Efficient Maximal Balanced Clique Enumeration in Signed Networks

Zi Chen  
East China Normal University  
zchen@stu.ecnu.edu.cn

Long Yuan∗  
Nanjing University of Science and Technology  
longyuan@njust.edu.cn

Xuemin Lin  
University of New South Wales  
lxue@cse.unsw.edu.au

Lu Qin  
University of Technology, Sydney  
lu.qin@uts.edu.au

Jianye Yang  
Hunan University  
jyyang@hnu.edu.cn

ABSTRACT

Clique is one of the most fundamental models for cohesive subgraph mining in network analysis. Existing clique model mainly focuses on unsigned networks. In real world, however, many applications are modeled as signed networks with positive and negative edges. As the signed networks hold their own properties different from the unsigned networks, the existing clique model is inapplicable for the signed networks. Motivated by this, we propose the balanced clique model that considers the most fundamental and dominant theory, structural balance theory, for signed networks, and study the maximal balanced clique enumeration problem which computes all the maximal balanced cliques in a given signed network. We show that the maximal balanced clique enumeration problem is NP-Hard. A straightforward solution for the maximal balanced clique enumeration problem is to treat the signed network as two unsigned networks and leverage the off-the-shelf techniques for unsigned networks. However, such a solution is inefficient for large signed networks. To address this problem, in this paper, we first propose a new maximal balanced clique enumeration algorithm by exploiting the unique properties of signed networks. Based on the new proposed algorithm, we devise two optimization strategies to further improve the efficiency of the enumeration. We conduct extensive experiments on large real and synthetic datasets. The experimental results demonstrate the efficiency, effectiveness and scalability of our proposed algorithms.

KEYWORDS

Maximal Balanced Clique, Signed Network, Graph Algorithm

1 INTRODUCTION

With the proliferation of graph applications, research efforts have been devoted to many fundamental problems in analyzing graph data [15, 28, 36, 39, 49, 53, 55]. Clique is one of the most fundamental cohesive subgraph models in graph analysis, which requires each pair of vertices has an edge. Due to the completeness requirement, clique model owns many interesting cohesiveness properties, such as the distance of any two vertices in a clique is one, every one vertex in a clique forms a dominate set of the clique and the diameter of a clique is one [38]. As a result, clique model has wide application scenarios in social network mining, financial analysis and computational biology and has been extensively investigated for decades. Existing studies on clique mainly focus on the unsigned networks, i.e., all the edges in the graph share the same property [4, 12, 13, 51]. Unfortunately, relationships between two entities in many real-world applications have completely opposite properties, such as friend-foe relationships between users in social networks [11, 23], support-dissent opinions in opinion networks [25], trust-distrust relationships in trust networks [26] and partnership-antagonism in protein-protein interaction networks [35]. Modelling these applications as signed networks with positive and negative edges allows them to capture more sophisticated semantics than unsigned networks [1, 5, 10, 26, 32, 33]. Consequently, existing studies on clique ignoring the sign associated with each edge may be inappropriate to characterize the cohesive subgraphs in a signed network and there is an urgent need to define an exclusive clique model tailored for the signed networks.

For the signed networks, the most fundamental and dominant theory revealing the dynamics and construction of the signed networks is the structural balance theory [1, 5, 10, 11, 18, 19, 26, 32, 33]. The intuition underlying the structural balance theory can be described as the aphorisms: "The friend (resp. enemy) of my friend (resp. enemy) is my friend, the friend (resp. enemy) of my enemy
(resp. friend) is my enemy”. Specifically, a signed network \( G \) is structural balanced if \( G \) can be split into two subgraphs such that the edges in the same subgraph are positive and the edges between subgraphs are negative [18]. In a signed network, an imbalanced sub-structure is unstable and tends to evolve into a balanced state. Consider the graph \( G \) shown in Figure 1 (a). The negative edge between \( v_1 \) and \( v_2 \) makes \( G \) imbalanced. Closely observing \( G \), we can find that \( v_1 \) and \( v_2 \) have a mutual “friend” \( v_3 \) and mutual “enemies” \( v_4, v_5 \) and \( v_6 \). It means \( v_1 \) and \( v_2 \) share more common grounds than differences. According to structural balance theory, \( v_1 \) and \( v_2 \) tend to be allies as time goes by. \( G' \) shown in Figure 1 (b) is the evolved balanced counterpart of \( G \). In \( G' \), the sign of the edge between \( v_1 \) and \( v_2 \) becomes positive. \( \{v_1, v_2, v_3\} \) and \( \{v_4, v_5, v_6\} \) form two alliances and the edges in the same alliance are positive and the edges connecting different alliances are negative. As illustrated in this example, structural balance reflects the key characteristics of the signed networks.

According to the above analysis, clique model is a fundamental cohesive subgraph model in graph analysis and can be used in many applications, but there is no appropriate counterpart in the signed networks. Meanwhile, the structure of the signed networks is expected to be balanced based on the structure balance theory. Motivated by this, we propose a maximal balanced clique model in this paper. Formally, given a signed network \( G \), a maximal balanced clique \( C \) is a maximal subgraph of \( G \) such that (1) \( C \) is complete, i.e., every pair of vertices in \( C \) has an edge. (2) \( C \) is balanced, i.e., \( C \) can be divided into two parts such that the edges in the same part are positive and the edges connecting two parts are negative. This definition not only catches the essence of the clique model in the unsigned networks but also guarantees that a detected clique is stable in the signed networks. In this paper, we aim to devise efficient algorithms to enumerate all maximal balanced cliques in a given signed network.

Applications. Maximal balanced clique enumeration can be used in many applications, for example:

1. Opinion leaders detection in opinion networks. Opinion leaders are people who are active in a community capturing the most representative opinions in the social networks [44]. Maximal balanced clique enumeration can be used to detect opinion leaders in the opinion networks. In an opinion network, each vertex represents a user and there is a positive/negative edge between two vertices if one user support/dissent another user. A maximal balanced clique in an opinion network represents a group of users that any two of them have an opinion with each other and can be further divided into two subgroups such that the intra-group users support each other and the inter-group users dissent each other. Since these users actively involve in the opinion networks (every two of them have an opinion with each other) and have their clear standpoints (support everyone in the same group and dissent everyone in the opposite group), the users in the maximal balanced cliques are good candidates of opinion leaders in the opinion network.

2. Finding international alliances-rivalries groups. The international relationships between nations can be modeled as a signed network, where each vertex represents a nation, positive and negative edges indicate alliances and rivalries, respectively. Computing the maximal balanced cliques in such networks reveals hostile groups of allied forces, such as the Allied and Axis power during World War II or the North Atlantic Treaty Organization and the Warsaw Pact during the Cold War [3, 11]. We can extend it to find the alliances-rivalries commercial groups among business organizations similarly, such as [Pepsi, KFC] vs [Coke, McDonald][21].

3. Synonym and antonym groups discovery. In a word network, each vertex represents a word and there is a positive edge between two synonyms and a negative edge between two antonyms[34]. In such signed networks, our model can discover synonym groups that are antonymous with each other, such as, [interior, internal, intimate] and [away, foreign, outer, outside, remote]. These discovered groups may be further used in applications such as automatic question generation [24] and semantic expansion [22].

Contributions. In this paper, we make the following contributions:

1. The first work to study the maximal balanced clique model. We formalize the balanced clique model in signed networks based on the structural balance theory. To the best of our knowledge, this is the first work considering the structural balance of the cliques in signed networks. We also prove the NP-Hardness of the problem.

2. A new framework tailored for maximal balanced clique enumeration in signed networks. After investigating the drawbacks of the straightforward approach, we propose a new framework for the maximal balanced clique enumeration. Our new framework enumerates the maximal balanced cliques based on the signed network directly and its memory consumption is linear to the size of the input signed network.

3. Two effective optimization strategies to further improve the enumeration performance. We explore two optimization strategies, in-enumeration optimization and pre-enumeration optimization, to further improve the enumeration performance. The in-enumeration optimization can avoid the exploration for unpromising vertices during the enumeration while the pre-enumeration techniques can prune unpromising vertices and edges before enumeration.

4. Extensive performance studies on real and synthetic datasets. We conduct extensive experimental studies to evaluate the proposed algorithms on real and synthetic datasets, one of which contains 3 million vertices and 105 million edges. As shown in our experiments, the baseline approach only works on small datasets while our approach can complete the enumeration efficiently on both small and large datasets.

Outline. Section 2 reviews the related work. Section 3 provides preliminaries including the definition of balanced clique model and problem statement. Section 4 introduces the baseline algorithm. Section 5 presents our new enumeration framework. Section 6 shows several optimization techniques. Section 7 reports the results of experimental studies. Section 8 concludes our paper.

2 RELATED WORK

Signed network analysis. Signed network analysis has attracted much attention in the literature. In these works, the theories explaining the potential social dynamics process in signed networks have been extensively studied. Among these theories, **structural...**
balance theory is the most fundamental and dominant one [58]. Structural balance theory is originally introduced in [19] and generalized in the graph formation in [5, 18]. After that, structural balance theory is developed extensively [1, 10, 26, 32, 33]. In these works, it is interesting to mention that the authors in [32] model the evolving procedure of a signed network and theoretically prove that the network would evolve into a balanced clique when the mean value of the initial friendliness among the vertices $\mu \leq 0$. [58] provides a comprehensive survey on structural balanced theory.

Besides theories on signed networks, a large body of literature on mining signed networks has been emerged. Among them, the most closely related work to ours is [27] in which an ($a, k$)-clique model is proposed. Given a signed network $G$, an ($a, k$)-clique is defined as a maximal clique $C$ such that the negative degree for each vertex in $C$ is not greater than $k$ and the positive degree for each vertex in $C$ is not less than $ak$. Compared with our model, ($a, k$)-clique model only considers the amount of positive and negative edges in the clique and the structural balance of the clique is totally ignored, which makes ($a, k$)-clique model essentially different from our model. In [17], a $k$-balanced trusted clique model is proposed. A $k$-balanced trusted clique is defined as a clique with $k$ vertices consisting with positive edges only. Although the $k$-balanced trusted clique model has a similar name with our model, it ignores the negative edges in the clique, which means the information of the negative edges are totally missed.

Community detection in signed networks is also related to our work. For example, [8, 16, 29–31, 45] aim to find the antagonistic communities in a signed network. These works mainly focus on exploring several groups of dense subgraphs and most of them don’t have a clear structural definition of their community model, while our work aims to enumerate the clique structure in a signed network. Moreover, these solutions generally involve a complicated optimization procedure, thereby, they are hard to handle large signed networks, while our proposed algorithm is scalable to enumerate all the maximal balanced cliques in large signed networks with hundreds of millions of edges as verified in our experiments. A survey on signed network mining can be found in [46].

Clique on unsigned networks. Clique model is one of the most fundamental cohesive subgraph models. [4] proposes an efficient algorithm for maximal clique enumeration based on backtracking search,[2] first considers the memory consumption during the maximal clique enumeration. Based on [4], more efficient algorithms for maximal clique enumeration are investigated [12, 13, 47]. [12] proposes a novel branch pruning strategy, pivot pruning, which can efficiently reduce the search space by ignoring the search process from the neighbors of the pivot. [57] studies the maximal biclique enumeration problem on bipartite graphs. [57] keeps growing the vertex set in one side and peeling the vertex set in another side to enumerate the maximal bicliques. It also utilizes some techniques to further improve the enumeration performance, such as choosing vertex with small degree from candidate set to reduce the search tree depth and pruning vertices which may produce non-maximal bicliques. These techniques for biclique enumeration inspire our techniques presented in Section 6.1. [14] reviews recently advances in maximal clique enumeration. Based on clique, other cohesive subgraph models are also studied recently, such as $k$-core [43], $k$-truss [9, 20], $k$-edge connected component [52, 54, 59], (r, s)-nuclei [40, 41]. Note that our balanced clique model is different from the existing cohesive subgraph models on unsigned networks and it cannot be well solved by the existing works. If we just consider the positive edge in the signed network and use the traditional methods on unsigned networks for community detection, the found results would ignore the negative edges and half meaningful information in the signed network is lost.

3 PROBLEM STATEMENT

In this paper, we consider an undirected and unweighted signed network $G = (V, E^+, E^-)$, where $V$ denotes the set of vertices, $E^+$ denotes the positive edges and $E^-$ denotes the negative edges connecting the vertices in $G$. We denote the number of vertices and number of edges by $n$ and $m$, respectively, i.e., $n = |V|$ and $m = |E^+| + |E^-|$. For each vertex $v \in G$, let $N^+_G(v)$ represents the positive neighbors of $v$, i.e., $N^+_G(v) = \{u | (v, u) \in E^+ \land u, v \in V\}$, and let $N^-_G(v)$ represents the negative neighbors of $v$, i.e., $N^-_G(v) = \{u | (v, u) \in E^- \land u, v \in V\}$. We use $d^+_G(v)$ and $d^-_G(v)$ to denote the positive and negative degree of $v$, respectively, i.e., $d^+_G(v) = |N^+_G(v)|$ and $d^-_G(v) = |N^-_G(v)|$. We also use $N_G(v)$ and $d_G(v)$ to denote the neighbors and degree of $v$, i.e., $N_G(v) = N^+_G(v) \cup N^-_G(v)$ and $d_G(v) = d^+_G(v) + d^-_G(v)$. For simplicity, we omit $G$ in the notations if the context is self-evident.

Definition 3.1. (Balanced Network [18]) Given a signed network $G = (V, E^+ , E^-)$, it’s balanced iff it can be split into two subgraphs $G_L$ and $G_R$, s.t. $\forall (u, v) \in E^+ \rightarrow u, v \in G_L$ or $u, v \in G_R$, and $\forall (u, v) \in E^- \rightarrow u \in G_L, v \in G_R$ or $u \in G_R, v \in G_L$.

Definition 3.2. (Maximal Balanced Clique) Given a signed network $G = (V, E^+, E^-)$, a maximal balanced clique $C$ is a maximal subgraph of $G$ that satisfies the following constraints:

- Complete: $C$ is complete, i.e., $\forall u, v \in C \rightarrow (u, v) \in E^+ \cup E^-$. 
- Balanced: $C$ is balanced, i.e., it can be split into two subcliques $C_L$ and $C_R$, s.t. $\forall u, v \in C_L$ or $u, v \in C_R \rightarrow (u, v) \in E^+$, and $\forall u \in C_L, v \in C_R$ or $u \in C_R, v \in C_L \rightarrow (u, v) \in E^-$. 

In this paper, we aim to enumerate all maximal balanced cliques in a given signed network. Since many real applications require that the number of vertices in $C_L$ and $C_R$ is not less than a fixed threshold, we add a size constraint on $|C_L|$ and $|C_R|$ s.t. $|C_L| \geq k$ and $|C_R| \geq k$. With the size constraint, users can control the size of the returned maximal balanced cliques based on their specific requirements. We formalize the studied problem as follows:

Problem Statement. Given a signed network $G$ and an integer $k$, maximal balanced clique enumeration (MBCE) computes all the maximal balanced cliques $C$ in $G$ s.t. $|C_L| \geq k$ and $|C_R| \geq k$. 

Example 3.3. Consider the signed network $G$ in Figure 2 in which positive/negative edges are denoted by solid/dashed lines. Assume $k = 2$, there are 4 maximal balanced cliques in $G$, namely, $C_1 = \{(v_9, v_1, v_3), \{v_5, v_6, v_7\}\}$, $C_2 = \{(v_9, v_1, v_2, v_3), \{v_5, v_6\}\}$, $C_3 = \{(v_9, v_1, v_2), \{v_5, v_6, v_8\}\}$, $C_4 = \{(v_9, v_1, v_2, v_3, v_{15})\}$, where vertices in $C_L$ and $C_R$ are marked with different colors. Take $C_4$ as an example, it is complete as any two vertices in $C_4$ have an edge. It is balanced as it can be split into $\{(v_9, v_{14})\}$ and $\{(v_{13}, v_{15})\}$, and two positive edges $(v_9, v_{14})$ and $(v_{13}, v_{15})$ exist in $E^+$. $v_9$ has negative
edges to \(v_{13}, v_{15}\) and similar negative edges exist for \(v_{13}, v_{14}\) and \(v_{15}\). \(C_4\) is maximal because no more vertices can be added into it to make it complete and balanced.

**Problem Hardness.** The MBCE problem is NP-Hard, which can be proved following the NP-Hardness of maximal clique enumeration problem [6, 42]. Given an unsigned network \(G = (V, E)\), we can transfer \(G\) to a signed network \(G'\) as follows: we first keep all the vertices of \(G\) and all the edges of \(G\) as positive edges in \(G'\); then, we add a new vertex \(v\) to \(G'\) and connect \(v\) to all the remaining vertices in \(G'\) with negative edges. It’s clear that each maximal clique \(C\) in \(G\) corresponds a maximal balanced clique \(\{\{v\}, C\}\) in \(G'\) (assume \(k = 1\)), and vice versa, which means the maximal clique enumeration problem in \(G\) can be reduced to the MBCE problem in \(G'\). As the maximal clique enumeration problem is NP-Hard [6, 42], our problem is also NP-Hard.

4 A BASELINE ALGORITHM

We first propose a baseline algorithm to address MBCE problem based on existing methods for maximal clique enumeration [13] and maximal biclique enumeration [57] in unsigned networks. For a signed network \(G = (V, E^+, E^-)\), we can treat it as the combination of two unsigned networks \(G^+ = (V, E^+)\) and \(G^- = (V, E^-)\). For any maximal balanced clique \(C = \{C_L, C_R\}\) in \(G\), it is clear that \(C_L\) (resp. \(C_R\)) is a clique in \(G^+\) and the subgraph induced by vertices in \(C_L\) and \(C_R\) in \(G^+\) is a biclique. Therefore, we can enumerate the maximal balanced cliques in \(G\) in two steps: 1) compute all the maximal cliques in \(G^+\) with [13]; 2) for each pair of the computed maximal cliques \(C_i\) and \(C_j\) in \(G^+\), compute the maximal bicliques in the bipartite subgraph induced by the vertices in \(C_i\) and \(C_j\) in \(G^+\). According to [57], the returned maximal bicliques in \(G^+\) are the maximal balanced cliques in \(G\). The pseudocode of Baseline solution is shown in Algorithm 1. As the pseudocode is self-explained, we omit the description. Note that although all the maximal cliques in \(G^+\) are enumerated in line 1 of Algorithm 1, Algorithm 1 does not require that the two component cliques of a maximal balanced clique are maximal in \(G^+\). Algorithm 1 just considers all maximal cliques as candidate subgraphs for further processing in step 2.

**Example 4.1.** Consider \(G\) in Figure 2, assume \(k = 2\), Baseline first enumerates all the maximal cliques in \(G^+\) with size not less than 2, such as \(\{v_6, v_1, v_2, v_3\}\), \(\{v_5, v_6, v_7\}\), \(\{v_5, v_6, v_8\}\), \(\{v_6, v_{14}\}\), \(\{v_{11}, v_{15}\}\). After that, for each pair of computed maximal cliques, computes the maximal bicliques in the induced bipartite subgraph in \(G^+\).

**Algorithm 1** Baseline\((G = (V, E^+, E^-), k)\)

1. enumerate the maximal cliques in \(G^+ = (V, E^+)\) with size not less than \(k\) by [13];
2. for each pair of computed maximal cliques \(C_i\) and \(C_j\) do
3. enumerate the maximal bicliques in the bipartite subgraph induced by \(C_i\) and \(C_j\) in \(G^- = (V, E^-)\) with size not less than \(k\) for both two parts by [57];
4. remove the duplicate bicliques computed in line 3;

Take \(\{v_6, v_1, v_2, v_3\}\) and \(\{v_5, v_6, v_7\}\) as an example, the maximal biclique in the induced bipartite subgraph in \(G^-\) are \(\{\{v_6, v_1, v_3\}, \{v_5, v_6, v_7\}\}\) and \(\{\{v_5, v_1, v_2, v_3\}, \{v_5, v_8\}\}\) which correspond \(C_i\) and \(C_j\) in \(G\), respectively. The remaining maximal balanced cliques can be enumerated similarly.

**Theorem 4.2.** Given a signed network \(G\), Baseline enumerates all the maximal balanced cliques in \(G\) correctly.

**Proof.** We first prove that all the maximal balanced cliques in \(G\) are found. Based on Definition 3.2, if there exists a maximal balanced clique \(C = \{C_L, C_R\}\) in \(G\), the vertices in \(C_L\) (resp. \(C_R\)) must be contained in a maximal cliques \(C_L'\) (resp. \(C_R'\)) in \(G^+\). As Baseline considers all the maximal cliques in \(G^+\), \(C_L\) and \(C_R\) are not missed in step 1. Following Definition 3.2, \(C_L\) and \(C_R\) form a maximal biclique in \(G^-\). In step 2, Baseline enumerates all the maximal bicliques in the induced subgraph in \(G^-\) by every pair of enumerated maximal cliques in step 1. Thus, Baseline can find all the maximal balanced cliques in \(G\). Moreover, as a maximal balanced clique maybe contained in multiple pairs of maximal cliques, Baseline removes all the duplicates in line 4. Therefore, Baseline outputs each maximal balanced clique once. The theorem is proved.

**Drawbacks of baseline.** Since Baseline does not consider the uniqueness of the signed networks and processes MBCE with the techniques for the unsigned networks, it has two drawbacks:

- **Memory consumption.** Baseline has to store all the maximal cliques in \(G^+\) in memory. The number of maximal cliques could be exponential to the number of vertices [12], which makes Baseline unable to handle large networks.
- **Efficiency.** In baseline, all the maximal cliques in \(G^+\) are enumerated and every pair of maximal cliques are explored. The time complexity of Baseline is \(O(T_{CB} + \eta^2 \cdot T_{BCB})\), where \(T_{BCB}\) represent the time complexity of maximal (bi)clique enumeration, and \(\eta\) is the number of enumerated maximal cliques in \(G^-\). Considering the maximal (bi)clique enumeration is time-consuming and the number of maximal cliques could be very large, it is inefficient for MBCE problem.

5 A NEW ENUMERATION FRAMEWORK

Revisiting baseline, the root leading to its drawbacks discussed above is that it treats the signed network as a specific combination of two unsigned networks and utilizes the existing techniques designed for the unsigned networks. Therefore, we have to explore new techniques by considering the uniqueness of signed networks to overcome the drawbacks of Baseline and improve the efficiency.
Algorithm 2 MBCEnum($G = (V, E^+, E^-)$, $k$)

1: Flag ← true;
2: for each $v_i \in \{v_0, v_1, \ldots, v_{n-1}\} \in V$ do
3:    $C_L \leftarrow \{v_i\}$, $C_R \leftarrow \emptyset$
4:    $P_L \leftarrow N^+_G(v_i) \cap \{v_1, v_2, \ldots, v_{n-1}\}$;
5:    $P_R \leftarrow N^-_G(v_i) \cap \{v_1, v_2, \ldots, v_{n-1}\}$;
6:    $Q_L \leftarrow N^+_G(v_i) \cap \{v_0, \ldots, v_{i-1}\}$;
7:    $Q_R \leftarrow N^-_G(v_i) \cap \{v_0, \ldots, v_{i-1}\}$;
8:    MBCEnumUtil($C_L$, $C_R$, $P_L$, $P_R$, $Q_L$, $Q_R$);
9: Procedure MBCEnumUtil($C_L$, $C_R$, $P_L$, $P_R$, $Q_L$, $Q_R$)
10: if $P_L = \emptyset$ and $P_R = \emptyset$ and $Q_L = \emptyset$ and $Q_R = \emptyset$ then
11:    output $C = \{C_L, C_R\}$;
12:    return
13: Flag ← false;
14: if Flag then
15:    for each $v \in P_L$ do
16:        MBCEnumUtil($C_L \cup \{v\}$, $C_R$, $P_L \setminus \{v\}$, $P_R$, $N^+_G(v) \cap \{v_0, \ldots, v_{i-1}\}$, $Q_L \setminus \{v\}$);
17:        for each $v \in P_R$ do
18:            MBCEnumUtil($C_L$, $C_R \cup \{v\}$, $P_L$, $P_R \setminus \{v\}$, $N^-_G(v) \cap \{v_0, \ldots, v_{i-1}\}$, $Q_R \setminus \{v\}$);
19:        else
20:            return
21: line 19-21; line 16-18;

of the enumeration. In this section, we present a new enumeration framework which aims to address the memory consumption problem. In next section, we further optimize the enumeration framework to improve the efficiency.

**Lemma 5.1.** Given a signed network $G$, for a balanced clique $C = \{C_L, C_R\}$ in $G$, if there is a vertex $v$ in $G$ such that $\forall u \in C_L \rightarrow (v, u) \in E^+$ and $\forall v \in C_R \rightarrow (v, w) \in E^-$, then $C' = \{C_L \cup \{v\}, C_R\}$ is also a balanced clique in $G$.

**Proof.** It can be proved following Definition 3.2 directly. □

According to Lemma 5.1, if we maintain a balanced clique $C = \{C_L, C_R\}$, let $P_L$ be the set of vertices that are positive neighbors of all the vertices in $C_L$ and negative neighbors of all the vertices in $C_R$, let $P_R$ be the set of vertices that are positive neighbors of all the vertices in $C_R$ and negative neighbors of all the vertices in $C_L$, we can enlarge $C$ by adding vertices from $P_L$ and $P_R$ into $C_L$ and $C_R$, respectively. Furthermore, if we update the $P_L$ and $P_R$ based on the new $C_L$ and $C_R$ accordingly and repeat the above enlargement procedure, we can obtain a maximal balanced clique when no more vertices can be added into $C_L$ or $C_R$.

**Algorithm.** Following the above idea, our algorithm for MBCE is shown in Algorithm 2. For each vertex $v_i$ in $G$ (line 2), we enumerate all the maximal balanced cliques containing $v_i$ (line 3-8). Note that $v_0, v_1, \ldots, v_{n-1}$ are in the degeneracy order [48] of $G$. We use $C_L$ and $C_R$ to maintain the balanced clique, which are initialized with $v_i$ and $\emptyset$, respectively (line 3). Similarly, we also initialize $P_L$ and $P_R$ as discussed above (line 4-5). Moreover, we use $Q_L$ and $Q_R$ to record the vertices that have been processed to avoid outputting duplicate maximal balanced cliques (line 6-7). After initializing these six sets,
Branch pruning aims to prune the unfruitful search states during the enumeration. At $S_1$, we assume that we have a balanced clique $C = \{C_L, C_R\}$ at this state. We first grow search branch by adding $v_2$ from $P_L$ into $C_L$. Since $v_2$ and $v_3$ are not $v_2$'s negative neighbors, they are removed from $P_R$ at $S_2$. Because $P_R$ is empty at $S_2$, we keep expending $C_L$ by adding $v_3$ from $P_L$. At $S_3$, $P_L, R_L, Q_L$ and $Q_R$ are empty. We obtain a maximal balanced clique $C_2 = \{\{v_0, v_1, v_2, v_3\}, \{v_5, v_6, v_7\}\}$ and this search branch starts from $v_2$ finishes. We return back to $S_1$ and $v_2$ is moved to $Q_1$. Then, we add $v_3$ into $C_L$ at $S_4$ and add $v_7$ into $C_R$ at $S_5$ and obtain $C_1 = \{\{0, 1, 3\}, \{5, 6, 7\}\}$. The search continues in a similar way until all the vertices in $P_L$ and $P_R$ at $S_1$ are explored.

Based on Algorithm 2, it is clear that the memory consumption of our enumeration framework is linear to the size of the input signed network. Therefore, the drawback of large memory consumption in Baseline is avoided.

6 OPTIMIZATION STRATEGIES

Although Algorithm 2 addresses the memory consumption problem in MBCE, the efficiency of Algorithm 2 is disappointing. In this section, we present two optimization strategies, namely in- enumeration optimization and pre-enumeration optimization, to further improve the efficiency of the enumeration.

6.1 In-Enumeration Optimization

Branch Pruning. Branch pruning aims to prune the unfruitful branches in the search tree of Algorithm 2 to improve the performance.

Pivot Choosing. Consider the maximal balanced clique search procedure of Algorithm 2, assume that we currently have $C_L, C_R, P_L$ and $P_R$, and we add a vertex $v$ from $P_L$ to $C_L$ in line 17. After finishing the search starting from $v$, we do not need to further explore the positive neighbors of $v$ in the for loop of line 16 and the negative neighbors of $v$ in the for loop of line 19. The reasons are as follows: w.o.l.g, let $v'$ be a positive neighbor of $v$, although we skip the maximal balanced clique search starting from $v'$, these maximal balanced cliques containing $v'$ must be explored by the searching branches starting $v$ or neighbors of $v'$. Therefore skipping the search starting from $v$'s neighbors does not affect the correctness of Algorithm 2.

In this paper, to maximize the benefits of pivot technology, we define the local degree for a vertex $v \in P_L \cup Q_L \cup P_R \cup Q_R$ as $d_L(v) = |N^+(v) \cap C_L| + |N^-(v) \cap C_R|$, and we choose the vertex $v$ that satisfies $\max_{v \in V'} (d_L(v))$ as the pivot, where $V' = P_L \cup P_R \cup Q_L \cup Q_R$.

Candidate Selection. In the search procedure of Algorithm 2, heuristically, search starting from a vertex with small local degree will have a short and narrow search branch, which means the search starting from the vertex will be finished very fast. Moreover, due to the search finish of the vertex, the vertex will be added into the excluded set and it can be used to further prune other search branches. Therefore, instead of adding vertices from $P_L$ and $P_R$ into $C_L$ and $C_R$ randomly in line 16 and 19 of Algorithm 2, we add vertices in the increasing order of their local degrees.

---

**Algorithm 3** MBCEUtil\(^*(G = (V, E^+, E^-), k)\)

1: line 1-7 of Algorithm 2;
2: MBCEUtil\(^*(C_L, C_R, P_L, P_R, Q_L, Q_R)\);
3: Procedure MBCEUtil\(^*(C_L, C_R, P_L, P_R, Q_L, Q_R)\):
4: line 10-13 of Algorithm 2:
5: if $|C_L| + |P_L| < k$ or $|C_R| + |P_R| < k$ then
6: return $; // ET Rule 1
7: if $\exists v \in Q_L$, s.t., $P_L \subseteq N_{G_L}(v)$ and $P_R \subseteq N_{G_R}(v)$ or $\exists v \in Q_R$, s.t., $P_R \subseteq N_{G_L}(v)$ and $P_L \subseteq N_{G_R}(v)$ then
8: return $; // ET Rule 2
9: if $\forall v \in Q_L$, s.t., $P_L \subseteq \{p_L \cup N_{G_R}(p_L)\}$ and $P_R \subseteq N_{G_L}(p_L)$ and $\forall v \in P_R$, s.t., $P_R \subseteq \{p_R \cup N_{G_L}(p_R)\}$ and $P_L \subseteq N_{G_R}(p_R)$ then
10: output $C = \{C_L \cup P_L, C_R \cup P_R\};$
11: if $\text{Flag}$ then
12: Flag $\leftarrow \text{false};$
13: $p \leftarrow \text{argmax}_{v \in P_L \cup P_R \cup Q_L \cup Q_R} (d_L(v)); // Pivot Choosing
14: $\text{newP}_L \leftarrow P_L \cup \{p\} \cup N_{G_R}(p);$
15: $\text{newP}_R \leftarrow P_R \cup N_{G_L}(p);$
16: $\text{sort}(\text{newP}_L); \text{sort}(\text{newP}_R); // Candidate Selection
17: if $\text{Flag}$ then
18: for each $v \in \text{newP}_L$
19: line 17-18 replacing MBCEUtil with MBCEUtil\(^*\);
20: for each $v \in \text{newP}_R$
21: line 20-23 replacing MBCEUtil with MBCEUtil\(^*\);

**Early Termination.** We consider different conditions that we can terminate the search early in Algorithm 2. For a balanced clique $C = \{C_L, C_R\}$, the maximal possible size of $C_L$ ($C_R$) for the final maximal balanced clique is $|C_L| + |P_L|$ ($|C_R| + |P_R|$). Based on the size constraint of $k$, we have the following rules:

- **ET Rule 1:** If $|C_L| + |P_L| < k$ or $|C_R| + |P_R| < k$, we can terminate current search directly.

In Algorithm 2, we use $Q_L$ and $Q_R$ to store such vertices that the maximal balanced cliques containing them have been enumerated. Therefore, during the enumeration, if there exists a vertex $v \in Q_L(Q_R)$ such that $P_L(P_R) \subseteq N_{G_L}(v)$ ($P_L(P_R) \subseteq N_{G_R}(v)$), then we can conclude that the maximal balanced cliques have been enumerated. Following this, we have our second rule:

- **ET Rule 2:** If $\exists v \in Q_L$, s.t., $P_L \subseteq N_{G_L}(v)$ and $P_R \subseteq N_{G_R}(v)$ or $\exists v \in Q_R$, s.t., $P_R \subseteq N_{G_L}(v)$ and $P_L \subseteq N_{G_R}(v)$, then we can terminate current search directly.

In a certain search of Algorithm 2, if all the vertices in $P_L (P_R)$ consist a clique formed by positive edges and every vertex in $P_L (P_R)$ has negative edges to all the vertices in $P_L (P_R)$, then $P_L$ and $P_R$ consist a balanced clique. Then, based on Definition 3.2, $C_L \cup P_L$ and $C_R \cup P_R$ consist a maximal balanced clique. Therefore, we have our third early termination rule:

- **ET Rule 3:** If $\forall v \in Q_L$, s.t., $P_L \subseteq \{p_L \cup N_{G_R}(p_L)\}$ and $P_R \subseteq N_{G_L}(p_L)$ and $\forall v \in P_R$, s.t., $P_R \subseteq \{p_R \cup N_{G_L}(p_R)\}$ and $P_L \subseteq N_{G_R}(p_R)$, we can output $C = \{C_L \cup P_L, C_R \cup P_R\}$ and terminate current search directly.

Note that, in order to avoid outputting duplicate maximal balanced cliques, ET Rule 3 must be applied after ET Rule 2.
Algorithm. The maximal balanced clique enumeration algorithm with in-enumeration optimization strategies is shown in Algorithm 3. Since the pseudocode is self-explained, we omit the detailed description here.

**Theorem 6.1.** Given a signed network $G$, the time complexity of Algorithm 3 to enumerate the maximal balanced cliques in $G$ is $O(n \cdot 3^{\sigma/3})$, where $\sigma$ is the degeneracy number of $G$.

Proof. Given a graph $G$, the degeneracy number of $G$ is $\sigma$. Let $P = P_L \cup P_R$, $Q = Q_L \cup Q_R$, we first prove the size constraint for $P$. In line 2 of Algorithm 2, we iterate $v_i$ in the degeneracy order of $G$ and vertices with a lower order than $v_i$ are not included in $P$. Therefore, for $P$ regarding $v_i$, we have $|P| \leq n$. Then, we analyse the time complexity of MBCEnumUtil in detail. ET Rule 1 can be done in $O(1)$ time. For ET Rule 2, we need to get local neighbors (within $P$) for each vertex in $Q$, it costs $O(|Q||P|)$ time. Similarly, for ET Rule 3, the time complexity for getting local neighbors for vertices in $P$ is $O(|P|^2)$. Moreover, pivot selection and candidates sort consume $O(|P| + |Q|)$ time and $O(|P| \log |P|)$ time, respectively, based on above computation. So far, the time complexity is $O(|P|^2 + |Q|)$. And because each recursion for MBCEnumUtil can invoke at most $|P|$ further recurrance, so the further time complexity is $O(|P|^3 + |Q|)$. Now, we formulate the time complexity function for MBCEnumUtil with parameters $|P|$ and $|Q|$, namely $T(|P|, |Q|) = \max \{\sum_{i=1}^{m}T(|P_i|, |Q_i|) + |P|^2(|P| + |Q|), \}$, note that $P_i$ is the new candidates reduced by neighbors of $i$-th vertex in the rest candidates. As we choose a vertex with maximum pruning size as pivot, we get $|P_i| < |P| - s$ where $s$ is the size of the rest candidates after pivot pruning. Hence we get $T(|P|, |Q|) \leq \max \{sT(|P| - s, |Q|) + |P|^2(|P| + |Q|), \}$, it’s proved that $T(|P|, |Q|)$ can be bounded by $O((\sigma + |Q|) \cdot 3^{\sqrt[3]{|P|/3}})$. Sum the time of $T(|P|, |Q|)$ for $n$ vertices, the finally time complexity is $\sum_{i=1}^{n}[(\sigma + |Q|) \cdot 3^{\sqrt[3]{|P_i|/3}}] = O((\sigma n + m) \cdot 3^{\sigma/3}) = O(\sigma n \cdot 3^{\sigma/3})$ due to the size constraint of $P$. □

6.2 Pre-Enumeration Optimization

In pre-enumeration optimization, we aim to remove the unpromising vertices and edges that not contained in any maximal balanced cliques based on their structural information. We explore two optimization strategies based on the neighbors of a vertex and the common neighbors of an edge.

**Vertex Reduction.** To reduce the size of a signed network, we first consider the neighbors of each vertex $v$, i.e., $N_C^G(v)$ and $N_G(v)$ to remove the unpromising vertices. We first define:

**Definition 6.2.** $(l, r)$-signed core Given a signed network $G = (V, E^+, E^-)$, two integers $l$ and $r$, a $(l, r)$-signed core is a maximal subgraph $C$ of $G$, s.t., $\min_{v \in C} |d^+_C(v)| = l$, $\min_{v \in C} |d^-_C(v)| = r$.

**Lemma 6.3.** Given a signed network $G$ and threshold $k$, a maximal balanced clique satisfying the size constraint with $k$ is contained in a $(k-1, k)$-signed core.

**Proof.** We can prove it by contradiction. Assume there is a vertex $v$ in a maximal balanced $G$ satisfying the size constraint with $\sigma$. Given a graph $G$, its degeneracy number, namely $\sigma$, is the least $d$ such that the vertices of $G$ can be arranged in a sequence so that each vertex is adjacent to at most $d$ of the vertices that follow it in the sequence [48].

**Algorithm 4 Vertex Reduction**

```
1: while $\exists v \in V$, s.t. $d^+_C(v) < k - 1$ or $d^-_C(v) < k$ do
2: for each $u \in N^+_C(v)$ do
3: $d^+_C(u) \leftarrow d^+_C(u) - 1$;
4: for each $u \in N^-_C(v)$ do
5: $d^-_C(u) \leftarrow d^-_C(u) - 1$;
6: $G \leftarrow G \setminus v$;
```

**Figure 4: Vertex Reduction and Edge Reduction**

$k$ but not in a $(k-1, k)$-signed core. Based on Definition 3.2, the positive degree of $v$ in $C$ is not less than $k - 1$ and the negative degree of $v$ in $C$ is not less than $k$. This contradicts with our assumption. Thus, the lemma holds. □

Therefore, in order to compute the maximal balanced cliques in a given signed network $G$ with integer $k$, we only need to compute the maximal balanced cliques in the corresponding $(k - 1, k)$-signed core of $G$. The remaining problem is how to efficiently compute the $(k - 1, k)$-signed core. We propose a linear algorithm to address this problem, which is shown in Algorithm 4.

**Algorithm.** Based on Definition 6.2, to compute the $(k - 1, k)$-signed core in the signed network $G$, we only need to identify the vertices like $v$ with $d^+_C(v) < k - 1$ or $d^-_C(v) < k$ and remove them from $G$. Due to the removal of such vertices, more vertices will be with positive degree less than $k - 1$ or negative degree less than $k$, we can further remove these vertices until no such kind of vertices exist in $G$. Following this idea, in Algorithm 4, we first identify a vertex $v$ with $d^+_C(v) < k - 1$ or $d^-_C(v) < k$ (line 1). Since $v$ will be removed from $G$, we decrease the positive degree by 1 for each positive neighbor of $v$ (line 2-3) and decrease the negative degree by 1 for each negative neighbor of $v$ (line 4-5). Then, we remove $v$ from $G$ (line 6). The algorithm terminates when no vertex with $d^+_C(v) < k - 1$ or $d^-_C(v) < k$ exists in $G$ (line 1). It is clear that Algorithm 4 correctly computes the $(k - 1, k)$-signed core of $G$. And we have the following theorem regarding its efficiency.

**Theorem 6.4.** Given a signed network $G$ and an integer $k$, the time complexity of Algorithm 4 is $O(n + m)$.

Proof. In Algorithm 4, we use a queue to store vertices that should be removed in line 6. Since every vertex is pushed in and popped from the queue at most once, the total processing time for this part is $O(n)$. Moreover, when a vertex is removed, we have to update the degrees for their neighbors once, the total time cost is $O(m)$. Therefore, the time complexity of Algorithm 4 is $O(n + m)$. □

**Example 6.5.** Let $k = 2$, Figure 4 shows an example of vertex reduction by Algorithm 4 on the signed network $G$ in Figure 2.
Algorithm 5 EdgeReduction\((G = (V, E^+, E^-), k)\)

1. for each \((u, v) \in E^+\) do
   2. compute \(\delta_{G}^+(u, v)\) and \(\delta_{G}^-(u, v)\);
   3. for each \((u, v) \in E^-\) do
      4. compute \(\delta_{G}^+(u, v)\) and \(\delta_{G}^-(u, v)\);
   5. Stop -- false;
5. while Stop = false do
   6. if \(\exists (u, v) \in E^+\) s.t. \(\delta_{G}^+(u, v) < k - 2\) or \(\delta_{G}^-(u, v) < k\) then
     7. \(G \leftarrow G \setminus (u, v)\);
   8. for each \(w \in V\) do
      9. for each \((u, v) \in E^+\) and \((u, w) \in E^-\) do
         10. \(\delta_{G}^+(u, w) \leftarrow \delta_{G}^+(u, w) - 1\);
         11. for each \((u, v) \in E^-\) do
             12. \(\delta_{G}^+(u, v) \leftarrow \delta_{G}^+(u, v) - 1\);
             13. \(\delta_{G}^+(w, v) \leftarrow \delta_{G}^+(w, v) - 1\);
             14. \(\delta_{G}^+(v, w) \leftarrow \delta_{G}^+(v, w) - 1\);
   15. Stop -- false;
   16. if \(\exists (u, v) \in E^-\) s.t. \(\delta_{G}^+(u, v) < k - 1\) or \(\delta_{G}^-(u, v) < k - 1\) then
     17. \(G \leftarrow G \setminus (u, v)\);
   18. for each \(w \in V\) do
      19. for each \((u, v) \in E^-\) and \((u, w) \in E^+\) do
         20. \(\delta_{G}^-(u, w) \leftarrow \delta_{G}^-(u, w) - 1\);
         21. \(\delta_{G}^-(u, v) \leftarrow \delta_{G}^-(u, v) - 1\);
         22. \(\delta_{G}^-(w, v) \leftarrow \delta_{G}^-(w, v) - 1\);
         23. \(\delta_{G}^-(v, w) \leftarrow \delta_{G}^-(v, w) - 1\);
   24. Stop -- false;

Theorem 8.7. Given a signed network \(G = (V, E^+, E^-)\), an integer \(k\), the time complexity of Algorithm 5 is \(O(m^{1.5})\).

Proof. We first prove that the time complexity of line 1-2 of Algorithm 5 can be bounded by \(O(m^{1.5})\), which follows the idea in [7]. Let \(d_G(v) = d_G^+(v) + d_G^-(v)\) and assume that \(d_G(u) < d_G(v)\). To compute the \(\delta_{G}^+(u, v)\), we can do it as follows: for each \(v \in N_G(u)\), we check whether \(v, w) \in E^-\). If \((v, w) \in E^-\) then \(\delta_{G}^+(u, v)\) increases by 1. Otherwise, \(\delta_{G}^+(u, v)\) keep the same. In this way, the time complexity of line 1-2 to compute \(\delta_{G}^+(u, v)\) is \(O(\sum_{v \in V} \sum_{u \in V} \delta^+(u, v) \cdot |N_G(u)|)\) can be bounded by \(O(\sum_{v \in V} \sum_{u \in V} \delta^+(u, v) \cdot |N_G(u)|)\), where \(|N_G(u)|\) is the number of positive neighbors \(u\) s. t. \(d_G(u) < d_G(v)\). Obviously, \(O(\sum_{v \in V} \sum_{u \in V} \delta^+(u, v) \cdot |N_G(u)|)\) can be bounded by \(O(\sum_{v \in V} \sum_{u \in V} \delta^+(u, v) \cdot |N_G(u)|)\), where \(|N_G(u)|\) is the number of neighbors \(v\) s. t. \(d_G(u) < d_G(v)\). Therefore, the time complexity of line 1-2 of Algorithm 5 can be bounded by \(O(m^{1.5})\). This is because if \(d_G(u) \leq \sqrt{m}\), \(|N_G(u)| \leq d_G(u) \leq d_G(v) \leq \sqrt{m}\) and \(\sum_{v \in V} \sum_{u \in V} \delta^+(u, v) \cdot |N_G(u)| \leq 2m^{1.5}\). If \(d_G(u) > \sqrt{m}\), \(|N_G(u)| \leq \sqrt{m}\) as well for \(d_G(u) \cdot |N_G(u)| \leq 2m^{1.5}\), \(|N_G(u)| \leq \sqrt{m}\) and \(|N_G(u)| \leq \sqrt{m}\). Similarly, we can prove that line 3-4 can be bounded by \(O(m^{1.5})\). And similar to line 6-25, the time complexity of Algorithm 5 is \(O(m^{1.5})\). □

Example 6.9. Let \(k = 2\), Figure 4 shows the result of edge reduction on \(G\) in Figure 2. As shown in Figure 4, \((v_9, v_{10}, v_{11})\) cannot be pruned by vertex reduction directly, as they are contained in a \((1, 2)\)-signed core. However, with edge reduction, as \(\delta_{G}^+(v_9, v_8) < 2\), \((v_8, v_9)\) is removed. The, \((v_8, v_9)\) and \((v_7, v_8)\) are removed as \(\delta_{G}^+(v_7, v_9) = 0\) and \(\delta_{G}^-(v_7, v_8) = 0\). As \(v_9\) has no edges afterwards, it is pruned. \(v_{10}\) and \(v_{11}\) are pruned similarly.
7 PERFORMANCE STUDIES

In this section, we present our experimental results. All the experiments are performed on a machine with two Intel Xeon 2.2GHz CPUs and 64GB RAM running CentOS 7.

**Algorithms.** We compare three algorithms: Baseline, MBCEnum and MBCEnum*. Baseline is the baseline solution shown in Section 4. MBCEnum is our algorithm shown in Section 5. MBCEnum* is the algorithm with the in-enumeration optimization shown in Section 6.1. Note that the pre-enumeration optimization strategies can be also used in Baseline and MBCEnum, thus, we apply them for all three algorithms for fairness.

All algorithms are implemented in C++, using g++ compiler with -O3. The time cost is measured as the amount of wall-clock time elapsed during the program’s execution. If an algorithm cannot finish in 12 hours, we denote the processing time as INF. We evaluate our algorithms on real and synthetic signed networks.

**Real datasets.** We first evaluate our algorithms on five real datasets. Slashdot and Epinions are signed networks in real world. AdjWordNet, DBLP and Douban are signed networks used in [8], [27] and [50], respectively. Slashdot and Epinions are downloaded from SNAP (http://snap.stanford.edu). Douban is from authors in [50]. AdjWordNet is downloaded from WordNet (https://wordnet.princeton.edu/). DBLP is downloaded from KONECT (http://konect.uni-koblenz.de/) and processed as the same shown in [27]. The details of each dataset are shown in Table 1.

**Exp-1: Efficiency when varying k.** In this experiment, we evaluate the efficiency of three algorithms when varying k from 4 to 10 and the results are shown in Figure 6.

As shown in Figure 6, Baseline consumes the most time among three algorithms on all datasets when we vary k and it can only handle the small datasets. For example, on Slashdot (Figure 6 (a)), MBCEnum and MBCEnum* are at least two orders of magnitude faster than Baseline. On Douban (Figure 6 (d)), Baseline cannot finish the enumeration in 12 hours. This is because Baseline does not consider the uniqueness of the signed networks and lots of unnecessary computations are involved in the enumeration of Baseline. MBCEnum is faster than Baseline on most of the test cases as MBCEnum takes the uniqueness of the signed networks into consideration and enumerates the maximal balanced cliques based on the signed network directly. MBCEnum* is the most efficient algorithm on all datasets when varying k due to the utilization of in-enumeration optimization strategies, which reveals the effectiveness of in-enumeration optimization strategies. Another phenomenon shown in Figure 6 is that the running time of all algorithms decreases as k increases. This is because as k increases, the pruning power of the optimization strategies proposed in Section 6 strengthens.

**Exp-2: Evaluation of the pre-enumeration optimization.** In this experiment, we evaluate the effectiveness and efficiency of the pre-enumeration optimization strategies proposed in Section 6.2. We report the number of pruned edges for VertexReduction and the sum of pruned edges for VertexReduction and EdgeReduction when varying k in Figure 7. Figure 8 shows the running time.

---

**Table 1: Statistic for the real datasets**

| Dataset  | n   | m   | $|E^+|$ | $|E^-|$ |
|----------|-----|-----|-------|-------|
| AdjWordNet | 21,247 | 426,896 | 378,993 | 47,903 |
| Slashdot | 77,357 | 516,575 | 396,378 | 120,197 |
| Epinions | 131,828 | 841,372 | 717,667 | 123,705 |
| DBLP    | 1,314,050 | 5,179,945 | 1,471,905 | 3,508,042 |
| Douban  | 1,588,565 | 13,918,375 | 9,034,537 | 4,883,838 |

---

![Figure 6: Running time of different algorithms varying k](image1)

![Figure 7: Pruned edges by pre-enumeration optimizations](image2)

![Figure 8: Running time of pre-enumeration optimizations](image3)
As shown in Figure 7, for VertexReduction, as \( k \) increases, the number of pruned edges increases as well. This is because as \( k \) increases, more vertices are not contained in the corresponding \((k - 1, k)\)-signed core. These vertices together with their incident edges are pruned. Figure 7 also reveals that EdgeReduction prunes much more edges than VertexReduction. This is because EdgeReduction adopts a more strict pruning condition. Figure 8 shows that as \( k \) increases, the running time of VertexReduction increases. Since as \( k \) increases, more vertices are explored by VertexReduction. On the other hand, the running time of them together decreases. This is because as \( k \) increases, more vertices are pruned by VertexReduction. As a result, EdgeReduction takes less time to conduct the pruning. Meanwhile, EdgeReduction is more time-consuming compared with VertexReduction. Thus, the running time together decreases.

**Exp-3: Scalability testing.** In this experiment, we test the scalability of MBCEnum and MBCEnum* on two large datasets DBLP and Douban by varying their vertices from 20% to 100%. Figure 9 shows the results.

As shown in Figure 9, when \( n \) increases, the running time of both algorithms increases as well, but MBCEnum* outperforms MBCEnum for all cases on both datasets. For example, on DBLP, when we sample 20% vertices, the running time of MBCEnum and MBCEnum* is 0.6 seconds and 0.5 seconds, respectively, while when sampling 80% vertices, their running times are 770.6 seconds and 4.0 seconds, respectively. It shows that MBCEnum* has a good scalability in practice.

**Exp-4: Case study on AdjWordNet.** In this experiment, we perform a case study on the real dataset AdjWordNet. In this dataset, two synonyms have a positive edge and two antonyms have a negative edge, and Table 2 shows some results obtained by our algorithm. As shown in Table 2, words in \( C_L \) or \( C_R \) have similar meaning while each word from \( C_L \) is an antonym to all words in \( C_R \). This case study verifies that maximal balanced clique enumeration can be applied in the applications to find synonym and antonym groups on dictionary data.

**Exp-5: Efficiency on synthetic datasets.** In this experiment, we evaluate our algorithms on synthetic datasets. We use the synthetic signed network generator, SRN, to generate the synthetic datasets with default settings [45, 56]. We generate four synthetic signed networks SN1-4 (details in Table 3) in different sizes and evaluate the efficiency of MBCEnum* and MBCEnum on SN1-4 similarly as Exp-1. The results are shown in Figure 10.

As shown in Figure 10, the trends on the synthetic datasets are similar to that on the real datasets. MBCEnum* outperforms MBCEnum when we vary \( k \), especially when \( k \) is small.

### Table 2: Case study on AdjWordNet

| \( C_L \) | \( C_R \) |
| --- | --- |
| raw, rough, rude | refined, smooth, suave |
| relaxing, reposeful, restful | restless, uneasy, ungratified, unsatisfied |
| interior, internal, intimate | away, foreign, outer, outside, remote |
| assumed, false, fictitious, fictive, mistaken, off-key, pretended, put-on, sham, sour, untrue | actual, existing, existing, factual, genuine, literal, real, tangible, touchable, true, truthful, unfeigned, veridical |
| active, animated, combat-ready, dynamic, dynamical, fighting, participating, alive, live |odynamic, asthenic, debilitated, enervated, undynamic, stagnant, light |
| following, undermentioned, next | ahead, in-the-lead, leading, preeminent, prima, star, starring, stellar |
| undesirable, unsuitable, unwanted | cherished, treasured, wanted, precious |

### Table 3: Statistic for the synthetic datasets

| Dataset | \( n \) | \( m \) | \( |E^+| \) | \( |E^-| \) |
| --- | --- | --- | --- | --- |
| SN1 | 500,000 | 17,535,536 | 10,192,418 | 7,343,118 |
| SN2 | 1,000,000 | 35,054,718 | 20,373,721 | 14,680,997 |
| SN3 | 2,000,000 | 70,156,704 | 40,776,222 | 29,380,482 |
| SN4 | 3,000,000 | 105,218,600 | 61,159,457 | 44,059,143 |

### Figure 10: Running time of different algorithms varying \( k \)

### 8 CONCLUSIONS

In this paper, we study the maximal balanced clique enumeration problem in signed networks. We propose a new enumeration algorithm tailored for signed networks. Based on the new enumeration algorithm, we explore two optimization strategies to further improve the efficiency of the enumeration algorithm. The experimental results on real and synthetic datasets demonstrate the efficiency, effectiveness and scalability of our solution.

**Acknowledgment.** Long Yuan is supported by NSFC61902184 and NSF of Jiangsu Province BK20190453. Xuemin Lin is supported by 2018YFB1003504, NSFC61232006, ARC DP180103096 and DP170101628. Lu Qin is supported by ARC DP160101513.
