **if** $z \sim U(0, 1) < p$ **then**
9 $\{op, \Delta_{\hat{\Phi}}\} \leftarrow \max(\{op, \Delta_{\hat{\Phi}}\},$ `GetTopNode`$(MH_F))$;
10 **if** $z \sim U(0, 1) < \frac{p \log |S|}{|S|}$ **then**
11 $\{\hat{op}, \overline{\Delta_{\hat{\Phi}}}\} \leftarrow$ `DelEval`(S);
12 $\{op, \Delta_{\hat{\Phi}}\} \leftarrow \max(\{op, \Delta_{\hat{\Phi}}\}, \{\hat{op}, \overline{\Delta_{\hat{\Phi}}}\})$;
13 $S \leftarrow$ Execute operation op on S;
14 $x \leftarrow$ the vertex of the op;
15 **for** $v \in N(x) \cup x$ **do**
16 Insert, update or delete v's operations in MH_I and MH_F;
17 $\hat{\Phi}_{cur} \leftarrow \hat{\Phi}_{cur} + \Delta_{\hat{\Phi}}$;
18 **if** $\hat{\Phi}_{cur} \leq \hat{\Phi}_{opt}$ **then** $t \leftarrow t - 1$;
19 **else** $\hat{\Phi}_{opt} \leftarrow \hat{\Phi}_{cur}, t \leftarrow t + T$;
20 Undo the last non-progressive operations on S;
21 **return** $\overline{H} = G[S], \hat{\Phi} = \hat{\Phi}_{opt}$;

4.3 Region-based Sampling

In the previous section, we propose a search process that starts from an arbitrary vertex s. It is reasonable to foresee that the choice of s might affect the result significantly. If we start only from too few vertices, our result in the end might be some local maximal solutions, which would be much worse than the global maximal one. One of the solutions is to enumerate all possible s, *i.e.*, all vertices in G. However, such pure enumeration may result in excessively high time complexity. To balance between performance and efficiency, we propose a *Region-based Sampling* strategy.

Our sampling method is mainly based on two hypotheses. The first hypothesis indicates that the probability of finding a nearly optimal subgraph is high if we are able to select some vertices in the optimal subgraph as the starting vertex. Here, the 'optimal' denotes the solution we found by enumerating all vertices as the starting vertex. We formally state such a hypothesis as follows.

Hypothesis 4.1. *Given a signed graph G and a tolerance parameter β, suppose the optimal subgraph found by Algorithm 1 starting with vertex x is G_x, and the optimal graph among all G_x is G_{opt}. For the given ϵ, there exists a subset $V' \subseteq V(G_{opt})$ with $\frac{|V'|}{|V(G_{opt})|} \geq \frac{1}{2}$ such that $\Phi(G_x, \beta) \geq (1 - \epsilon)\Phi(G_{opt}, \beta), \forall x \in V'$.*

Another hypothesis describes the relation between $\hat{\Phi}$ values returned by two different calls of Algorithm 1, if we select different starting vertex. We argue that if the return $\hat{\Phi}$ value is larger, the found subgraph will likely be bigger. We formally state such a hypothesis as follows.

Hypothesis 4.2. *Given a signed graph G and a tolerance parameter β, suppose the optimal subgraph and coloring found by Algorithm 1*

Algorithm 2: RH: Sampling

Input: Signed graph G; Tolerance parameter β; Iteration constant C; Non-incremental probability p; Early stop turn limit T.
Output: $H \subseteq G$: the found balanced subgraph with β-tolerance.

1 $H \leftarrow \emptyset$;
2 $\hat{\Phi}_{opt} \leftarrow 0$;
3 $TotalSize \leftarrow 0$;
4 **while** $TotalSize < C|V(G)|$ **do**
5 $s \leftarrow$ Sample a vertex from $V(G)$;
6 $\overline{H}, \hat{\Phi} \leftarrow \texttt{Search}(G, \beta, s, p, T)$;
7 **if** $\hat{\Phi} > \Phi_{opt}$ **then**
8 $\hat{\Phi} \leftarrow \hat{\Phi}_{opt}$;
9 $H \leftarrow \overline{H}$;
10 $TotalSize \leftarrow TotalSize + |V(\overline{H})|$;
11 **return** H;

starting with vertex a are G_a and $\mathcal{X}_a$, starting with b are G_b and $\mathcal{X}_b$. If $\hat{\Phi}(G_a, \beta, \mathcal{X}_a) \geq \hat{\Phi}(G_b, \beta, \mathcal{X}_b)$, we have $|V(G_b)| \leq 2|V(G_a)|$.

With these two hypotheses, we have the following lemma that describes a sampling strategy that is able to find a $(1-\epsilon)$-optimal subgraph within an acceptable number of calls of the search process. The proof is deferred to Appendix A.2.

Lemma 4.3. *Given a signed graph G, a tolerance parameter β, and a positive $\epsilon < 1$, suppose we run Algorithm 1 in k iterations, where the i-th iteration starts with a uniformly sampled vertex $x_i \in V(G)$, and the optimal subgraph found is G_i. If Hypothesis 4.1 and Hypothesis 4.2 hold, the expected number of iterations that find a $(1-\epsilon)$-optimal subgraph $\mathbb{E}[X]$ is $\Omega(\sum_{i=1}^{k} |V(G_i)|/|V(G)|)$.*

We provide an implementation of such sampling strategy in Algorithm 2, which is, in fact, an application of Lemma 4.3. We use $H, \hat{\Phi}_{opt}$ to keep track of the current optimal subgraph and its corresponding $\hat{\Phi}$ value. They are initialized to be $\emptyset$ and 0 in the beginning (line 1 to 2). We also use a variable $TotalSize$ to keep track of the total size of all return subgraphs from Algorithm 1, which is also initialized to be 0 (line 3).

For each iteration, we randomly select a vertex s (line 5) as the starting vertex and pass it into Algorithm 1 (line 6). After receiving the result from Algorithm 1, we update $H, \hat{\Phi}_{opt}$ (line 8 to 9) if the newly found subgraph is better (line 7). Before the new iteration, we accumulate the size of the newly found subgraph into $TotalSize$ (line 10. The whole process will stop when $TotalSize \geq C|V(G)|$ (line 4). By Lemma 4.3, we can set a proper termination condition by accumulating the subgraph size from each search process. More specifically, When $TotalSize$ reaches $C|V(G)|$, it is expected to find a nearly optimal solution C times. In the end, we return H as the main result (line 11).

Time Complexity Analysis. We show that the time complexity of Algorithm 2 is $O(\Delta(C+1)n\log^2 n)$, where n is the number of vertices in G. We start with the time complexity of Algorithm 1.

Lemma 4.4. *Suppose that the loop starting on Line 6 repeats for a total of t times. Let h be the size of S on Line 21. Then we have, with high probability, $h = \Omega(t/\log n)$ and that the running time of Algorithm 1 is $O(\Delta t \log n)$ in expectation.*

Proof. By Lemma A.1, with high probability, $h = \Omega(t/\log n)$.

Let Δ be the maximum degree of G. The running time of each iteration is dominated by the cost of updating the heaps and the time for `DelEval(S)`. Updating the heaps cost $O(\Delta \log n)$ time. The function `DelEval(S)` costs $O(\Delta|S|)$ time. It is called with probability $\frac{p\log|S|}{|S|}$ in each iteration. Thus, its expected runtime in each iteration is $O(\Delta \log n)$. We have the running time of Algorithm 1 is $t \cdot O(\Delta \log n) = O(\Delta t \log n)$. □

Lemma 4.5. *With high probability, the expected time complexity of Algorithm 2 is $O(\Delta(C+1)n\log^2 n)$.*

Proof. Suppose we call Line 6 for k times. For the i-th call, let t_i be the number of iterations of the loop on Line 6 and let h_i $(1 \leq i \leq k)$ be the size of S on Line 21. By Lemma 4.4, the total time complexity in expectation is $O(\sum_{i=1}^{k} \Delta t_i \log n)$, and the sum of h_i satisfies $\sum_{i=1}^{k} h_i = \Omega(\sum_{i=1}^{k} t_i/\log n)$. Since $\sum_{i=1}^{k} h_i \leq (C+1)n$, we have that the total runtime of Algorithm 2 in expectation is $O(\sum_{i=1}^{k} \Delta t_i \log n) = O(\Delta(C+1)n\log^2 n)$. □

5 EVALUATION

In this section, we address the following research questions to evaluate various important aspects of our algorithm:

- **RQ1 (Effectiveness - TMBS):** Given various β, what are the optimal subgraphs in terms of size, and Tolerant Index Count, found by our method and baselines?
- **RQ2 (Effectiveness - MBS):** What are our method's and baselines' performances in finding the maximum balanced subgraph?
- **RQ3 (Efficiency):** What are the runtimes for our method and the baselines, and how do they scale with very large networks?
- **RQ4 (Generalizability):** Can our model and method be adapted to other related tasks in the signed networks?

We also conduct three experiments shown in appendix B, which are designed to validate our hypotheses, determine optimal hyperparameters, and assess stability.

5.1 Experimental Setting

Baselines. We compare our method with the baselines from highly related works (*e.g.,* MBS and polarized community detection), including spectral and other heuristic methods.

Note that since finding tolerant balanced subgraphs is typically more challenging than previous community detection tasks in signed networks, we also adapt our TBC-relaxation in Section 4.1 for all baseline methods to ensure that they can return the subgraphs satisfying the tolerance constraint. We summarize the core ideas of each baseline method as follows.

- Eigen: Based on a spectral method from [10], we first compute $\mathbf{v}$, the eigenvector of the signed adjacency matrix A corresponding to the largest eigen value λ. For each vertex i, if a Bernoulli experiment with success probability $p = |\mathbf{v}_i|$ is successful, we assign vertex i a color determined by $\text{sgn}(\mathbf{v}_i)$. The maximum

Table 1: Signed network datasets used in experiments, including the number of vertices ($|V|$) and edges ($|E|$), the ratio of negative edges (ρ^-), and the ratio of non-zero elements (δ).

Dataset	$\|V\|$	$\|E^+ \cup E^-\|$	$\rho^- = \frac{\|E^-\|}{\|E^+ \cup E^-\|}$	$\delta = \frac{2\|E^+ \cup E^-\|}{\|V\|(\|V\|-1)}$
Bitcoin	5k	21k	0.15	1.2e-03
Epinions	131k	711k	0.17	8.2e-05
Slashdot	82k	500k	0.23	1.4e-04
Twitter	10k	251k	0.05	4.2e-03
Conflict	116k	2M	0.62	2.9e-04
Elections	7k	100k	0.22	3.9e-03
Politics	138k	715k	0.12	7.4e-05
Growth	1.87M	40M	0.50	2.3e-05

connected components with non-negative $\hat{\Phi}$ serve as solutions to Problem 3 and Problem 4, while the connected component with the maximum $\hat{\Phi}$ provides the solution to Problem 5.

- Timbal: Based on a state-of-the-art method for MBS from [33], we start from the entire graph and repeatedly remove vertices from it, guided by an eigenvalue approximation outlined in their paper. Specifically, we compute $\hat{\Phi}$ by the optimal coloring among two non-trivial methods [10, 21] for TMBS problems.
- GreSt: This method combines a classical and effective algorithm in dense subgraph [13] and a heuristic coloring method for signed networks by spanning tree [21]. Firstly, we determine the coloring of the entire graph by a random spanning tree. We then repeatedly remove vertices maximizing $\hat{\Phi}$ of the remaining graph. A min-heap is used to speed up the process of finding vertices. To acquire the solution to different tasks, we need to store the deletion operations. Then we can restore the graph by undoing deletions one by one. Such reversal allows us to keep track of the optimal connected component efficiently, instead of scanning the whole graph after each deletion.
- RH (LS): Instead of using region-based sampling, we execute our local search in Section 4.2 for all starting vertices and select the optimal solutions, to investigate the effectiveness of the regional-based sampling. This method may provide a better solution than RH but is not efficient.

Datasets. We select 7 publicly-available real-world signed networks[3], Bitcoin, Epinions, Slashdot, Twitter, Conflict, Elections, and Politics, which were widely used in previous related works [10, 32, 33]. In addition, to investigate our methods' scalability, we generate Growth, a very large signed network induced from the real temporal network Wikipedia-growth[4]. Specifically, we select a threshold τ, and give positive signs for the edges with time stamp $t(e) \geq \tau$ and negative signs for the edges with $t(e) < \tau$. By using a proper τ, the ratio of negative edges of the induced graph can be 0.5. In this network, a balanced graph contains two communities, where the edges within each community are recently formed, while the crossing edges are relatively old. Detailed information for each dataset can be found in Table 1.

All experiments are conducted on a Ubuntu 22.04 LTS workstation, equipped with a 12th Gen Intel(R) Core(TM) i9-12900HX. We set hyperparameters $T = 20, p = 0.8, C = 1.5$ for all experiments.

[3]From konect.cc and snap.stanford.edu
[4]http://konect.cc/networks/wikipedia-growth

5.2 Finding Balanced Subgraphs with Tolerance

We aim to identify sizable and polarized subgraphs while taking the tolerance into account (**RQ1**). Firstly, by running algorithms in Section 5.1, we keep track of the optimal subgraph that we have found to Problem 3, Problem 4, and Problem 5 respectively. Noticeably, we do not modify either RH or RH (LS) for Problem 3 and Problem 4. That is to say, we directly use our result on solving Problem 5 to compare with other algorithms which are tailored for either Problem 3 or Problem 4, which demonstrates our proposed Problem 5 is a good approximation for the other two problems.

The comparison results for these three problems are shown in Figure 3, Figure 4, and Figure 5, respectively. In our experiments, we consider 15 different tolerance parameters ($\beta_i = 2^{-i/2}$ for all $2 \leq i \leq 16$), where $i = 2$ implies allowance for all connected subgraphs, while $i = 16$ means that the found subgraphs are almost strictly balanced. Since Problem 3 and Problem 4 become trivial when $\beta \geq \frac{L(G)}{|E^+ \cup E^-|}$, we shade the corresponding ranges[5] in Figure 3 and Figure 4. We omit the results for methods that cannot finish in 100,000 seconds.

Observing the experiment results, our proposed method RH outperforms all other baselines significantly in all three problems. This also demonstrates that our proposed tolerance model is a more realistic model for the general case of balanced signed graph models. In addition, we can see that RH produces results that are close to RH (LS). Therefore, we manage to empirically support our hypothesis in Section 4.3 in real-world data.

5.3 Finding Strictly Balanced Subgraphs

Finding the Maximum Balanced Subgraph (Problem 1 and 2) is an important and well-studied problem in signed networks [33] (**RQ2**). As previously discussed, MBS problems are special forms of proposed TMBS problems (Problem 3 and 4) by setting $\beta < \frac{1}{|E^+ \cup E^-|}$.

Table 2 shows the results of our case study on the MBS problems, where we omit the results for Eigen since it cannot return subgraphs other than a single vertex. RH produces significantly better results than the previous state-of-the-art method Timbal, which is specifically designed for the MBS problems while ours is not.

5.4 Running Time Analysis

This experiment is designed to study the efficiency and scalability of our method (**RQ3**). As shown in Table 3, when the graph size is up to ($|V| = 1.87$M, $|E^+ \cup E^-| = 40$M), RH can still produce reasonable results within 1,000 seconds. On the other hand, the other four compared methods either produce much worse results while taking longer time or even fail to finish execution in 100,000 seconds. In all, the empirical results demonstrate the efficiency of our method in real-world data in addition to its asymptotic theoretical complexity.

5.5 Solving the 2PC Problem

This experiment aims to evaluate the generalizability when adapting RH and our tolerant model to different variants of polarity

[5]As computing the exact value of $L(G)$ is **NP**-hard, we cannot calculate the exact range of β corresponding to the trivial cases. Instead, we calculate an upper bound $\overline{L}$ of $L(G)$ by a promising valid solution and shade only the subrange $\beta \geq \frac{\overline{L}}{|E^+ \cup E^-|}$.

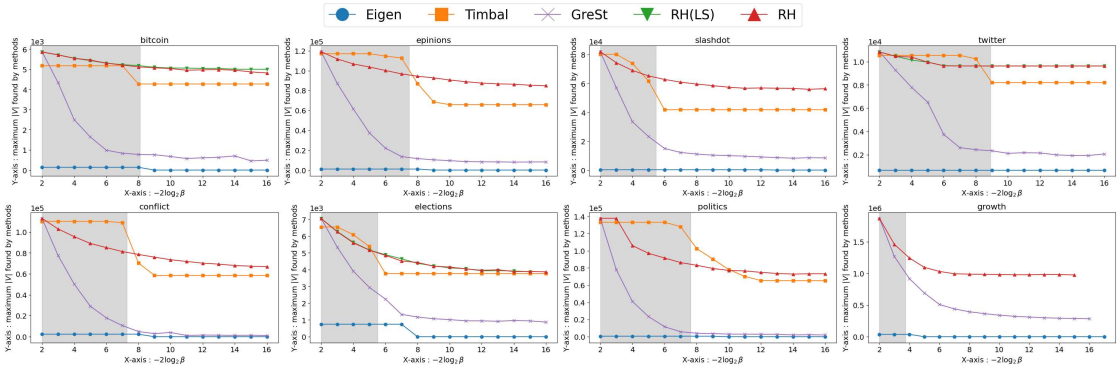

Figure 3: Problem 3: Comparing maximum tolerantly balanced subgraphs in vertex cardinality ($|V|$) across various tolerances ($\beta = 2^{-x/2}$), where the part corresponding to trivial β values for this problem has been shaded.

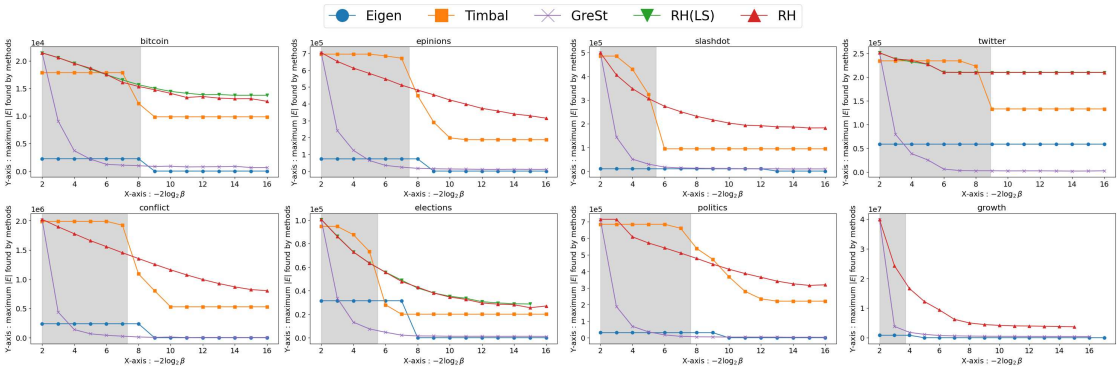

Figure 4: Problem 4: Comparing maximum tolerantly balanced subgraphs in edge cardinality ($|E^+ \cup E^-|$) across various tolerances ($\beta = 2^{-x/2}$), where the part corresponding to trivial β values for this problem has been shaded.

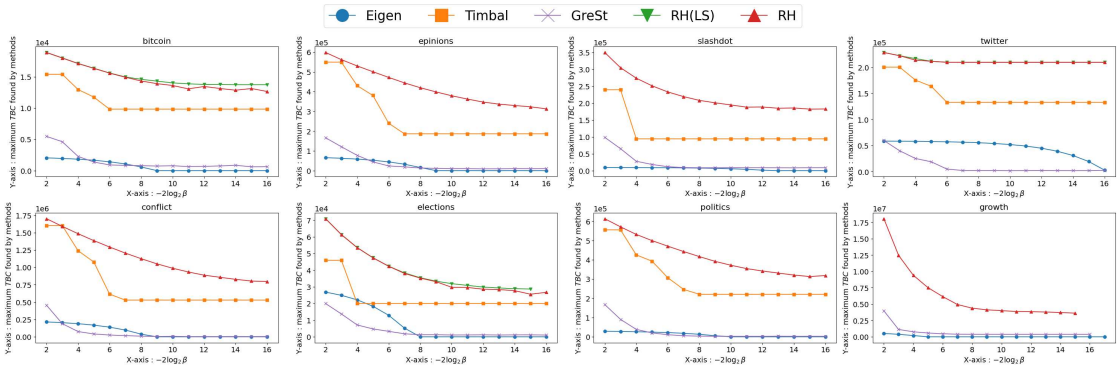

Figure 5: Problem 5: Comparing maximum tolerantly balanced subgraphs in TBC ($\hat{\Phi}$) across various tolerances ($\beta = 2^{-x/2}$).

Table 2: Case study: Largest strictly balanced subgraph (in terms of $|V|$ or $|E|$) found by each method for each dataset ($\beta < \frac{1}{|E^+ \cup E^-|}$), where *NA* denotes the corresponding method cannot finish in 100,000 seconds on the respective dataset.

Method	Bitcoin		Epinions		Slashdot		Twitter		Conflict		Elections		Politics		Growth	
	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$
Timbal	4,050	9,757	65,879	190,751	43,742	101,590	8,636	129,927	51,463	361,663	3,579	17,633	65,188	230,529	*NA*	*NA*
GreSt	571	713	7,078	9,229	7,923	8,465	1,895	2,277	759	4,885	935	1,232	2,371	3,155	272,621	375,603
RH (LS)	**5,002**	**13,746**	*NA*	*NA*	*NA*	*NA*	**9,628**	**209,633**	*NA*	*NA*	**3,970**	**28,453**	*NA*	*NA*	*NA*	*NA*
RH	4,935	13,050	**84,165**	**302,152**	**55,968**	**181,069**	**9,628**	**209,633**	**65,468**	**746,640**	3,926	26,478	**72,431**	**317,384**	**999,504**	**3,415,441**

community detection (**RQ4**). In addition to balance-related problems, 2-Polarized-Communities (2PC), proposed by Bonchi and Galimberti [10], serves as another model for polarized community detection. In their model, the measurement of polarity is penalized by the size of the solution:

Problem 6 (2PC). *Given a signed graph G with signed adjacency matrix A, find a vector $x \in \{-1, 0, 1\}$ that maximizes $\frac{x^T Ax}{x^T x}$.*

To solve the 2PC problem by our tolerance balance model, we add an additional penalty term to the solution size in the tolerant balance count: We define $\hat{\Phi}'(G, \beta, \mathcal{X}, \rho) = \hat{\Phi}(G, \beta, \mathcal{X}) - \rho|V|$ to serve as a new object function in our RH algorithm. Throughout the algorithm, we set $\beta = \frac{1}{2}$. This is because when $\beta = \frac{1}{2}$, if $\hat{\Phi}' \geq 0$, the polarity is no less than ρ. Therefore, Problem 6 can be solved by the new RH algorithm, where we apply an iterative mechanism on ρ. The algorithm details can be found in Appendix A.3.

We compare RH's results with the optimal result computed by the two methods in Bonchi and Galimberti's paper [10] (*i.e.,* Eigen and Greedy). The results are shown in Table 4. RH's slight modifications efficiently yield promising communities, demonstrating the adaptability and effectiveness of our tolerant balance model and algorithm in varied polarity community detection scenarios.

Table 3: Mean running time (s) for each method on the various β in Section 5.2.

Dataset	Eigen	Timbal	GreSt	RH (LS)	RH
Bitcoin	0.979	2.670	0.018	135.040	0.057
Epinions	9.213	220.919	0.319	$\geq 100,000$	4.162
Slashdot	7.287	226.805	0.218	$\geq 100,000$	3.175
Twitter	1.704	7.306	0.085	2,516.671	0.493
Conflict	12.899	310.736	0.781	$\geq 100,000$	9.031
Elections	1.022	4.542	0.039	675.484	0.288
Politics	10.003	143.605	0.304	$\geq 100,000$	4.782
Growth	297.299	$\geq 100,000$	48.915	$\geq 100,000$	813.099

Table 4: Optimal subgraphs in the term of 2PC-polarity ($\frac{x^T Ax}{x^T x}$) found by each method, associated with running time.

Dataset	RH		Eigen		Greedy	
	polarity	time (s)	polarity	time (s)	polarity	time (s)
Bitcoin	**14.82**	**0.035**	14.76	0.706	14.50	0.999
Epinions	**85.27**	**0.873**	64.36	10.271	85.15	428.625
Slashdot	41.21	**0.584**	39.85	7.325	**41.36**	154.383
Twitter	**87.21**	**0.818**	87.04	1.717	86.97	7.853
Conflict	**94.27**	15.134	87.83	**13.144**	63.99	552.629
Elections	36.27	**0.395**	35.87	0.950	**36.34**	2.223
Politics	44.95	**0.862**	44.22	9.931	**45.01**	475.353

6 CONCLUSIONS

This paper presents a new and versatile model to identify polarized communities in signed graphs. The model accommodates inherent imbalances in polarized communities through a tolerance feature. Additionally, we propose a region-based heuristic algorithm. Through a wide variety of experiments on graphs of up to 40 M edges, it demonstrates effectiveness and efficiency beyond the state-of-the-art methods in addressing both traditional and generalized MBS problems. We also adapt our model and algorithm to 2PC, a related but essentially different task, further verifying their generalizability.

7 REPRODUCIBILITY

Codes for our methods and for reproducing all the experimental results are available at GitHub[6].

ACKNOWLEDGMENTS

Chenhao Ma was partially supported by NSFC under Grant 62302421, Basic and Applied Basic Research Fund in Guangdong Province under Grant 2023A1515011280, and the Guangdong Provincial Key Laboratory of Big Data Computing, the Chinese University of Hong Kong, Shenzhen. We thank Yixiang Fang (The Chinese University of Hong Kong, Shenzhen), Qingyu Shi (Hailiang Foreign Language School), and Xinwen Zhang (The Chinese University of Hong Kong, Shenzhen) for valuable advice on this project.

[6] https://github.com/joyemang33/RH-TMBS

REFERENCES

[1] Robert P Abelson and Milton J Rosenberg. 1958. Symbolic psycho-logic: A model of attitudinal cognition. *Behavioral Science* (1958).

[2] Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. 2005. O($\sqrt{\log n}$) Approximation Algorithms for Min UnCut, Min 2CNF Deletion, and Directed Cut Problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. 573–581.

[3] Jin Akiyama, David Avis, Vasek Chvátal, and Hiroshi Era. 1981. Balancing signed graphs. *Discrete Applied Mathematics* 3, 4 (1981), 227–233.

[4] Pranay Anchuri and Malik Magdon-Ismail. 2012. Communities and balance in signed networks: A spectral approach. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 235–242.

[5] Samin Aref, Andrew J Mason, and Mark C Wilson. 2018. Computing the line index of balance using integer programming optimisation. *Optimization Problems in Graph Theory: In Honor of Gregory Z. Gutin's 60th Birthday* (2018), 65–84.

[6] Samin Aref, Andrew J Mason, and Mark C Wilson. 2020. A modeling and computational study of the frustration index in signed networks. *Networks* 75, 1 (2020), 95–110.

[7] Adi Avidor and Michael Langberg. 2007. The multi-multiway cut problem. *Theoretical Computer Science* 377, 1-3 (2007), 35–42.

[8] Prithu Banerjee, Wei Chen, and Laks VS Lakshmanan. 2023. Mitigating Filter Bubbles Under a Competitive Diffusion Model. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–26.

[9] Fabian Baumann, Philipp Lorenz-Spreen, Igor M Sokolov, and Michele Starnini. 2020. Modeling echo chambers and polarization dynamics in social networks. *Physical Review Letters* 124, 4 (2020), 048301.

[10] Francesco Bonchi, Edoardo Galimberti, Aristides Gionis, Bruno Ordozgoiti, and Giancarlo Ruffo. 2019. Discovering Polarized Communities in Signed Networks. *CoRR* abs/1910.02438 (2019). arXiv:1910.02438 http://arxiv.org/abs/1910.02438

[11] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2016. *Recent advances in graph partitioning*. Springer.

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

[13] Moses Charikar. 2000. Greedy Approximation Algorithms for Finding Dense Components in a Graph. In *Approximation Algorithms for Combinatorial Optimization*, Klaus Jansen and Samir Khuller (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 84–95.

[14] Uthsav Chitra and Christopher Musco. 2020. Analyzing the impact of filter bubbles on social network polarization. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 115–123.

[15] Tom Coleman, James Saunderson, and Anthony Wirth. 2008. A local-search 2-approximation for 2-correlation-clustering. In *European Symposium on Algorithms*. Springer, 308–319.

[16] Robert Crowston, Gregory Gutin, Mark Jones, and Gabriele Muciaccia. 2013. Maximum balanced subgraph problem parameterized above lower bound. *Theoretical Computer Science* 513 (2013), 53–64.

[17] Bhaskar DasGupta, German Andres Enciso, Eduardo Sontag, and Yi Zhang. 2007. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. *Biosystems* 90, 1 (2007), 161–178.

[18] Patrick Doreian and Andrej Mrvar. 1996. A partitioning approach to structural balance. *Social networks* 18, 2 (1996), 149–168.

[19] 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.

[20] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[21] 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

[22] Frank Harary. 1953. On the notion of balance of a signed graph. *Michigan Mathematical Journal* 2, 2 (1953), 143–146.

[23] Frank Harary. 1959. On the measurement of structural balance. *Behavioral Science* 4, 4 (1959), 316–323.

[24] Frank Harary and Jerald A Kabell. 1980. A simple algorithm to detect balance in signed graphs. *Mathematical Social Sciences* 1, 1 (1980), 131–136.

[25] John Hopcroft and Robert Tarjan. 1973. Algorithm 447: Efficient Algorithms for Graph Manipulation. *Commun. ACM* 16, 6 (jun 1973), 372–378. https://doi.org/10.1145/362248.362272

[26] Yaoping Hou, Jiongsheng Li, and Yongliang Pan. 2003. On the Laplacian eigenvalues of signed graphs. *Linear and Multilinear Algebra* 51, 1 (2003), 21–30.

[27] Yao Ping Hou. 2005. Bounds for the least Laplacian eigenvalue of a signed graph. *Acta Mathematica Sinica* 21, 4 (2005), 955–960.

[28] Falk Hüffner, Nadja Betzler, and Rolf Niedermeier. 2007. Optimal edge deletions for signed graph balancing. In *International Workshop on Experimental and Efficient Algorithms*. Springer, 297–310.

[29] Falk Hüffner, Nadja Betzler, and Rolf Niedermeier. 2010. Separator-based data reduction for signed graph balancing. *Journal of combinatorial optimization* 20 (2010), 335–360.

[30] Subhash Khot. 2002. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 767–775.

[31] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*. 677–686.

[32] Jason Niu and A. Erdem Sarıyüce. 2023. On Cohesively Polarized Communities in Signed Networks. In *Companion Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 1339–1347. https://doi.org/10.1145/3543873.3587698

[33] Bruno Ordozgoiti, Antonis Matakos, and Aristides Gionis. 2020. Finding large balanced subgraphs in signed networks. In *Proceedings of The Web Conference 2020*. 1378–1388.

[34] 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.

[35] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[36] Yichen Xu, Chenhao Ma, Yixiang Fang, and Zhifeng Bao. 2023. Efficient and Effective Algorithms for Generalized Densest Subgraph Discovery. *Proc. ACM Manag. Data* 1, 2, Article 169 (jun 2023), 27 pages. https://doi.org/10.1145/3589314

[37] Bo Yang, William Cheung, and Jiming Liu. 2007. Community mining from signed social networks. *IEEE transactions on knowledge and data engineering* 19, 10 (2007), 1333–1348.

A OMITTED PROOF AND ALGORITHM

A.1 Omitted Proof in Section 3

Proof of lemma 3.1. We prove the lemma by a reduction from the calculation of the Frustration Index. If we can decide whether G is balanced under β-tolerance within the polynomial runtime, we will be able to decide whether $L(G)$ is greater than any guessing value k by setting $\beta = \frac{k}{|E^+ \cup E^-|}$. Combined with a binary search, we can compute the Frustration Index of G within the polynomial runtime, which has been proven as a **NP**-hard problem [2]. □

Proof of lemma 3.2. We first show that the solutions to Problem 2 is a $\frac{1}{\Delta}$-approximations for the Problem 1, and the relations between Problem 3 and Problem 4 can be proven similarly.

Given a signed graph G, let the optimal subgraph to Problem 1 be G_1 with n_1 vertices and m_1 edges, and the optimal subgraph to Problem 2 be G_2 with n_2 vertices and m_2 edges. Since G_1 is connected, we have

$$m_1 \geq n_1 \tag{1}$$

Since Δ is the maximum degree of vertices in G, we have

$$n_2 \geq \frac{m_2}{\Delta} \tag{2}$$

Combined with (1) and (2), we have $n_2 \geq \frac{m_2}{\Delta} \geq \frac{m_1}{\Delta} \geq \frac{n_1}{\Delta}$, which implies G_2 is a $\frac{1}{\Delta}$-approximations for the Problem 1. □

A.2 Omitted Lemma and Proof in Section 4

Proof of lemma 4.1. The lemma is equivalent to that:

$\sum_{(i,j)\in E^+} \mathbb{1}\{x_i \neq x_j\} + \sum_{(i,j)\in E^-} \mathbb{1}\{x_i = x_j\}$ is a tight upper bound on $L(G)$.

First, given a coloring $\mathcal{X}$, if we remove all edges that let the indicator functions be true from G, the remaining graph will be balanced for each connected component due to Definition 3.1. This implies that $\sum_{(i,j)\in E^+} \mathbb{1}\{x_i \neq x_j\} + \sum_{(i,j)\in E^-} \mathbb{1}\{x_i = x_j\}$ is no less than the $L(G)$, which is the minimum number of edges to be removed.

We then show the bound is tight. Consider the remaining graph G' from removing all edges contributing to $L(G)$. Since all connected components of G' are balanced, we can find a partition $V = V_1 \cup V_2$ and $V_1 \neq V_2$ such that if we assign $x_i = 0$ for all $i \in V_1$ and $x_j = 1$ for all $j \in V_2$, the value of $\sum_{(i,j)\in E^+} \mathbb{1}\{x_i \neq x_j\} + \sum_{(i,j)\in E^-} \mathbb{1}\{x_i = x_j\}$ will become to zero for G'. This implies the same value for G under the coloring is equal to $L(G)$ and thus the bound is tight. □

Proof of lemma 4.3. Suppose the optimal subgraph found by Algorithm 1 among all starting vertex is G_{opt} and the probability of Algorithm 2 finding a $(1-\epsilon)$-optimal subgraph in one iteration is Pr[Sucess]. By Hypothesis 4.1, we have *(i)* $\Pr[\text{Sucess}] \geq \frac{|V(G_{opt})|}{2|V(G)|}$. Let $C = \frac{\sum_{i=1}^{k} |V(G_i)|}{|V(G)|}$, and then we have $\sum_{i=1}^{k} |V(G_i)| = C|V(G)|$. By Hypothesis 4.2, $\sum_{i=1}^{k} 2|V(G_{opt})| \geq C|V(G)|$, and thus *(ii)* $k \geq \frac{C}{2} \frac{|V(G)|}{|V(G_{opt})|}$. Combining *(i)* and *(ii)*, we have $\mathbb{E}[X] = k \Pr[\text{Sucess}] \geq \frac{C}{2} \frac{|V(G)|}{|V(G_{opt})|} \cdot \frac{|V(G_{opt})|}{2|V(G)|} = \frac{C}{4} = \Omega(C)$. □

Lemma A.1. *With high probability, for all valid i, after the i-th iteration of the on Line 6 of Algorithm 1,* $|S| = \Omega(\frac{i}{\log n})$.

Proof. Let x denote the value of $|S|$. The initial value of x is 1. x changes according to the following rules in each iteration:

- With probability $(1-p)(1-p\frac{\log x}{x})$, x increases by 1;
- With probability $p\frac{\log x}{x}$, x increases by 1, 0, or -1;
- With probability $p(1-p\frac{\log x}{x})$, x increases by 1, or 0;
- The process ends when x reaches n.

Suppose the process lasts for t steps. We shall prove that with high probability, it holds simultaneously for all i from 1 to t that the value of x after the first i steps is $\Omega(\frac{i}{\log n})$.

We denote $(1-p)(1-p\frac{\log x}{x})$ by $p^+(x)$, $p(1-p\frac{\log x}{x})$ by $p^0(x)$, and $p\frac{\log x}{x}$ by $p^-(x)$.

Let $next(x)$ be the value of x in the next step.

Intuitively, x increases over steps because $p^+(x)$ is much larger than $p^-(x)$ for large enough x. Thus, we will partition steps into two parts by the value of x. Let x_0 be the smallest x such that $p^-(x) < p^+(x)/2$. When x is at least x_0, x increases in expectation. When x is smaller than x_0, x will reach x_0 soon.

We can solve for such x_0 by the definition of $p^+(x)$ and $p^-(x)$. We have that x_0 is a constant depending on p satisfying $\frac{\log x_0}{x_0} = \frac{1-p}{3p-p^2}$. Thus, for any $x \geq x_0$, $\mathbb{E}[next(x)] \geq x + \frac{1-p}{3-p}$.

We first sketch the proof. Consider the first i iterations. We partition the i iterations into $\Theta(\log(n))$ intervals with $i/\log n$ length each. At the beginning of each interval, with high probability, within $O(\log n)$ iterations, x reaches a value that is at least x_0. After reaching x_0, x will never go below x_0 during the current interval with a constant probability. Conditioning on x never going below x_0, the expected increment of x in the current interval is at least cl for some constant c where l is the number of remaining iterations in the interval. l is at least $k/\log n - O(\log n)$. Because x may increase by at most l, we have, by Markov bound, $\Pr[x \text{ increases by no more than } cl/10] \leq \frac{10-10c}{10-c}$. In other words, with probability at least $1 - \frac{10-10c}{10-c}$, the value of x at the end of the current interval is at least $cl/10$.

In summary, for each interval, the event that the value of x at the end of the interval is at least $cl/10$ happens with constant probability where l is at least $\Omega(i/\log n)$. If the event does not happen, we move on to the beginning of the next interval. Since we have $\Omega(\log n)$ intervals, with high probability, the event happens at least once. Once the event happens, we use Hoeffding's inequality to guarantee that the value of x is $\Omega(l)$ after iteration i.

Now we prove each part of the sketch in detail.

- At the beginning of each interval, with high probability, within $O(\log n)$ iterations, x reaches a value that is at least x_0 : Let c be the maximum expected number of iterations for the variable x to reach x_0 from any initial value $x_{init} < x_0$. c is a constant depending on p. By Markov bound, we have that in every $2c$ iterations, with probability at least $1/2$, there exists at least one iteration before which x is at least x_0. Thus, in $O(\log n)$ iterations, with high probability, x reaches some value at least x_0 at least once.
- After reaching x_0, x will never go below x_0 in the current interval with a constant probability: We will prove a stronger proposition that with constant probability, x never goes below x_0 after arbitrary number of iterations. For this, we consider

another random process where a random variable y increases with probability $q \stackrel{\text{def}}{=} \frac{1}{2} + \frac{1-p}{2(3-p)}$ and decreases with probability $1-q = \frac{1}{2} - \frac{1-p}{2(3-p)}$. If x and y starts with the same value, the probability that x never goes below x_0 is lower bounded by the probability that y never goes below x_0, because for each value $v \geq x_0$, if both x and y start at v, $\Pr[x \text{ reaches } v+1 \text{ before } v-1] \geq \Pr[y \text{ reaches } v+1 \text{ before } v-1]$. (Equality holds when $p^+(v) = q$ and $p^-(v) = 1-q$.) Now we may consider the random process of y. Let $p_{i,j}$ ($i \in \{-1, 0\}, j \in \{0, 1\}$) denote the probability that y eventually reaches $x_0 + i$ starting with $y = x_0 + j$. Then we have $p_{-1,0} = q + (1-q)p_{-1,1}$. We also have $p_{-1,1} = p_{-1,0}p_{0,1} = p_{-1,0}^2$. These solve to $p_{-1,0} = \frac{1-\sqrt{1-4q(1-q)}}{2-2q}$ or $p_{-1,0} = 1$. The later case is impossible for the following reason: Let $p_{i,j}(d)$ be the probability that y eventually reaches $x_0 + i$ starting with $y = x_0 + j$ with in d iterations. Denote $\frac{1-\sqrt{1-4q(1-q)}}{2-2q}$ by r. $p_{-1,0}(1) = q \leq r$. If $p_{-1,0}(d) \leq r$, $p_{-1,0}(d+1) = q + (1-q)p_{-1,1}(d) \leq q + (1-q)p_{-1,0}(d)p_{0,1}(d) = q + (1-q)(p_{-1,0}(d))^2 \leq r$ where the last inequality is by the fact that r is a root of $q + (1-q)r^2 = r$.
$1 - r$ lower bounds the probability that x never goes below x_0 after arbitrary number of iterations.

- Conditioning on x never going below x_0, the expected increment of x in the current interval is at least cl for some constant c where l is the number of iterations remaining in the current interval: We know that for any $x \geq x_0$, $\mathbb{E}[next(x)] \geq x + \frac{1-p}{3-p}$. Setting $c = \frac{1-p}{3-p}$ completes the proof.
- Once the value of x is at least $cl/10$ at the end of some interval, x will be $\Omega(i/\log n)$ after iteration i: Consider the following $h = cl/20$ steps after the interval at the end of which x is at least $cl/10$. At each of the h steps, x increases by at least c in expectation because $x \geq cl/20$ during these steps. Let $x_{\texttt{before}}$ ($x_{\texttt{after}}$) be the value of x before (after) the h steps. We have $\mathbb{E}[x_{\texttt{after}}] = x_{\texttt{before}} + c^2 l/20$. We use Hoeffding's inequality to lower bound $x_{\texttt{after}}$.

$$\begin{aligned}
&\Pr[x_{\texttt{after}} < cl/10] \\
\leq &\Pr[|x_{\texttt{after}} - \mathbb{E}[x_{\texttt{after}}]| > \mathbb{E}[x_{\texttt{after}}] - cl/10] \\
\leq &\Pr\left[|x_{\texttt{after}} - \mathbb{E}[x_{\texttt{after}}]| > c^2 l/20\right] \\
\leq &\exp(-\Omega(l)).
\end{aligned}$$

In other words, with high probability, after the h steps, the value of x is still at least $cl/10$. We can repeat the argument for $i/h = O(\log n)$ times to conclude that after the first i iterations, the value of x is at least $cl/10$.

□

A.3 Omitted Algorithm in Section 5.5

To compute solutions for the 2PC problem (Problem 6), we design an iterative mechanism (Algorithm 3) for calling the search problem (Algorithm 1) properly, replacing Algorithm 2.

H is for storing the current optimal subgraph, initialized as $\emptyset$ (line 1). Our iterative strategy is related with the current optimal polarity value ρ. We initialize it as 0 (line 1) in the beginning.

We pick a starting vertex s at first, sampling from all vertices in $V(G)$ (line 2). Each turn, we execute a search process with specific

Algorithm 3: RH: Iterative mechanism for 2PC

Input: Signed graph G;
Output: $H \subseteq G$: the found subgraph with high 2PC polarity; ρ : the 2PC polarity of H;

1 $H \leftarrow \emptyset$; $\rho \leftarrow 0$;
2 $s \leftarrow$ Sample a vertex from $V(G)$;
3 **while** *true* **do**
4 $\quad \overline{H}, \hat{\rho} \leftarrow \texttt{Search}(G, \beta = \frac{1}{2}, \sigma = 0.9\rho, s)$; `// guided by the new object function` $\hat{\Phi}' = \hat{\Phi} - \sigma|S|$. `hyperparameters` T `and` p `are omitted.`
5 $\quad$ **if** $\hat{\rho} > \rho$ **then**
6 $\quad\quad \rho \leftarrow \hat{\rho}$;
7 $\quad\quad H \leftarrow \overline{H}$;
8 $\quad$ **else** $s \leftarrow$ Sample a vertex from $V(G)$;
9 $\quad$ **if** *meets stop-conditions* **then** break;
10 $\quad$ remove all vertice with degree less than ρ from G;
11 **return** H, ρ;

hyperparameters to try to find a new candidate answer (line 4). We chose the tolerance parameter β to be $\frac{1}{2}$. Since our new modified object function is $\hat{\Phi}' = \hat{\Phi} - \sigma|S|$, we also need to pass in σ as 0.9ρ. The search process will have to return the corresponding polarity $\hat{\rho}$ along with the found subgraph. We compare $\hat{\rho}$ with the previous found ρ. If it is better, we update ρ and H correspondingly (line 5 to 7). Note that we do not sample a new starting vertex each time. Instead, we also do so when the search process does not return a better result (line 8). We regard this as a sign showing that we have reached the limit of the current chosen starting vertex.

The whole process will stop when a certain condition is met (line 9). The detail is omitted and can be referred to in our provided implementation. There are a few optimizations of such iteration mechanisms that can help improve the performance. Note that if the current polarity is ρ, for any vertex with a degree less than ρ, removing them will not worsen the answer. Therefore, we remove all such vertices from G after the ρ gets updated (line 10).

B ADDITIONAL EXPERIMENTS

B.1 Stability Study

In our search process (Algorithm 1), we will execute either vertex deletion or color flipping in a certain probability p. We conduct some experiments to study how such non-incremental operations help improve the stability of our algorithm, shown in Table B.1.

We modify the algorithm RH to a version with only insertion operations, denoted as RH (insertion only). On all datasets, we execute both algorithms for 100 rounds. As common, we compute the variance σ^2 to measure the stability. Here, for each dataset, we also compute the maximum ($\max \hat{\Phi}$), minimum ($\min \hat{\Phi}$), and average (μ) values for tolerant balance count (TBC).

As shown in Table B.1, we can see with these two operations, the results are more stable and consistent, which can be observed in the rightmost column.

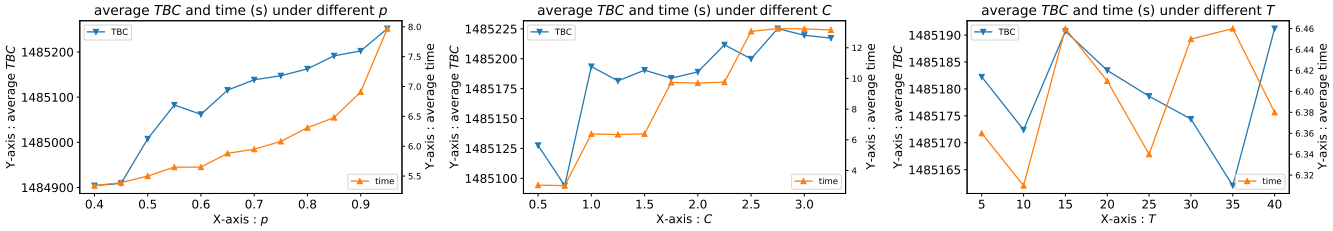

Figure B.1: We study the influence of hyperparameters (*i.e.,* Non-incremental probability p, Iteration constant C and Early stop turn limit T) on CONFLICT dataset with $\beta = 1/4$: For each hyperparameter, we execute the algorithm when setting the hyperparameter as various values while keeping other hyperparameters fixed. The results include the average Tolerant Balance Count (TBC) and running time in 25 rounds.

Table B.1: Stability study of Tolerant Balance Count (TBC) under $\beta = 1/8$ as example: We conduct 100 rounds of RH and RH (INSERTION ONLY) on each dataset. The results include maximum TBC ($\max \hat{\Phi}$), minimum TBC ($\min \hat{\Phi}$), mean TBC (μ), and the variance (σ^2) across the 100 rounds. Additionally, we present the variance reduction achieved by considering *non-incremental operations.*

Dataset	RH (INSERTION ONLY)				RH				Variance
	$\min \hat{\Phi}$	$\max \hat{\Phi}$	μ	σ^2	$\min \hat{\Phi}$	$\max \hat{\Phi}$	μ	σ^2	reduction
BITCOIN	14,864	15,523	15,499	4,977	15,578	15,619	15,604	25	**198**×
EPINIONS	462,237	471,120	470,536	1,301,027	472,158	472,240	472,195	315	**4,129**×
SLASHDOT	223,054	232,757	232,110	1,564,480	231,988	233,149	232,957	17,521	**88**×
TWITTER	187,549	209,701	208,051	30,265,009	197,072	209,701	209,097	4,455,495	**5.7**×
CONFLICT	1,284,391	1,289,332	1,288,405	430,597	1,292,253	1,293,741	1,293,318	84,706	**4**×
ELECTIONS	40,819	42,187	41,981	33,946	42,206	42,358	42,276	1,031	**29**×
POLITICS	470,347	470,764	470,576	4,289	470,534	470,905	470,642	4,042	**0.06**×

B.2 Statistical Hypothesis Testing

In our sampling strategy and performance analysis, we propose two hypotheses: 4.1 and 4.2. Here, we conduct some experiments on 3 datasets and 4 different β to verify if these hypotheses are reasonable.

For HYPOTHESIS 4.1, we compute the maximum $1-\epsilon$ such that there exists a subset $V' \subseteq V(G_{opt})$ with $\frac{|V'|}{|V(G_{opt})|} \geq \frac{1}{2}$ such that $\Phi(G_x, \beta) \geq (1-\epsilon)\Phi(G_{opt}, \beta), \forall x \in V'$. The result is shown in Table B.2. The hypothesis is supported by the fact that all values are very close to 1 and not sensitive to the value of β, which indicates ϵ's lower bound approaches to 0.

To verify HYPOTHESIS 4.2, we run Algorithm 1 from every possible starting vertex s (similar to RH (LS) in Section 5). For each setting, we compute $\max \frac{|V(G_b)|}{|V(G_a)|}$ for any two starting vertices a, b satisfying $\hat{\Phi}(G_a, \beta, \mathcal{X}_a) \geq \hat{\Phi}(G_b, \beta, \mathcal{X}_b)$. The result is shown in Table B.3, where all values are very close to 1 and far from the bound of 2 in the hypothesis. Therefore, Hypothesis 4.2 is also empirically supported.

Table B.2: Verifing HYPOTHESIS 4.1: For each dataset, we compute the maximum $1-\epsilon$ such that there exists a subset $V' \subseteq V(G_{opt})$ with $\frac{|V'|}{|V(G_{opt})|} \geq \frac{1}{2}$ such that $\Phi(G_x, \beta) \geq (1-\epsilon)\Phi(G_{opt}, \beta), \forall x \in V'$ under various β.

Dataset	$\beta = \frac{1}{4}$	$\beta = \frac{1}{8}$	$\beta = \frac{1}{16}$	$\beta = \frac{1}{32}$
BITCOIN	1.000	0.999	0.982	0.948
TWITTER	0.987	1.000	1.000	1.000
ELECTIONS	0.999	0.997	0.987	0.946

Table B.3: Verifing HYPOTHESIS 4.2: For each dataset, we compute $\max \frac{|V(G_b)|}{|V(G_a)|}$ for any two starting vertices a, b satisfying $\hat{\Phi}(G_a, \beta, \mathcal{X}_a) \geq \hat{\Phi}(G_b, \beta, \mathcal{X}_b)$ under various β.

Dataset	$\beta = \frac{1}{4}$	$\beta = \frac{1}{8}$	$\beta = \frac{1}{16}$	$\beta = \frac{1}{32}$
BITCOIN	1.018	1.014	1.020	1.037
TWITTER	1.053	1.111	1.094	1.084
ELECTIONS	1.009	1.032	1.040	1.061

B.3 Hyperparameter Analysis

For the three hyperparameters p, C, T in Algorithm 2, we conduct a series of experiments to demonstrate that our choice is natural, shown in Figure B.1.

For each hyperparameter, we fix the other two unchanged as our option in Section 4 and enumerate its value in a range. For each value, we compute the average runtime and TBC over 25 times the executions of solving on the CONFLICT with setting $\beta = \frac{1}{4}$. A larger p may lead to more non-incremental operations, thus a longer runtime is expected. Similarly, a larger C leads to more executions of the search process. Therefore, as p, C gets larger, the runtime and the average TBC might be getting larger as well. This can be observed from the experiment, shown in Figure B.1.

On the other hand, although such patterns can be observed, the difference between average runtime and TBC is not significant for all three hyperparameters. This indicates that our algorithm is robust and does not tune towards the dataset.