

# Maximal Structural Balanced $k$ -plex Enumeration in Signed Graphs

Kai Yao

College of Computer Science  
Sichuan Normal University  
Chengdu, China  
kaiyao@sicnu.edu.cn

Xinchen Zhang

School of Computer Science  
The University of Sydney  
Sydney, Australia  
xzha9245@uni.sydney.edu.au

Lijun Chang

School of Computer Science  
The University of Sydney  
Sydney, Australia  
lijun.chang@sydney.edu.au

**Abstract**—Signed graphs, which represent interactions as friendly (positive edges) or antagonistic (negative edges), offer a powerful model to capture interesting structural properties of real-world phenomena. Recently, the structural balanced clique model has been proposed to identify polarized structures in signed graphs. A signed graph is a structural balanced clique if every pair of its vertices is connected by an edge and its vertices can be partitioned into two sets such that all edges inside each partition are positive and all cross-partition edges are negative. However, this model’s rigidity restricts its applicability in practice. In this paper, we formulate the structural balanced  $k$ -plex, which is a relaxation of structural balanced clique by allowing each vertex to miss a few connections or slightly violate structural balanced theory. We prove that the problem of enumerating maximal balanced  $k$ -plexes is #P-hard. To solve this problem, we first introduce the backtracking algorithm MBPE-BK and propose optimization techniques to enhance its practical performance. However, MBPE-BK’s efficiency is hindered by overlapping candidate sets. To overcome this challenge, we propose a novel algorithm MBPE with better time complexity than MBPE-BK (i.e.,  $\mathcal{O}^*(2^\delta)$  v.s.  $\mathcal{O}^*(3^{\delta D})$ ). In addition, we adopt the minimum-degree branching strategy to speed up the critical enumeration procedure, which improves the worst-case time complexity of MBPE to  $\mathcal{O}^*(\alpha_k^\delta)$ , where  $1 < \alpha_k < 2$  is a constant that depends only on  $k$ . Extensive experiments on real-world and synthetic datasets are conducted to demonstrate the efficiency of our algorithms and the effectiveness of our model.

## I. INTRODUCTION

Signed graphs provide an enriched representation of conventional graphs by incorporating the concept of *polarity* to signify relationships between entities/vertices using *positive* and *negative* edge signs [1]. This enhancement enables the capture of various types of relationships in different domains. For example, in social networks, signed graphs model friend-foe relationships [2]. In opinion networks, they represent support-dissent opinions [3]. In protein-protein interaction networks, they model activation-inhibition interactions [4].

In the literature, many traditional graph analysis tasks have been extended to signed graphs by treating positive and negative edges differently, such as community detection [5], [6], link prediction [7], [8] and recommendation systems [9], [10]. Among them, the problem of structural balanced clique computation receives an increasing attention recently [11]–[15]. A graph is a structural balanced clique if (1) it is a clique (i.e., every pair of its vertices is connected by an edge), and

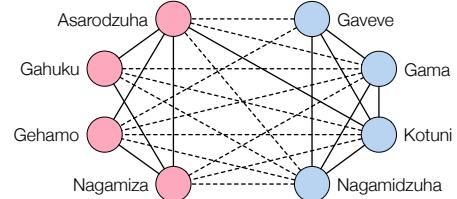


Fig. 1. Polarized groups in Tribes

(2) it obeys structural balanced theory (i.e., its vertices can be partitioned into two sets such that all edges between vertices in the same set are positive and all edges between vertices from different sets are negative). Identifying balanced cliques holds significant relevance in various practical domains, such as polarized community discovery in social networks [6], [16], [17], synonym-antonym groups detection in word networks [5], and protein complexes identification in protein-protein interaction networks [18].

However, the structural balanced clique concept is often too restrictive for practical applications. Firstly, not every user is connected to all other users in a polarized structure. Secondly, not every user strictly agrees with all members of the same group or disagrees with all members of the opposite group. To illustrate, Figure 1 shows a real-world polarized group (i.e., red vertices and blue vertices) in Tribes [19], which depicts the friendship (solid lines) and enmity (dashed lines) among 16 tribes situated in the Eastern Central Highlands. Although this is a clearly defined polarized structure, not every user is connected to all others (e.g., “Gahuku” and “Gehamo”), and not every user obeys structural balanced theory (e.g., “Asarodzuha” and “Kotuni” are friends).

In this paper, we formulate the structural balanced  $k$ -plex, which is a relaxation of structural balanced clique by allowing each vertex to miss a few connections or slightly violate structural balanced theory. Specifically, given a signed graph  $G$ , two disjoint vertex subsets  $P_L$  and  $P_R$  induce a structural balanced  $k$ -plex if for each vertex  $u \in P_L \cup P_R$ , there are no more than  $k$  vertices that: (1) are non-neighbors of  $u$ , (2) are in the same set as  $u$  and are negative neighbors of  $u$ , or (3) are in the opposite set as  $u$  and are positive neighbors of  $u$ . Equivalently, for each vertex  $u \in P_L \cup P_R$ , the sum of  $u$ ’s same-set positive neighbors and opposite-set negative

neighbors should be at least  $|P_L \cup P_R| - k$ . This definition ensures that each member holds a clear standpoint within the polarized structure while allowing personal relationships to a limited extent through the parameter  $k$ . Referring back to Figure 1, the sets of red and blue vertices induce a structural balanced 3-plex, as each vertex has at least 5 same-set positive neighbors and opposite-set negative neighbors. For simplicity, we refer to structural balanced as balanced and denote a balanced  $k$ -plex as  $(P_L, P_R)$  in the following.

We prove that the problem of enumerating maximal balanced  $k$ -plexes is #P-hard. To solve this problem, we first propose a baseline algorithm MBPE-BK, by drawing inspiration from the well-known Bron-Kerbosch algorithm [20], which is designed for maximal clique enumeration in unsigned graphs. The main idea of MBPE-BK is that, for each vertex  $v_i$  in the graph, we iteratively build up a partial balanced  $k$ -plex  $(P_L, P_R)$ , where  $P_L = \{v_i\}$  and  $P_R = \emptyset$  initially. In addition, we maintain two candidate sets  $C_L$  and  $C_R$  of vertices that are used for growing  $P_L$  and  $P_R$ , respectively. Then, we iteratively try each vertex of  $C_L$  to be added to  $P_L$  and each vertex of  $C_R$  to be added to  $P_R$ , to grow the solution and conduct the recursion until maximal balanced  $k$ -plexes are detected.

However, the performance of MBPE-BK is unsatisfactory due to the heavy overlap between  $C_L$  and  $C_R$ . To overcome this issue, we propose the algorithm MBPE that adopts a different strategy at the root level of the backtracking search tree. Specifically, for each vertex  $v_i$ , we exhaustively enumerate all possible initial balanced  $k$ -plex  $(P_L, P_R)$  such that  $v_i \in P_L$  and  $(P_L, P_R)$  contains at most  $k$  vertices that: (1) are non-neighbors of  $v_i$ ; (2) are negative neighbors of  $v_i$  within  $P_L$ ; (3) are positive neighbors of  $v_i$  within  $P_R$ . For each initial balanced  $k$ -plex, we set  $C_L$  and  $C_R$  as  $v_i$ 's remaining positive neighbors and negative neighbors, respectively. In this way,  $C_L$  and  $C_R$  become disjoint automatically. By utilizing this fact, we propose subgraph sparsification and partition-based vertex reduction techniques to further improve the efficiency of MBPE. We prove that MBPE achieves a better theoretical time complexity than MBPE-BK (i.e.,  $\mathcal{O}^*(2^\delta)$  v.s.  $\mathcal{O}^*(3^{\delta D})$ ).<sup>1</sup> Additionally, we speed up the enumeration procedure by adopting the minimum-degree pivoting strategy [21], where the minimum-degree vertex of  $P_L \cup P_R$  is chosen as the branching vertex throughout the enumeration process. This strategy improves the worst-case time complexity of MBPE to  $\mathcal{O}^*(\alpha_k^\delta)$ , where  $1 < \alpha_k < 2$  is a constant depending only on  $k$ , e.g.,  $\alpha_2 = 1.839$ ,  $\alpha_3 = 1.928$ , and  $\alpha_4 = 1.966$ .

Our main contributions are summarized as follows.

- We formalize the structural balanced  $k$ -plex model in signed graphs, and prove that enumerating maximal structural balanced  $k$ -plexes is #P-hard.
- We propose a basic backtracking algorithm MBPE-BK as well as several optimization techniques to improve its practical performance.

<sup>1</sup>For presentation simplicity, we also use the  $\mathcal{O}^*$  notation, in addition to the usual  $\mathcal{O}$  notation, for time complexity analysis.  $\mathcal{O}^*$  notation focuses on the exponential part of the time complexity by hiding polynomial factors.

- We propose a novel algorithm MBPE with better time complexity than MBPE-BK (i.e.,  $\mathcal{O}^*(2^\delta)$  v.s.  $\mathcal{O}^*(3^{\delta D})$ ). We also propose subgraph sparsification and partition-based vertex reduction to improve the efficiency of MBPE.
- We adopt the minimum-degree branching strategy to accelerate the enumeration procedure, improving the worst-case time complexity of MBPE to  $\mathcal{O}^*(\alpha_k^\delta)$ , where  $1 < \alpha_k < 2$  is a constant that depends only on  $k$ , e.g.,  $\alpha_2 = 1.839$ ,  $\alpha_3 = 1.928$ , and  $\alpha_4 = 1.966$ .

We conduct extensive experiments on both real-world and synthetic signed graphs. The results demonstrate that our advanced algorithm not only has better theoretical time complexity but also runs efficiently in practice. Furthermore, case studies on real-world datasets show that our model can discover more interesting results compared to existing models.

**Organization.** The rest of the paper is organized as follows. Section II provides preliminaries and defines the problem studied in this paper. Section III presents our baseline algorithm and its optimizations. Section IV proposes the advanced algorithm and a novel branching strategy. Experimental results are reported in Section V. Related works are discussed in Section VI. Finally, Section VII concludes the paper. All the proofs for lemmas and theorems can be found in Appendix A.

## II. PRELIMINARIES

In the paper, we focus on undirected signed graphs  $G = (V, E^+, E^-)$ , where  $V$  is the set of vertices, and  $E^+$  and  $E^-$  are the sets of positive and negative edges, respectively. The numbers of vertices and edges in  $G$  are denoted by  $n = |V|$  and  $m = |E^+| + |E^-|$ . For each vertex  $v \in V$ ,  $N_G^+(v)$  denotes the positive neighbors of  $v$ , i.e.,  $N_G^+(v) = \{u \in V \mid (v, u) \in E^+\}$ , and  $N_G^-(v)$  denotes the negative neighbors of  $v$ , i.e.,  $N_G^-(v) = \{u \in V \mid (v, u) \in E^-\}$ ,  $N_G(v) = N_G^-(v) \cup N_G^+(v)$  denotes neighbors of  $v$ . Additionally, we use  $d_G^+(v) = |N_G^+(v)|$ ,  $d_G^-(v) = |N_G^-(v)|$ , and  $d_G(v) = |N_G(v)|$  to respectively denote the positive degree, negative degree, and degree of vertex  $v$ . Given a vertex subset  $S \subseteq V$ , we use  $G[S]$  to denote the induced subgraph of  $G$  that consists of all edges between vertices of  $S$ , i.e.,  $G[S] = (S, \{(u, v) \in E^+ \cup E^- \mid u, v \in S\})$ . We use  $N_G^2(v)$  to denote 2-hop neighbors of  $v$ , i.e., the set of vertices with distance exactly 2 to  $v$  in  $G$ . When the context is clear, we omit the subscript  $G$ .

**Definition 1** (Structural Balanced Graph [22]). A signed graph  $G = (V, E^+, E^-)$  is *structural balanced* if its vertices can be partitioned into two sets  $V_L$  and  $V_R$  such that all edges between vertices in the same set are positive and all edges between vertices from different sets are negative.

**Definition 2** ( $k$ -Plex [23]). A signed graph  $G = (V, E^+, E^-)$  is a  *$k$ -plex* if for each  $u \in V$ , its degree satisfies  $d_G(u) \geq |V| - k$ , i.e.,  $u$  misses connections to at most  $k$  vertices (including  $u$  itself).

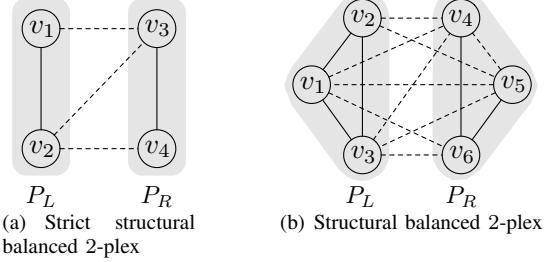


Fig. 2. Illustration of (strict) structural balanced  $k$ -plex

**Definition 3** (Strict Structural Balanced  $k$ -Plex). Given a signed graph  $G = (V, E^+, E^-)$ , two disjoint vertex subsets  $P_L$  and  $P_R$  induce a *strict structural balanced  $k$ -plex* if the subgraph of  $G$  induced by  $P_L \cup P_R$  is structural balanced and is a  $k$ -plex.

For the signed graph in Figure 2(a),  $P_L = \{v_1, v_2\}$  and  $P_R = \{v_3, v_4\}$  form a strict structural balanced 2-plex since the subgraph induced by  $P_L \cup P_R$ , i.e.,  $G[P_L \cup P_R]$ , is structural balanced and is a 2-plex. However, this model adheres strictly to the structural balanced theory. Next, we propose a new model that further relaxes this constraint, making it more suitable for practical applications.

Given two disjoint vertex subsets  $P_L$  and  $P_R$  in a signed graph  $G$ , we define the *maximal structural balanced subgraph of  $G$  induced by  $(P_L, P_R)$*  as the subgraph of  $G$  whose vertices are  $P_L \cup P_R$  and whose edges consist of (1) all positive edges of  $G$  between vertices of  $P_L$ , (2) all positive edges between vertices of  $P_R$ , and (3) all negative edges between vertices of  $P_L$  and vertices of  $P_R$  (i.e., removing edges that violate structural balanced theory). For convenience we denote the maximal structural balanced subgraph induced by  $(P_L, P_R)$  as  $g(P_L, P_R)$ . It is easy to see that any  $g(P_L, P_R)$  is structural balanced.

**Definition 4** (Structural Balanced  $k$ -Plex). Given a signed graph  $G = (V, E^+, E^-)$ , two disjoint vertex subsets  $P_L$  and  $P_R$  induce a *structural balanced  $k$ -plex* if the maximal structural balanced subgraph of  $G$  induced by  $P_L$  and  $P_R$ , i.e.,  $g(P_L, P_R)$ , is a  $k$ -plex.

For example, consider the signed graph  $G$  in Figure 2(b). The maximal structural balanced subgraph of  $G$  induced by  $P_L = \{v_1, v_2, v_3\}$  and  $P_R = \{v_4, v_5, v_6\}$  contains all edges except  $(v_4, v_5)$ ; it is easy to see that this subgraph is a 2-plex. Therefore,  $(P_L = \{v_1, v_2, v_3\}, P_R = \{v_4, v_5, v_6\})$  can be identified as a structural balanced 2-plex. The concept of a structural balanced  $k$ -plex (Definition 4) serves as a generalization of the strict structural balanced  $k$ -plex (Definition 3), by allowing each vertex to have a limited number of negative links to vertices in the same set and positive links to vertices in the opposite set. We evaluate the performance of these two models in the experimental part.

For simplicity, we refer to  $(P_L, P_R)$  as a *balanced  $k$ -plex* if  $P_L$  and  $P_R$  induce a structural balanced  $k$ -plex. For a balanced  $k$ -plex, the role of  $P_L$  and  $P_R$  can be swapped, i.e.,  $(P_L, P_R)$  is same as  $(P_R, P_L)$ . We measure the size of balanced  $k$ -

plex by its number of vertices, i.e.,  $|P_L \cup P_R|$ . The property of balanced  $k$ -plex is *hereditary*; that is, given a balanced  $k$ -plex  $(P_L, P_R)$ ,  $(P'_L, P'_R)$  is also a balanced  $k$ -plex for any  $P'_L \subseteq P_L$  and  $P'_R \subseteq P_R$ . A balanced  $k$ -plex  $(P_L, P_R)$  is *maximal* if adding any vertex to either  $P_L$  or  $P_R$  would violate the balanced  $k$ -plex property.

In this paper, we aim to enumerate maximal balanced  $k$ -plexes  $(P_L, P_R)$  in a given signed graph. Moreover, to avoid generating trivial solutions, we require the numbers of vertices in  $P_L$  and in  $P_R$  to be no less than a user-given threshold  $\tau$ , i.e.,  $|P_L| \geq \tau$  and  $|P_R| \geq \tau$ . We also require  $\tau \geq k$  such that each vertex  $v$  in the balanced  $k$ -plex possesses a sufficient number of neighbors (i.e., positive neighbors in the same set with  $v$  and negative neighbors in the opposite set with  $v$ ). By satisfying this condition, we ensure that every member of the balanced  $k$ -plex holds a clear standpoint, resulting in a well-defined and polarized structure for the entire solution. The studied problem is formulated as follows.

**Problem 1** (Maximal Balanced  $k$ -Plex Enumeration). Given a signed graph  $G$  and two positive integers  $k$  and  $\tau$  such that  $\tau \geq k$ , the maximal balanced  $k$ -plex enumeration problem aims to identify all maximal balanced  $k$ -plexes  $(P_L, P_R)$  in  $G$  s.t.  $|P_L| \geq \tau$  and  $|P_R| \geq \tau$ .

**Theorem 1.** *The maximal balanced  $k$ -plex enumeration problem is #P-hard.*

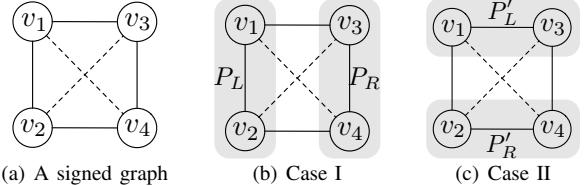


Fig. 3. Uniqueness of the balanced  $k$ -plex

**Uniqueness of the balanced  $k$ -plex.** It is noteworthy that two distinct balanced  $k$ -plexes, denoted as  $(P_L, P_R)$  and  $(P'_L, P'_R)$ , can share the same set of vertices, meaning  $P_L \cup P_R = P'_L \cup P'_R$ . For instance, in the signed graph illustrated in Figure 3(a), both  $(P_L = \{v_1, v_2\}, P_R = \{v_3, v_4\})$  and  $(P'_L = \{v_1, v_3\}, P'_R = \{v_2, v_4\})$  represent balanced 2-plexes, yet they consist of the same vertices. In this study, we regard them as different balanced  $k$ -plexes since different partitions imply different perspectives of the vertices. To gain a deeper understanding of balanced  $k$ -plexes, the following lemma demonstrates that when two distinct balanced  $k$ -plexes have the same set of vertices, certain constraints are imposed.

**Lemma 1.** *Given two different balanced  $k$ -plexes  $(P_L, P_R)$  and  $(P'_L, P'_R)$ , if  $P_L \cup P_R = P'_L \cup P'_R$ , we must have  $|P_L \cup P_R| \leq 4k - 4$ .*

### III. A BRON-KERBOSCH-BASED ALGORITHM

The classic Bron-Kerbosch algorithm is proposed in [20] for enumerating maximal cliques in unsigned graphs. In this section, we adopt the Bron-Kerbosch framework to solve the

maximal balanced  $k$ -plex enumeration problem. The main idea is to iteratively build up two vertex sets  $P_L$  and  $P_R$  such that  $(P_L, P_R)$  is a balanced  $k$ -plex. In addition, we maintain two candidate sets  $C_L$  and  $C_R$  of vertices that are used for growing  $P_L$  and  $P_R$ , respectively. Then, we iteratively try each vertex of  $C_L$  to be added to  $P_L$  and each vertex of  $C_R$  to be added to  $P_R$ , to grow the partial balanced  $k$ -plex and conduct the recursion. We also maintain exclusive sets  $X_L$  and  $X_R$  to record the vertices that have been processed to avoid generating duplicate results. Before introducing the details of our algorithm, we analyze two properties of balanced  $k$ -plex that are important for our algorithm design.

**Lemma 2.** *Given a signed graph  $G$ , if the size of a balanced  $k$ -plex  $(P_L, P_R)$  is at least  $2k-1$ , the diameter of the subgraph of  $G$  induced by  $P_L \cup P_R$  is bounded by 2.*

We categorize the 2-hop neighbors of  $v$  into four distinct types: positive-positive, positive-negative, negative-positive, and negative-negative neighbors. These are denoted as  $N_G^{++}(v)$ ,  $N_G^{+-}(v)$ ,  $N_G^{-+}(v)$  and  $N_G^{--}(v)$ , respectively. Note that each type of 2-hop neighbor set may overlap with the others.

**Lemma 3.** *Suppose  $v$  is an arbitrary vertex in a signed graph  $G$ . For any balanced  $k$ -plex  $(P_L, P_R)$  with size at least  $2k-1$  and  $v \in P_L$ , we have that  $P_L \subseteq \{v\} \cup N_G(v) \cup N_G^{++}(v) \cup N_G^{--}(v)$  and  $P_R \subseteq N_G(v) \cup N_G^{+-}(v) \cup N_G^{-+}(v)$ .*

---

### Algorithm 1: MBPE-BK

---

```

Input: A signed graph  $G = (V, E^+, E^-)$  and integers  $k, \tau$ 
Output: Maximal balanced  $k$ -plexes
1 Sort  $V$  by degeneracy order as  $\{v_1, \dots, v_n\}$ ;
2 for each  $v_i$  in degeneracy order do
3    $P_L \leftarrow \{v_i\}$ ,  $P_R \leftarrow \emptyset$ ;
4    $C_L \leftarrow \{v_{i+1}, \dots, v_n\} \cap (N_G(v_i) \cup N_G^{++}(v_i) \cup N_G^{--}(v_i))$ ;
5    $C_R \leftarrow \{v_{i+1}, \dots, v_n\} \cap (N_G(v_i) \cup N_G^{+-}(v_i) \cup N_G^{-+}(v_i))$ ;
6    $X_L \leftarrow \{v_1, \dots, v_{i-1}\} \cap (N_G(v_i) \cup N_G^{++}(v_i) \cup N_G^{--}(v_i))$ ;
7    $X_R \leftarrow \{v_1, \dots, v_{i-1}\} \cap (N_G(v_i) \cup N_G^{+-}(v_i) \cup N_G^{-+}(v_i))$ ;
8   Enum( $P_L, P_R, C_L, C_R, X_L, X_R$ );
9 Procedure Enum( $P_L, P_R, C_L, C_R, X_L, X_R$ )
10   $C_L \leftarrow \{v \in C_L : (P_L \cup \{v\}, P_R)$  is a balanced  $k$ -plex};
11   $C_R \leftarrow \{v \in C_R : (P_L, P_R \cup \{v\})$  is a balanced  $k$ -plex};
12   $X_L \leftarrow \{v \in X_L : (P_L \cup \{v\}, P_R)$  is a balanced  $k$ -plex};
13   $X_R \leftarrow \{v \in X_R : (P_L, P_R \cup \{v\})$  is a balanced  $k$ -plex};
14  if  $C_L \cup C_R = \emptyset$  and  $X_L \cup X_R = \emptyset$  then
15    if  $|P_L| \geq \tau$  and  $|P_R| \geq \tau$  then emit  $(P_L, P_R)$ ;
16    for each  $v \in C_L$  do
17       $C_L \leftarrow C_L \setminus \{v\}$ ;
18      Enum( $P_R, P_L \cup \{v\}, C_R \setminus \{v\}, C_L, X_R \setminus \{v\}, X_L$ );
19       $X_L \leftarrow X_L \cup \{v\}$ ;
20    for each  $v \in C_R$  do
21       $C_R \leftarrow C_R \setminus \{v\}$ ;
22      Enum( $P_R \cup \{v\}, P_L, C_R, C_L \setminus \{v\}, X_R, X_L \setminus \{v\}$ );
23       $X_R \leftarrow X_R \cup \{v\}$ ;

```

---

The pseudo-code of our algorithm is presented in Algorithm 1, denoted MBPE-BK. The major framework is firstly sort vertices in degeneracy order  $\eta = \{v_1, \dots, v_n\}$  (Line 1), and then iteratively search for all the maximal  $v_i$ -lead balanced

$k$ -plex, i.e., those balanced  $k$ -plexes in which  $v_i$  precedes all other vertices according to  $\eta$  (Lines 2-8). The degeneracy order  $\eta = \{v_1, \dots, v_n\}$  is defined such that for each  $1 \leq i \leq n$ ,  $v_i$  is the minimum-degree vertex in  $G[\{v_1, \dots, v_n\}]$  [24]. This ordering minimizes the maximum number of neighbors that precede  $v_i$  for each  $v_i \in V$  in  $G[\{v_1, \dots, v_n\}]$ , potentially reducing the size of initial candidate sets. It uses  $P_L$  and  $P_R$  to maintain the balanced  $k$ -plex, which are initialized with  $v_i$  and  $\emptyset$ . It also uses  $C_L$  and  $C_R$  to maintain the set of candidates that are used to grow the partial balanced  $k$ -plex. Since we require  $\tau \geq k$ , the target balanced  $k$ -plex must have size of at least  $2k-1$ . Thus Lemma 3 can be employed to optimize the initialization of  $C_L$  and  $C_R$  (Lines 4-5). Note that here we only need to consider the vertices that succeed  $v_i$  in the ordering  $\eta$ . To avoid outputting duplicate results,  $X_L$  and  $X_R$  are used to record the vertices that have been processed (Lines 6-7). After initializing these six sets, it invokes the enumeration procedure **Enum** to find maximal  $v_i$ -lead balanced  $k$ -plexes (Line 8).

Procedure **Enum** performs the maximal balanced  $k$ -plex enumeration based on the given six sets. Firstly,  $C_L$ ,  $C_R$ ,  $X_L$  and  $X_R$  are refined by removing vertices that cannot form a larger balanced  $k$ -plex with the partial solution  $(P_L, P_R)$  (Lines 9-12). If these four sets are empty and  $(P_L, P_R)$  satisfies the size constraint,  $(P_L, P_R)$  is selected as the result (Line 14). Otherwise, it expands  $(P_L, P_R)$  by adding each vertex  $v \in C_L$  to  $P_L$  and recursively invokes itself to further enlarge the balanced  $k$ -plex (Line 17). Note that in the new recursion,  $v$ 's copies in  $C_R$  and  $X_R$  (if there are) should be removed. After  $v$  is processed, it is added to  $X_L$  (Line 18). Then, the algorithm continues to expand  $(P_L, P_R)$  by adding each vertex  $v \in C_R$  to  $P_R$  in a similar way (Lines 19-22). Note that at Lines 17 and 21, we swap the roles of  $P_L$  and  $P_R$ ,  $C_L$  and  $C_R$ ,  $X_L$  and  $X_R$ , such that we are adding vertices to the two sides of the growing balanced  $k$ -plex in alternating order; this is to avoid generating too skewed intermediate results.

**Theorem 2.** *The time complexity of MBPE-BK (Algorithm 1) is  $\mathcal{O}(n(\delta D)^2 3^{\delta D})$ , where  $\delta$  and  $D$  are the degeneracy and maximum degree of  $G$ , respectively.*

#### A. Optimizations on MBPE-BK

In this subsection, we present optimizations to improve the efficiency of MBPE-BK.

**Vertex reduction.** By utilizing the size constraint  $\tau$ , we can remove unpromising vertices that are not contained in any maximal balanced  $k$ -plex. We have the following lemma.

**Lemma 4.** *Given a signed graph  $G$ , and integers  $k$  and  $\tau$ , the vertex  $v$  in a maximal balanced  $k$ -plex must satisfy:*

- $d_G^+(v) \geq \tau - k$ ;
- $d_G^-(v) \geq \tau - k + 1$ ;
- $d_G(v) \geq 2\tau - k$ .

The pseudo-code of vertex reduction is shown in Algorithm 2. Firstly, it identifies the unqualified vertex  $v$  (Line 1). Since  $v$  will be removed from the graph, it decreases the positive (resp. negative) degree and degree of its positive (resp.

negative) neighbors by 1 (Lines 2-5). Lastly, the algorithm removes  $v$  and its associated edges from the graph (Line 6). The algorithm repeats the above steps until all the remaining vertices satisfy the degree constraints. The time complexity of VertexReduction is  $\mathcal{O}(n + m)$ .

---

**Algorithm 2:** VertexReduction

---

**Input:** A signed graph  $G = (V, E^+, E^-)$  and integers  $k, \tau$   
**Output:** A reduced signed graph  
1 while  $\exists v \in V$  s.t.  $d_G^+(v) < \tau - k$  or  $d_G^-(v) < \tau - k + 1$  or  
 $d_G(v) < 2\tau - k$  do  
2   for each  $u \in N_G^+(v)$  do  
3      $d_G^+(u) \leftarrow d_G^+(u) - 1$ ;  $d_G(u) \leftarrow d_G(u) - 1$ ;  
4   for each  $u \in N_G^-(v)$  do  
5      $d_G^-(u) \leftarrow d_G^-(u) - 1$ ;  $d_G(u) \leftarrow d_G(u) - 1$ ;  
6   Remove  $v$  from  $G$ ;

---

**Edge reduction.** We can also prune unpromising edges based on the common neighbors between two vertices. For a positive edge  $(u, v)$  in  $G$ , we define its positive-positive support as the number of vertices that positively link to both  $u$  and  $v$  (i.e.,  $\Delta_G^{++}(u, v) = |\{w \mid (u, w) \in E^+ \wedge (v, w) \in E^+\}|$ ), and define its negative-negative support as the number of vertices that negatively link to both  $u$  and  $v$  (i.e.,  $\Delta_G^{--}(u, v) = |\{w \mid (u, w) \in E^- \wedge (v, w) \in E^-\}|$ ). For a negative edge  $(u, v)$ , we define its positive-negative support as the number of vertices that positively link to  $u$  and negatively link to  $v$  (i.e.,  $\Delta_G^{+-}(u, v) = |\{w \mid (u, w) \in E^+ \wedge (v, w) \in E^-\}|$ ), and define its negative-positive support as the number of vertices that negatively link to  $u$  and positively link to  $v$  (i.e.,  $\Delta_G^{-+}(u, v) = |\{w \mid (u, w) \in E^- \wedge (v, w) \in E^+\}|$ ). We have the following lemma.

**Lemma 5.** Given a signed graph  $G$ , and integers  $k$  and  $\tau$ . Suppose  $(P_L, P_R)$  is a valid balanced  $k$ -plex, any positive edge  $(u, v)$  in  $g(P_L, P_R)$  must satisfy:

- $\Delta_G^{++}(u, v) \geq \tau - 2k$ ;
- $\Delta_G^{--}(u, v) \geq \tau - 2k + 2$ ;
- $\Delta_G^{++}(u, v) + \Delta_G^{--}(u, v) \geq 2\tau - 2k$ .

Similarly, any negative edge  $(u, v)$  in  $g(P_L, P_R)$  must satisfy:

- $\Delta_G^{+-}(u, v) \geq \tau - 2k + 1$ ;
- $\Delta_G^{-+}(u, v) \geq \tau - 2k + 1$ ;
- $\Delta_G^{+-}(u, v) + \Delta_G^{-+}(u, v) \geq 2\tau - 2k$ .

The pseudo-code of edge reduction is shown in Algorithm 3. Firstly, it computes  $\Delta_G^{++}(u, v)$  and  $\Delta_G^{--}(u, v)$  for each positive edge, and  $\Delta_G^{+-}(u, v)$  and  $\Delta_G^{-+}(u, v)$  for each negative edge (Lines 1-2). Then the positive edge  $(u, v)$  will be removed if it violates Lemma 5 (Line 5). At the same time, the algorithm updates the common neighbor supports for those affected edges (Lines 6-11). The negative edge  $(u, v)$  will also be removed by Lemma 5 and the common neighbor supports for those affected edges are updated similarly (Lines 13-19). The algorithm terminates if no edge violates Lemma 5 (Line 3). The time complexity of EdgeReduction is  $\mathcal{O}(m^{1.5})$ .

In the implementation, we apply vertex and edge reductions on the original graph  $G$  at the beginning of MBPE-BK. Note

---

**Algorithm 3:** EdgeReduction

---

**Input:** A signed graph  $G = (V, E^+, E^-)$  and integers  $k, \tau$   
**Output:** A reduced signed graph  
1 for each  $(u, v) \in E^+$  do Obtain  $\Delta_G^{++}(u, v)$  and  $\Delta_G^{--}(u, v)$ ;  
2 for each  $(u, v) \in E^-$  do Obtain  $\Delta_G^{+-}(u, v)$  and  $\Delta_G^{-+}(u, v)$ ;  
3 while  $\exists(u, v) \in E^+ \cup E^-$  violates Lemma 5 do  
4   if  $(u, v) \in E^+$  then  
5     Remove  $(u, v)$  from  $G$ ;  
6     for each  $w$  s.t.  $(u, w) \in E^+$  and  $(v, w) \in E^+$  do  
7        $\Delta_G^{++}(u, w) \leftarrow \Delta_G^{++}(u, w) - 1$ ;  
8        $\Delta_G^{++}(v, w) \leftarrow \Delta_G^{++}(v, w) - 1$ ;  
9     for each  $w$  s.t.  $(u, w) \in E^-$  and  $(v, w) \in E^-$  do  
10        $\Delta_G^{--}(u, w) \leftarrow \Delta_G^{--}(u, w) - 1$ ;  
11        $\Delta_G^{--}(v, w) \leftarrow \Delta_G^{--}(v, w) - 1$ ;  
12   else  
13     Remove  $(u, v)$  from  $G$ ;  
14     for each  $w$  s.t.  $(u, w) \in E^+$  and  $(v, w) \in E^-$  do  
15        $\Delta_G^{--}(u, w) \leftarrow \Delta_G^{--}(u, w) - 1$ ;  
16        $\Delta_G^{++}(v, w) \leftarrow \Delta_G^{++}(v, w) - 1$ ;  
17     for each  $w$  s.t.  $(u, w) \in E^-$  and  $(v, w) \in E^+$  do  
18        $\Delta_G^{++}(u, w) \leftarrow \Delta_G^{++}(u, w) - 1$ ;  
19        $\Delta_G^{--}(v, w) \leftarrow \Delta_G^{--}(v, w) - 1$ ;

---

that vertex and edge reductions may interact. That is, removing unqualified vertices can cause some edges to become invalid, and removing unqualified edges can cause some vertices to become invalid. Therefore, we repeatedly apply vertex and edge reductions until no more vertices or edges can be pruned. Additionally, during the enumeration process, we perform reductions on the subgraph that the current search instance is focusing on. Specifically, at the beginning of Enum, we apply vertex reduction on the subgraph  $G[P_L \cup P_R \cup C_L \cup C_R]$ . Note that we do not apply edge reduction in Enum due to its high computational cost.

**Pivoting technique.** The Bron-Kerbosch algorithm designed for maximal clique enumeration uses an effective technique called pivoting to reduce the number of branches in the search tree. The main idea of pivoting is that each maximal clique must include either the vertex  $u$  (i.e., the pivot) or a non-neighbor of  $u$ . Otherwise, a clique only contains neighbors of  $u$  and thus we can enlarge it by adding  $u$  to it, which makes it non-maximal. Motivated by this, we present a generalized version of pivoting that is applicable to the problem of maximal balanced  $k$ -plex enumeration. For clarity, we use  $P$  to denote  $(P_L, P_R)$  and  $C$  to denote  $(C_L, C_R)$ . We define  $u \in C$  if  $u \in C_L \cup C_R$ . For a vertex  $u \in C$ ,  $P \cup \{u\}$  is defined as  $(P_L \cup \{u\}, P_R)$  if  $u \in C_L$  and as  $(P_L, P_R \cup \{u\})$  if  $u \in C_R$ .

**Lemma 6.** Let  $P = (P_L, P_R)$  be a partial balanced  $k$ -plex, and  $C = (C_L, C_R)$  be the corresponding candidate set. Given a vertex  $u \in C$ , any expanded balanced  $k$ -plex must contain either  $u$  or the vertex  $v$  that  $v \neq u$  and satisfy one of the following conditions:

- (1)  $v \in C$  and  $u, v$  are non-neighbors in  $g(P \cup \{u, v\})$ ;
- (2)  $v \in C$  and  $u, v$  are neighbors in  $g(P \cup \{u, v\})$  and  $u, v$  have common non-neighbors in  $g(P \cup \{u, v\})$ .

Lemma 6 suggests that if a candidate vertex  $v$  is a neighbor of candidate vertex  $u$  and does not share any common non-neighbor with  $u$  in  $g(P \cup \{u, v\})$ , then we do not need to branch on  $v$  without affecting the correctness of the algorithm. Suppose  $u$  is the pivot from  $C_L$ , we have the following branching rule. At Line 15, Algorithm 1 only generates branches for  $v \in C_L$  such that  $v \notin N^+(u)$  or  $P_L \setminus (N^+(u) \cup N^+(v)) \neq \emptyset$  or  $P_R \setminus (N^-(u) \cup N^-(v)) \neq \emptyset$ . At Line 19, Algorithm 1 only generates branches for  $v \in C_R$  such that  $v \notin N^-(u)$  or  $P_L \setminus (N^+(u) \cup N^-(v)) \neq \emptyset$  or  $P_R \setminus (N^-(u) \cup N^+(v)) \neq \emptyset$ . The branching rule can be derived similarly when the pivot  $u$  is from  $C_R$ . To reduce the number of recursive calls as many as possible, we pick the vertex  $u$  which cuts the most branches from the search tree as the pivot.

### B. The Drawback of MBPE-BK

Despite incorporating all the above optimizations, the performance of MBPE-BK remains unsatisfactory. The primary reason is that the candidate sets  $C_L$  and  $C_R$  often overlap significantly. This overlap arises from the fact that balanced  $k$ -plex model allows each member  $u$  to have a few non-neighbors or a few neighbors that violates structural balanced theory. Thus, at the early stage of the enumeration procedure,  $C_L$  and  $C_R$  may overlap heavily with each other. Such overlap can further weaken the effectiveness of vertex reduction during the enumeration process. To address this issue, we propose a new enumeration framework that ensures  $C_L$  and  $C_R$  are disjoint and contain only the neighbors of the lead vertex  $v_i$ . As a result, the candidate set will be of size  $\delta$ , where  $\delta$  represents the graph's degeneracy, typically a small value in real-world graphs.

## IV. THE ADVANCED ALGORITHM

In this section, we first propose a new search framework in Section IV-A. Then, we optimize the enumeration procedure in Section IV-B by proposing a minimum degree pivoting technique. We discuss the extension of our algorithms to the strict balanced  $k$ -plex computation in Section IV-C.

### A. A New Enumeration Framework

In MBPE-BK, the overlap between  $C_L$  and  $C_R$  arises from the fact that each vertex  $v_i$  in the balanced  $k$ -plex  $(P_L, P_R)$  is allowed to have at most  $k$  non-neighbors (including itself) in  $g(P_L, P_R)$ . In an ideal scenario, if we can determine in advance the set of  $v_i$ 's non-neighbors in  $g(P_L, P_R)$ , then we only need to focus on its neighbors, i.e.,  $C_L$  only contains  $v_i$ 's positive neighbors, and  $C_R$  only contains  $v_i$ 's negative neighbors. Thus  $C_L$  and  $C_R$  automatically become disjoint. To achieve this, for each vertex  $v_i$ , we enumerate all possible initial balanced  $k$ -plexes  $(P_L, P_R)$  such that  $v_i \in P_L$  and  $(P_L, P_R)$  contains at most  $k$  vertices that: (1) are non-neighbors of  $v_i$ ; (2) are negative neighbors of  $v_i$  within  $P_L$ ; (3) are positive neighbors of  $v_i$  within  $P_R$ . For each initial balanced  $k$ -plex, we set  $C_L$  and  $C_R$  as  $v_i$ 's remaining positive neighbors and negative neighbors, respectively, and identify all maximal  $v_i$ -lead balanced  $k$ -plexes under this search instance.

---

### Algorithm 4: MBPE

---

```

Input: A signed graph  $G = (V, E^+, E^-)$  and integers  $k, \tau$ 
Output: maximal balanced  $k$ -plexes
1 Sort  $V$  by degeneracy order:  $\eta = \{v_1, \dots, v_n\}$ ;
2 for each  $v_i$  in degeneracy order do
3   for each  $S \subseteq N_{\succ \eta}(v_i) \cup N_{\prec \eta}^2(v_i)$  s.t.  $|S| \leq k - 1$  do
4      $S_L \leftarrow \{u \in S : u \in N_{\prec \eta}^-(v_i)\};$ 
5      $S_R \leftarrow \{v \in S : v \in N_{\succ \eta}^+(v_i)\};$ 
6      $S_C \leftarrow \{v \in S : v \in N_{\prec \eta}^2(v_i)\};$ 
7     for each partition  $(T_L, T_R)$  of  $S_C$  do
8        $P_L \leftarrow \{v_i\} \cup S_L \cup T_L;$ 
9        $P_R \leftarrow S_R \cup T_R;$ 
10       $C_L \leftarrow N_{\succ \eta}^+(v_i) \setminus S_R;$ 
11       $C_R \leftarrow N_{\prec \eta}^-(v_i) \setminus S_L;$ 
12       $X_L \leftarrow N_{\prec \eta}(v_i) \cup N_{\prec \eta}^2(v_i) \cup (N_{\succ \eta}^-(v_i) \cup$ 
13       $N_{\prec \eta}^2(v_i)) \setminus (P_L \cup P_R);$ 
14       $X_R \leftarrow N_{\succ \eta}(v_i) \cup N_{\prec \eta}^2(v_i) \cup (N_{\succ \eta}^+(v_i) \cup$ 
15       $N_{\prec \eta}^2(v_i)) \setminus (P_L \cup P_R);$ 
16      Enum $(P_L, P_R, C_L, C_R, X_L, X_R);$ 

```

---

The pseudo-code of our algorithm MBPE is shown in Algorithm 4. It first sorts the vertices in the degeneracy order  $\eta = \{v_1, \dots, v_n\}$  (Line 1). Then, from  $v_1$  to  $v_n$ , the algorithm iteratively finds maximal  $v_i$ -lead balanced  $k$ -plexes (Lines 2-14). Let  $N_{\prec \eta}(v_i) = N(v_i) \cap \{v_1, \dots, v_{i-1}\}$  and  $N_{\succ \eta}(v_i) = N(v_i) \cap \{v_{i+1}, \dots, v_n\}$  be backward and forward neighbors of  $v_i$ , respectively;  $N_{\prec \eta}^2(v_i) = N^2(v_i) \cap \{v_1, \dots, v_{i-1}\}$ ,  $N_{\succ \eta}^2(v_i) = N^2(v_i) \cap \{v_{i+1}, \dots, v_n\}$  be the backward and forward 2-hop neighbors of  $v_i$ , respectively. For each vertex  $v_i \in V$ , it obtains the candidates that are at most 2-hop neighbors succeeding  $v_i$ , i.e.,  $N_{\succ \eta}(v_i) \cup N_{\prec \eta}^2(v_i)$ . Then, it exhaustively explores all possible vertex subsets  $S \subseteq N_{\succ \eta}(v_i) \cup N_{\prec \eta}^2(v_i)$  of size at most  $k - 1$ , which contain  $v_i$ 's potentially non-neighbors in  $g(P_L, P_R)$  (Line 3). Next, the vertices in  $S$  are allocated to sets  $S_L$ ,  $S_R$ , and  $S_C$  as follows:  $S_L$  (resp.  $S_R$ ) is the set of negative (resp. positive) neighbors of  $v_i$ , which will be placed in the same (resp. opposite) set with  $v_i$  to become  $v_i$ 's non-neighbors in  $g(P_L, P_R)$  (Lines 4-6).  $S_C$  contains  $v_i$ 's 2-hop neighbors and a further partition of  $S_C$  ensures all possibilities are exhausted (Line 7), since any vertex  $u \in S_C$  will be  $v_i$ 's non-neighbor in  $g(P_L, P_R)$  regardless of which set  $u$  is placed in. For each instance, the initial balanced  $k$ -plex  $(P_L, P_R)$  is established (Lines 8-9). Sets  $C_L$  and  $C_R$  are initialized to contain only the positive and negative neighbors of  $v_i$  by excluding vertices already in the initial balanced  $k$ -plex (Lines 10-11). The exclusive sets  $X_L$  and  $X_R$  include any other vertices that can be added to the initial balanced  $k$ -plex (Lines 12-13). After initializing these six sets, the algorithm invokes **Enum** to search for all maximal  $v_i$ -lead balanced  $k$ -plexes under this instance (Line 14).

**Subgraph sparsification.** In MBPE, it is guaranteed that  $C_L$  and  $C_R$  are disjoint (i.e.,  $C_L$  contains only  $v_i$ 's positive neighbors and  $C_R$  contains only  $v_i$ 's negative neighbors). By utilizing this fact, we can reduce the subgraph  $G[P_L \cup P_R \cup C_L \cup C_R]$  in the **Enum** procedure, by removing *conflicting* edges that violate structural balanced theory:

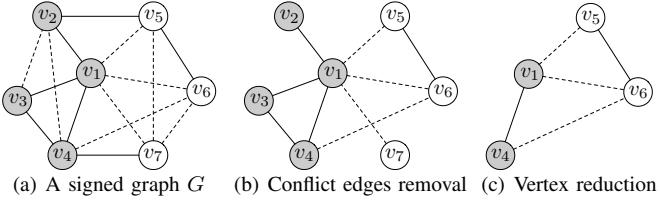


Fig. 4. Conflict edges removal and vertex reduction

- negative edges between vertices of  $P_L \cup C_L$ ,
- negative edges between vertices of  $P_R \cup C_R$ , and
- positive edges between a vertex of  $P_L \cup C_L$  and a vertex of  $P_R \cup C_R$ .

This approach is justified as vertices from  $P_L \cup C_L$  will belong to the same set of any derived balanced  $k$ -plex, and only positive edges among them contribute to forming a balanced  $k$ -plex. Thus, removing negative edges between vertices of  $P_L \cup C_L$  does not compromise the algorithm's correctness. Similarly, negative edges between vertices of  $P_R \cup C_R$  can also be removed. Additionally, since any vertex  $v \in P_L \cup C_L$  and any vertex  $v' \in P_R \cup C_R$  will be on opposite sets of any derived balanced  $k$ -plex, only the negative edges between them will be counted for the form of balanced  $k$ -plex. Therefore, we can safely remove positive edges between vertices of  $P_L \cup C_L$  and  $P_R \cup C_R$ .

Removing conflicting edges offers two significant advantages. Firstly, it leads to the sparsification of the subgraph  $G[P_L \cup P_R \cup C_L \cup C_R]$  that the current search instance is focusing on. Secondly, it enhances the efficiency of the vertex reduction methods proposed in Section III-A, resulting in further reductions in the computational costs. The following example illustrates this benefit.

**Example 1.** Consider the signed graph  $G$  in Figure 4(a) with  $k = 2$  and  $\tau = 2$ . Suppose  $v_1$  precedes all other vertices in the ordering  $\eta$ . When processing  $v_1$  with  $S = \emptyset$ , we have  $P_L = \{v_1\}$ ,  $P_R = \emptyset$ ,  $C_L = \{v_2, v_3, v_4\}$ , and  $C_R = \{v_5, v_6, v_7\}$ . Conflicting edges to be removed include negative edges  $(v_2, v_3)$ ,  $(v_2, v_4)$ ,  $(v_5, v_7)$ , and  $(v_6, v_7)$ , as well as positive edges  $(v_2, v_5)$  and  $(v_4, v_7)$ . The subgraph after removing these conflicting edges is shown in Figure 4(b). We can then apply vertex reduction to further prune unpromising vertices, resulting in the final reduced subgraph shown in Figure 4(c). Note that without removing conflicting edges, the vertex reduction would not be able to identify unpromising vertices.

**Reduce the number of generated subset  $S$ .** Our new algorithm MBPE makes candidate sets disjoint by considering all possible subsets  $S$  of size at most  $k - 1$  from  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$ . In the worst case, the size of  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$  can be as large as  $\delta D$ , leading to a substantial number of generated instances. We aim to prune unpromising vertices from  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$ . This is challenging since the vertices in  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$  can be placed in either  $P_L$  or  $P_R$  of the initial balanced  $k$ -plex, making it difficult to derive the pruning criteria. The vertex reduction proposed in Section III-A is

ineffective here, since it does not consider the possibility that vertices in  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$  can be placed into different sets of a balanced  $k$ -plex. To address this, we propose a partition-based vertex reduction technique. The main idea is that, for each vertex  $u \in N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$ , we explore the cases where it is placed in  $P_L$  or  $P_R$ . In each case, we derive the minimum criteria that  $u$  must meet to be part of a valid balanced  $k$ -plex with  $v_i$ . If  $u$  fails to satisfy this criterion, it can be safely removed from  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$ .

Specifically, we partition  $N_{\succ_\eta}(v_i)$  into  $N_{\succ_\eta}^+(v_i) \cup N_{\succ_\eta}^-(v_i)$  and denote  $L = \{v_i\} \cup N_{\succ_\eta}^+(v_i)$ ,  $R = N_{\succ_\eta}^-(v_i)$ , and  $T = N_{\succ_\eta}^2(v_i)$ . For each vertex  $u \in L \cup R \cup T$ , we define its  $L$ -part positive (or negative) neighbors as  $N_L^+(u) = \{v \in L \mid (u, v) \in E^+\}$  (or  $N_L^-(u) = \{v \in L \mid (u, v) \in E^-\}$ ). We also define its  $L$ -part positive (or negative) degree as  $d_L^+(u) = |N_L^+(u)|$  (or  $d_L^-(u) = |N_L^-(u)|$ ). Similarly,  $R$ -part positive and negative neighbors/degrees are defined by substituting  $L$  with  $R$ . In MBPE, for each vertex  $v_i$ , before generating all possible subsets  $S$  from  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$ , we can reduce the size of  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$  using the following lemma.

**Lemma 7.** Consider a signed graph  $G = (V, E^+, E^-)$  and an arbitrary order of vertices  $\eta = \{v_1, \dots, v_n\}$ . For a vertex  $u \in N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$  to be part of a  $v_i$ -lead balanced  $k$ -plex, it must meet the following conditions:

- If  $u \in N_{\succ_\eta}^+(v_i)$ , it satisfies at least one of the following:
  - $d_L^+(u) \geq \tau - k - |S| \wedge d_R^-(u) \geq \tau - k - |S| + 1$
  - $d_R^+(u) \geq \tau - k - |S| + 2 \wedge d_L^-(u) \geq \tau - k - |S| + 2$
- If  $u \in N_{\succ_\eta}^-(v_i)$ , it satisfies at least one of the following:
  - $d_L^+(u) \geq \tau - k - |S| + 1 \wedge d_R^-(u) \geq \tau - k - |S| + 3$
  - $d_R^+(u) \geq \tau - k - |S| \wedge d_L^-(u) \geq \tau - k - |S| + 1$
- If  $u \in N_{\succ_\eta}^2(v_i)$ , it satisfies at least one of the following:
  - $d_L^+(u) \geq \tau - k - |S| + 1 \wedge d_R^-(u) \geq \tau - k - |S| + 3$
  - $d_R^+(u) \geq \tau - k - |S| + 2 \wedge d_L^-(u) \geq \tau - k - |S| + 2$

The pseudo-code of PartitionVertexReduction is shown in Algorithm 5. Firstly, it obtains the three vertex sets  $L$ ,  $R$ , and  $T$ , and compute the  $L$ -part and  $R$ -part positive/negative degrees for each vertex (Lines 1-2). Then it identifies the vertex  $u$  that violates the degree constraints in Lemma 7. If  $u$  is from  $L$ , it decreases the  $L$ -part positive (resp. negative) degree by one for each of  $u$ 's positive (resp. negative) neighbors in  $L \cup R \cup T$  (Lines 4-6). If  $u$  is from  $R$ , it decreases the  $R$ -part positive (resp. negative) degree by one for each of  $u$ 's positive (resp. negative) neighbors in  $L \cup R \cup T$  (Lines 7-9). Then, it removes  $u$  from  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$  (Line 10). Note that the removal of a vertex in  $T$  will not cause the update of  $L$ -part and  $R$ -part degrees of other vertices. The algorithm terminates when no vertex violates Lemma 7 (Line 3).

**Example 2.** Consider the signed graph  $G$  in Figure 5(a) and let  $k = 2, \tau = 2$ . Suppose  $v_1$  precedes all other vertices in the ordering  $\eta$ . We have  $L = \{v_1, v_2, v_3\}$ ,  $R = \{v_4, v_5, v_6\}$ , and  $T = \{v_7, v_8\}$ . Firstly let us see how to prune unpromising vertices when  $|S| = 0$ . In this case,  $v_2$  is unqualified since  $d_R^-(v_2) = 0 < \tau - k - |S| + 1 = 1$  and  $d_L^+(v_2) = 0 < \tau - k - |S| + 1 = 1$ .

---

**Algorithm 5:** PartitionVertexReduction

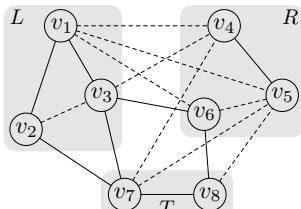
**Input:** The candidate set  $N_{\succ\eta}(v_i) \cup N_{\succ\eta}^2(v_i)$   
**Output:** Reduced candidate set

```

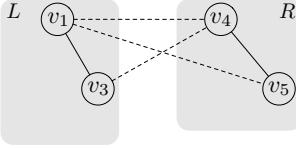
1  $L \leftarrow \{v_i\} \cup N_{\succ\eta}^+(v_i)$ ,  $R \leftarrow N_{\succ\eta}^-(v_i)$ ,  $T \leftarrow N_{\succ\eta}^2(v_i)$ ;
2 Compute  $d_L^+(u)$ ,  $d_L^-(u)$ ,  $d_R^+(u)$ ,  $d_R^-(u)$  for each  $u \in L \cup R \cup T$  ;
3 while  $\exists u \in N_{\succ\eta}(v_i) \cup N_{\succ\eta}^2(v_i)$  s.t.  $u \neq v_i$  violates Lemma 7 do
4   if  $u \in L$  then
5     for each  $v \in N_{L \cup R \cup P}^+(u)$  do  $d_L^+(v) \leftarrow d_L^+(v) - 1$ ;
6     for each  $v \in N_{L \cup R \cup P}^-(u)$  do  $d_L^-(v) \leftarrow d_L^-(v) - 1$ ;
7   else if  $u \in R$  then
8     for each  $v \in N_{L \cup R \cup P}^+(u)$  do  $d_R^+(v) \leftarrow d_R^+(v) - 1$ ;
9     for each  $v \in N_{L \cup R \cup P}^-(u)$  do  $d_R^-(v) \leftarrow d_R^-(v) - 1$ ;
10  Remove  $u$  from  $N_{\succ\eta}(v_i) \cup N_{\succ\eta}^2(v_i)$ ;
```

---

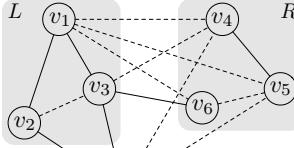
$|S| + 2 = 2$ . Besides,  $v_6$  is unqualified since  $d_R^-(v_6) = 1 < \tau - k - |S| + 3 = 3$  and  $d_R^+(v_6) + d_L^-(v_6) = 1 < 2\tau - k - |S| = 2$ , and  $v_7$  is unqualified since  $d_R^-(v_7) = 2 < \tau - k - |S| + 3 = 3$  and  $d_L^-(v_7) = 0 < \tau - k - |S| + 2 = 2$ .  $v_8$  is also unqualified due to the same reason as  $v_7$ . After removing all unpromising vertices,  $N_{\succ\eta}(v_i) \cup N_{\succ\eta}^2(v_i)$  is reduced as  $\{v_3, v_4, v_5\}$ , as shown in Figure 5(b). For the case of  $|S| = 1$ ,  $v_8$  is unqualified since  $d_R^-(v_8) = 1 < \tau - k - |S| + 3 = 2$  and  $d_L^-(v_8) = 0 < \tau - k - |S| + 2 = 1$ . In this case,  $N_{\succ\eta}(v_i) \cup N_{\succ\eta}^2(v_i)$  is reduced as  $\{v_2, v_3, v_4, v_5, v_6, v_7\}$ , as shown in Figure 5(c).



(a) An example signed graph



(b) The case of  $|S| = 0$



(c) The case of  $|S| = 1$

Fig. 5. Partition-based vertex reduction

**Theorem 3.** The time complexity of MBPE (Algorithm 4) is  $\mathcal{O}(n2^{k-1}(\delta D)^{k+1}2^\delta)$ , where  $\delta$  and  $D$  are the degeneracy and maximum degree of  $G$ , respectively.

#### B. Minimum Degree Pivot

In our algorithm MBPE, Enum is the most critical component and has exponential time complexity. To improve practical performance of our algorithm, we adopt a novel branching technique to speed up the enumeration procedure, which can reduce the generated branches in the search tree. In addition, we prove that this branching technique leads to a better time complexity of MBPE algorithm. The main idea

---

**Algorithm 6:** Enum-MDP

**Input:**  $P_L, P_R, C_L, C_R, X_L, X_R$   
**Output:** Maximal balanced  $k$ -plexes

```

1  $C_L \leftarrow \{v \in C_L : (P_L \cup \{v\}, P_R) \text{ is a balanced } k\text{-plex}\};$ 
2  $C_R \leftarrow \{v \in C_R : (P_L, P_R \cup \{v\}) \text{ is a balanced } k\text{-plex}\};$ 
3  $X_L \leftarrow \{v \in X_L : (P_L \cup \{v\}, P_R) \text{ is a balanced } k\text{-plex}\};$ 
4  $X_R \leftarrow \{v \in X_R : (P_L, P_R \cup \{v\}) \text{ is a balanced } k\text{-plex}\};$ 
5 if  $C_L \cup C_R = \emptyset$  and  $X_L \cup X_R = \emptyset$  then
6   if  $|P_L| \geq \tau$  and  $|P_R| \geq \tau$  then emit  $(P_L, P_R)$ ;
7   return
8   /* denote  $g = g(P_L \cup C_L, P_R \cup C_R)$  */
9 u  $\leftarrow$  the vertex of minimum degree in  $g$ ;
10 if  $d_g(u) \geq V(g) - k$  then
11   if  $g$  is maximal and satisfies the size constraint then
12     emit  $(P_L \cup C_L, P_R \cup C_R)$ ;
13   Let  $S = \{v_1, \dots, v_{q_2}\}$  be the set of vertices in  $(C_L \cup C_R) \setminus N_g(u)$ ;
14    $k' \leftarrow k - |(P_L \cup P_R) \setminus N_g(u)|$ ;
15   for each  $i \in \{1, \dots, k'\}$  do
16     if  $v_i \in C_L$  then
17        $C_L \leftarrow C_L \setminus \{v_i\}$ ;
18       Enum-MDP( $P_L, P_R, C_L, C_R, X_L \cup \{v_i\}, X_R$ );
19        $P_L \leftarrow P_L \cup \{v_i\}$ ;  $X_R \leftarrow X_R \setminus \{v_i\}$ ;
20     else
21        $C_R \leftarrow C_R \setminus \{v_i\}$ ;
22       Enum-MDP( $P_L, P_R, C_L, C_R, X_L, X_R \cup \{v_i\}$ );
23        $P_R \leftarrow P_R \cup \{v_i\}$ ;  $X_L \leftarrow X_L \setminus \{v_i\}$ ;
24   Enum-MDP( $P_L, P_R, C_L \setminus S, C_R \setminus S, X_L, X_R$ );
25 else if  $u \in C_L$  then
26   Enum-MDP( $P_L, P_R, C_L \setminus \{u\}, C_R, X_L \cup \{u\}, X_R$ );
27   Enum-MDP( $P_L \cup \{u\}, P_R, C_L \setminus \{u\}, C_R, X_L, X_R \setminus \{u\}$ );
28 else
29   Enum-MDP( $P_L, P_R, C_L, C_R \setminus \{u\}, X_L, X_R \cup \{u\}$ );
30   Enum-MDP( $P_L, P_R \cup \{u\}, C_L, C_R \setminus \{u\}, X_L \setminus \{u\}, X_R$ );
```

---

of this branching strategy is to select the vertex  $u \in P_L \cup P_R$  with the minimum degree in  $g(P_L \cup C_L, P_R \cup C_R)$ . Then, based on  $u$ , we expand the partial balanced  $k$ -plex  $(P_L, P_R)$  by prioritizing the candidate vertices in  $C_L \cup C_R$  that are  $u$ 's non-neighbors in  $g(P_L \cup C_L, P_R \cup C_R)$ . When the number of  $u$ 's non-neighbors in  $g(P_L, P_R)$  reaches  $k$ , we no longer generate branches for  $u$ 's remaining non-neighbors in  $C_L \cup C_R$ . In this way, the number of generated branches in the enumeration process can be reduced significantly. For simplicity, we will refer to  $g(P_L \cup C_L, P_R \cup C_R)$  as  $g$  in the following sections.

The pseudo-code of Enum-MDP is shown in Algorithm 6. First,  $C_L, C_R, X_L$  and  $X_R$  are refined by removing vertices that cannot form a larger balanced  $k$ -plex with the current balanced  $k$ -plex  $(P_L, P_R)$  (Lines 1-4). If these four sets are empty and  $(P_L, P_R)$  satisfies the size constraint, it emits this maximal balanced  $k$ -plex (Lines 5-7). Next, the algorithm selects a vertex  $u$  with the minimum degree in  $g$  (Line 8) and check if  $g$  is maximal and satisfies the size constraint. If so, it emits  $(P_L \cup C_L, P_R \cup C_R)$  and terminates the current branch (Lines 9-11). Otherwise, we know that  $u$  has more than  $k$  non-neighbors in  $g$  and branches are generated as follows:

- If  $u \in P_L \cup P_R$ , let  $q_1 = |(P_L \cup P_R) \setminus N_g(u)|$ , representing the number of  $u$ 's non-neighbors in  $P_L \cup P_R$ , and  $q_2 = |(C_L \cup C_R) \setminus N_g(u)|$ , representing the number of  $u$ 's non-

neighbors in  $C_L \cup C_R$ . Since  $u$  has more than  $k$  non-neighbors in  $g$ , we have  $q_1 < k$  (otherwise,  $u$  cannot have more than  $k$  non-neighbors in  $g$  due to the refinement of  $C_L$  and  $C_R$  at Lines 1-2). Let  $k' = k - q_1$ . At most  $k'$  vertices in  $(C_L \cup C_R) \setminus N_g(u)$  can be included in the current balanced  $k$ -plex. If we use  $\{v_1, \dots, v_{q_2}\}$  to denote  $u$ 's non-neighbors in the candidate set  $C_L \cup C_R$ ,  $k' + 1$  branches will be generated (Lines 12-24):

- 1) The first branch moves  $v_1$  from  $C_L$  (resp.  $C_R$ ) to  $X_L$  (resp.  $X_R$ );
  - 2) The second branch moves  $v_1$  from  $C_L$  (resp.  $C_R$ ) to  $P_L$  (resp.  $P_R$ ) and  $v_2$  from  $C_L$  (resp.  $C_R$ ) to  $X_L$  (resp.  $X_R$ );
  - 3) The  $i$ -th ( $3 \leq i \leq k'$ ) branch moves  $\{v_1, \dots, v_{i-1}\}$  from  $C_L$  (resp.  $C_R$ ) to  $P_L$  (resp.  $P_R$ ) and  $v_i$  from  $C_L$  (resp.  $C_R$ ) to  $X_L$  (resp.  $X_R$ );
  - 4) The last branch moves  $\{v_1, \dots, v_{k'}\}$  from  $C_L$  (resp.  $C_R$ ) to  $P_L$  (resp.  $P_R$ ) and the rest of  $u$ 's non-neighbors  $\{v_{k'+1}, \dots, v_{q_2}\}$  are removed from  $C_L$  (resp.  $C_R$ ) to  $X_L$  (resp.  $X_R$ ).
- If  $u \notin P_L \cup P_R$ , two branches are generated by moving  $u$  from  $C_L$  (resp.  $C_R$ ) to  $X_L$  (resp.  $X_R$ ) and by moving  $u$  from  $C_L$  (resp.  $C_R$ ) to  $P_L$  (resp.  $P_R$ ) for the case of  $u \in C_L$  (resp.  $u \in C_R$ ) (Lines 25-30). The second branch will fall into the first case.

The maximality check at Line 10 can be efficiently performed by verifying if there is any vertex in  $X_L \cup X_R$  that can be added to  $g$  to form a larger balanced  $k$ -plex. This process can be efficient since if there is a vertex  $v$  in  $g$  that has already missed  $k$  vertices, we only need to consider  $v$ 's neighbors in  $X_L \cup X_R$ . By substituting the Enum procedure with Enum-MDP in MBPE (Algorithm 4), we obtain a new algorithm, MBPE\*. The time complexity of MBPE\* is analyzed in the theorem below.

**Theorem 4.** *The time complexity of MBPE\* is  $\mathcal{O}(n^{2^{k-1}}(\delta D)^{k+1}\alpha_k^\delta)$ , where  $\delta$  and  $D$  are the degeneracy order and the maximum degree of the graph, respectively, and  $\alpha_k$  is strictly smaller than 2 and depends only on  $k$ , e.g.,  $\alpha_2 = 1.839$ ,  $\alpha_3 = 1.928$ , and  $\alpha_4 = 1.966$ .*

#### C. Extension to Strict Balanced $k$ -plex Computation

Our algorithm can be easily adapted for enumerating the maximal strict balanced  $k$ -plex (Definition 3). Specifically, at Line 3 of Algorithm 4, we replace  $S \subseteq N_{>\eta}(v_i) \cup N_{>\eta}^2(v_i)$  by  $S \subseteq N_{>\eta}^2(v_i)$ . This is because a strict balanced  $k$ -plex requires each vertex to be positively connected to same-set vertices and negatively connected to opposite-set vertices. Therefore,  $S$  only contains  $v_i$ 's 2-hop neighbors, and  $S_L$  and  $S_R$  will always be empty. The subsequent initialization of six vertex sets and the invocation of the Enum procedure follow the same way. We refer to this adapted algorithm for maximal strict balanced  $k$ -plex enumeration as MBPE\*-Strict.

## V. EXPERIMENTS

In this section, we empirically evaluate the efficiency and effectiveness of our algorithms. We evaluate the following

TABLE I  
STATISTICS OF DATASETS

Dataset	$ V $	$ E $	$\frac{ E^- }{ E }$	$D$	$\delta$	Category
Bitcoin	5,881	21,492	15%	795	21	Trade
AdjWordNet	16,259	76,845	32%	477	63	Language
WikiElection	7,118	100,693	22%	1065	53	Editing
Reddit	54,075	220,151	8%	4,164	76	Social
Referendum	10,884	251,406	5%	2,784	104	Political
Slashdot	82,140	500,481	24%	2,548	54	Social
Epinions	131,828	841,372	17%	3,558	121	Social
WikiConflict	116,836	2,027,871	62%	20,153	145	Editing
DBLP	2,387,365	11,915,023	72%	3,282	448	Cooperation
SN1	2,000,000	83,326,093	29%	1,301	489	Synthetic
SN2	3,000,000	124,990,528	31%	1,448	538	Synthetic

algorithms:

- MBPE-BK: The baseline algorithm (Algorithm 1) equipped with all optimizations proposed in Section III-A except the pivoting technique.
- MBPE-BK\*: MBPE-BK equipped with the pivoting technique.
- MBPE: The advanced algorithm (Algorithm 4) proposed in Section IV-A, equipped with subgraph sparsification and partition based vertex reduction.
- MBPE\*: MBPE equipped with minimum degree pivot technique, i.e., by replacing Enum with Enum-MDP (Algorithm 6).

All the algorithms are implemented in C++, compiled with g++ 7.5.0 with the -O3 flag.<sup>2</sup> All the experiments are conducted on a machine with an Intel Core-i7 3.20GHz CPU and 64GB RAM running Ubuntu 18.04. The time cost is measured as the amount of wall-clock time elapsed during the program's execution. The time limit is set as 20 hours.

**Datasets.** We evaluate our algorithms on nine publicly-available real-world signed graphs, whose main characteristics are summarized in Table I. Bitcoin<sup>3</sup> and Epinions<sup>3</sup> are who-trusts-whom graphs of users on Bitcoin OTC and Epinions, respectively. AdjWordNet<sup>4</sup> captures the synonym-and-antonym relation among adjectives in the English language. Wiki-Election<sup>5</sup> is the graph of users from the English Wikipedia that voted for and against each other during administration elections. Reddit<sup>3</sup> captures the sentiment among different subreddits on Reddit. Referendum [25] records Twitter data on the 2016 Italian Referendum, where interactions are negative for users with conflicting stances and positive otherwise. Slashdot<sup>3</sup> contains friend/foe links among Slashdot users. WikiConflict<sup>5</sup> represents positive and negative edit conflicts among the users of the English Wikipedia. DBLP is the signed network used in [26], which is downloaded from the dblp database<sup>6</sup> and processed in the same way as [26]. In addition, we also evaluate the algorithms on two synthetic datasets, SN1 and SN2, which are generated by the synthetic signed network generator SRN with default settings [27].

<sup>2</sup>Source code is available at <https://github.com/kyaocs/MBPE>

<sup>3</sup><https://snap.stanford.edu/>

<sup>4</sup><https://wordnet.princeton.edu/>

<sup>5</sup><https://konect.cc>

<sup>6</sup><https://dblp.uni-trier.de/>

TABLE II  
THE RUNNING TIME OF ENUMERATING MAXIMAL BALANCED  $k$ -PLEXES  
BY VARYING  $\tau$  ( $k = 2$ )

Graphs	$\tau$	Running time (sec)			
		MBPE-BK	MBPE-BK*	MBPE	MBPE*
Bitcoin	3	0.19	0.077	0.028	<b>0.021</b>
	4	0.06	0.015	0.013	<b>0.008</b>
	5	0.05	0.006	0.004	<b>0.002</b>
AdjWordNet	3	8930.63	15.15	8.39	<b>7.01</b>
	4	7923.66	9.57	4.26	<b>3.26</b>
	5	1929.86	4.69	2.30	<b>1.60</b>
WikiElection	3	15.38	14.31	1.21	<b>1.17</b>
	4	0.92	0.92	0.23	<b>0.23</b>
	5	0.87	0.87	0.16	<b>0.16</b>
Reddit	3	64.94	58.85	4.57	<b>4.45</b>
	4	7.77	7.67	0.69	<b>0.66</b>
	5	5.90	5.90	0.29	<b>0.29</b>
Referendum	3	1341.34	34.75	12.21	<b>10.36</b>
	4	114.98	6.91	2.74	<b>1.85</b>
	5	10.94	1.26	0.63	<b>0.39</b>
Slashdot	3	9.94	8.86	2.41	<b>2.33</b>
	4	0.68	0.67	0.64	<b>0.62</b>
	5	0.52	0.56	0.52	<b>0.50</b>
Epinions	3	594.79	483.13	42.82	<b>40.69</b>
	4	92.53	73.86	8.53	<b>7.32</b>
	5	27.83	19.60	2.69	<b>2.31</b>
WikiConflict	3	1143.19	1055.63	<b>141.43</b>	142.22
	4	4.42	4.43	3.10	<b>3.10</b>
	5	2.56	2.44	2.39	<b>2.26</b>
DBLP	3	17394.57	4620.07	381.52	<b>320.31</b>
	4	7305.79	2746.66	53.63	<b>46.87</b>
	5	46.21	46.50	32.01	<b>28.50</b>
SN1	3	-	19486.49	2249.52	<b>2136.59</b>
	4	2139.66	2136.74	428.77	<b>406.65</b>
	5	502.34	501.92	392.01	<b>388.82</b>
SN2	3	-	33747.44	3495.38	<b>3371.46</b>
	4	3130.09	3128.66	643.63	<b>635.24</b>
	5	735.60	734.53	585.92	<b>583.66</b>

**Exp-1: Efficiency when varying  $\tau$ .** In this experiment, we evaluate the efficiency of the algorithm by varying  $\tau$  from 3 to 5 (with  $k$  fixed at 2). The running time for each algorithm is presented in Table II. The minimum running time for each instance is highlighted in bold. As  $\tau$  increases, the running time decrease. This is because a larger size constraint  $\tau$  enables the algorithm to prune more unpromising vertices early in the process. Notably, the pivot technique employed by MBPE-BK\* significantly accelerates its performance, making it approximately two orders of magnitude faster than MBPE-BK. For instance, on AdjWordNet with  $\tau = 4$ , MBPE-BK\* completes in 9.57 seconds, while MBPE-BK takes 7923.66 seconds. This highlights the effectiveness of the pivot technique. However, despite the gains made with the pivot technique, MBPE-BK\* remains slower than MBPE in all cases, with a gap of approximately one order of magnitude. For instance, on Reddit with  $\tau = 3$ , MBPE takes 4.57 seconds, whereas MBPE-BK\* takes 58.85 seconds. This discrepancy does demonstrate the effectiveness of the new search framework, thereby benefiting practical performance. The running time of MBPE and MBPE\* are comparable, with MBPE\* being slightly faster in most cases.

**Exp-2: Efficiency when varying  $k$ .** In this experiment, we evaluate the efficiency of the algorithm by varying  $k$  from 2 to 4 (setting  $\tau = k + 2$ ). The results, shown in Table III,

reveal that the running time of all algorithms increases with larger values of  $k$ . This is attributed to the fact that a larger  $k$  allows each member in a balanced  $k$ -plex to have more non-neighbors, significantly expanding the search space. In this experiment, the superiority of our minimum degree pivot technique becomes evident. MBPE\* consistently outperforms MBPE, especially when  $k$  becomes large. For instance, on Referendum with  $k = 4$ , MBPE\* is around 2 times faster than MBPE (i.e., 1676.66 seconds v.s. 3213.36 seconds), illustrating the improved efficiency from the minimum degree pivot technique.

TABLE III  
THE RUNNING TIME OF ENUMERATING MAXIMAL BALANCED  $k$ -PLEXES  
BY VARYING  $k$  ( $\tau = k + 2$ )

Graphs	$k$	Running time (sec)			
		MBPE-BK	MBPE-BK*	MBPE	MBPE*
Bitcoin	2	0.06	0.015	0.013	<b>0.008</b>
	3	0.35	0.079	0.097	<b>0.024</b>
	4	1.03	0.29	0.34	<b>0.081</b>
AdjWordNet	2	7923.66	9.57	4.26	<b>3.26</b>
	3	13492.20	564.26	50.34	<b>21.22</b>
	4	-	14356.45	920.71	<b>792.94</b>
WikiElection	2	0.92	0.92	0.23	<b>0.23</b>
	3	438.69	499.22	1.36	<b>1.31</b>
	4	-	-	97.51	<b>94.23</b>
Reddit	2	7.77	7.67	0.69	<b>0.66</b>
	3	5827.53	6667.67	84.66	<b>70.82</b>
	4	-	-	10403.30	<b>8759.93</b>
Referendum	2	114.98	6.91	2.74	<b>1.85</b>
	3	3818.18	670.82	122.70	<b>62.51</b>
	4	-	-	3213.36	<b>1676.66</b>
Slashdot	2	0.68	0.67	0.64	<b>0.62</b>
	3	1.75	1.45	0.68	<b>0.67</b>
	4	85.84	89.61	1.39	<b>1.34</b>
Epinions	2	92.53	73.86	8.53	<b>7.32</b>
	3	-	-	568.62	<b>465.23</b>
	4	-	-	53502.45	<b>46098.02</b>
WikiConflict	2	4.42	4.43	3.10	<b>3.10</b>
	3	2569.45	2613.50	6.50	<b>6.40</b>
	4	-	-	1413.70	<b>1389.39</b>
DBLP	2	7305.79	2746.66	53.63	<b>46.87</b>
	3	-	-	1614.75	<b>1487.92</b>
	4	-	-	37495.30	<b>29516.18</b>
SN1	2	2139.66	2136.74	428.77	<b>406.65</b>
	3	4921.64	4926.66	988.77	<b>936.65</b>
	4	-	-	18793.56	<b>18108.82</b>
SN2	2	3130.09	3128.66	643.63	<b>635.24</b>
	3	7470.37	7469.89	1560.08	<b>1548.87</b>
	4	-	-	27765.03	<b>25829.51</b>

**Exp-3: Evaluation of each technique.** In this experiment, we evaluate the effectiveness of each optimization technique by varying  $k$  from 2 to 4 (setting  $\tau = k + 2$ ). We employ the algorithm MBPE\* as our baseline and investigate three different variants of it. MBPE\*-A does not apply vertex reduction and edge reduction on the original graph (proposed in Section III). MBPE\*-B does not apply the subgraph sparsification technique (proposed in Section IV). MBPE\*-C does not apply the partition-based vertex reduction technique (proposed in Section IV). As shown in Table IV, MBPE\* consistently outperforms all other algorithms. For example, on WikiConflict, when  $k = 3$ , MBPE\* is three orders of magnitude faster than MBPE\*-A, which demonstrates the effectiveness of

our vertex and edge reduction techniques. On **Reddit**, when  $k = 3$ , MBPE\* is around 6 times faster than MBPE\*-B, showcasing the effectiveness of our subgraph sparsification technique. On **WikiElection**, when  $k = 4$ , MBPE\* is around one order of magnitude faster than MBPE\*-C, validating the effectiveness of our partition based vertex reduction. This experiment demonstrates that each of our optimization techniques contributes to the algorithm efficiency.

TABLE IV  
EVALUATION OF EACH TECHNIQUE BY VARYING  $k$  ( $\tau = k + 2$ )

Graphs	$k$	Running time (sec)			
		MBPE*-A	MBPE*-B	MBPE*-C	MBPE*
Bitcoin	2	0.27	0.009	0.009	<b>0.008</b>
	3	0.29	0.025	0.026	<b>0.024</b>
	4	0.36	0.090	0.081	<b>0.081</b>
AdjWordNet	2	62.82	3.38	3.80	<b>3.26</b>
	3	79.60	31.11	24.46	<b>21.22</b>
	4	865.18	1821.60	802.84	<b>792.94</b>
WikiElection	2	3.18	0.26	0.30	<b>0.23</b>
	3	8.89	4.97	3.92	<b>1.31</b>
	4	182.77	299.75	839.88	<b>94.23</b>
Reddit	2	199.03	1.26	0.83	<b>0.66</b>
	3	467.32	422.72	79.52	<b>70.82</b>
	4	-	-	36455.61	<b>8759.93</b>
Referendum	2	12.32	2.06	2.10	<b>1.85</b>
	3	94.94	77.46	66.61	<b>62.51</b>
	4	2477.84	2768.67	1825.48	<b>1676.66</b>
Slashdot	2	124.24	0.65	0.65	<b>0.62</b>
	3	235.22	0.71	0.70	<b>0.67</b>
	4	373.80	1.85	2.60	<b>1.34</b>
Epinions	2	2384.19	10.94	9.50	<b>7.32</b>
	3	4540.34	1775.48	531.59	<b>465.23</b>
	4	-	-	-	<b>46098.02</b>
WikiConflict	2	1498.75	3.14	3.24	<b>3.10</b>
	3	6129.94	12.72	19.10	<b>6.40</b>
	4	-	2383.00	11029.96	<b>1389.39</b>
DBLP	2	3394.64	53.14	159.25	<b>46.87</b>
	3	-	1957.65	9566.00	<b>1487.92</b>
	4	-	-	-	<b>29516.18</b>
SN1	2	36546.30	460.06	1513.33	<b>406.65</b>
	3	-	1112.40	8639.49	<b>936.65</b>
	4	-	-	-	<b>18108.82</b>
SN2	2	57143.52	696.56	2194.52	<b>635.24</b>
	3	-	1676.66	13239.49	<b>1548.87</b>
	4	-	-	-	<b>25829.51</b>

**Exp-4: Evaluation of different models by varying  $k$ .** In this experiment, we evaluate two proposed models: strict balanced  $k$ -plex (Definition 3) and balanced  $k$ -plex (Definition 4), by varying  $k$  from 2 to 4. The algorithm MBPE\* aims to detect maximal balanced  $k$ -plexes, while MBPE\*-Strict focuses on detecting maximal strict balanced  $k$ -plexes. As shown in Table V, MBPE\* can detect more results than MBPE\*-Strict in most cases, albeit with a slightly higher computational time. For instance, on **Reddit**, when  $k = 3$  and  $\tau = 5$ , there are 3646 maximal balanced 3-plexes compared to 1361 maximal strict balanced 3-plexes, and MBPE\* and MBPE\*-Strict use 70.82 seconds and 43.66 seconds, respectively. This outcome aligns with theoretical expectations, as the balanced  $k$ -plex model generalizes the strict balanced  $k$ -plex model, allowing each vertex to miss a few connections or slightly violate structural balanced theory.

**Exp-5: Synonym and antonym group detection on Adj-**

TABLE V  
EVALUATION OF DIFFERENT MODELS BY VARYING  $k$  ( $\tau = k + 2$ )

Graphs	$k$	MBPE*		MBPE*-Strict	
		# results	Time (sec)	# results	Time (sec)
Bitcoin	2	1026	0.008	1026	0.007
	3	7583	0.024	7583	0.023
	4	23739	0.081	23739	0.072
AdjWordNet	2	52374	3.26	52209	3.26
	3	3213645	21.22	3201555	20.98
	4	145899163	792.94	145455233	800.39
WikiElection	2	13	0.23	12	0.22
	3	7	1.31	5	1.00
	4	1	94.23	1	77.51
Reddit	2	752	0.66	520	0.64
	3	3646	70.82	1361	43.66
	4	11873	8759.93	2136	4897.61
Referendum	2	138924	1.85	138924	1.81
	3	3378920	62.51	3378920	62.11
	4	47040124	1676.66	47040124	1659.07
Slashdot	2	100	0.62	100	0.64
	3	40	0.67	40	0.60
	4	14	1.34	14	1.11
Epinions	2	319386	7.32	294265	7.26
	3	7961979	465.23	6702015	371.08
	4	146189731	46098.02	116817331	33502.45
WikiConflict	2	623	3.10	506	3.03
	3	762	6.40	528	5.93
	4	464	1389.39	291	1250.03
DBLP	2	612062	46.87	523948	43.54
	3	75018028	1487.92	63294958	1406.33
	4	820939485	29516.18	793049854	29176.66
SN1	2	102	406.65	97	423.59
	3	13	936.65	11	976.65
	4	2	18108.82	2	18108.82
SN2	2	120	635.24	114	650.01
	3	19	1548.87	16	1565.24
	4	1	25829.51	1	27462.79

**WordNet.** In this experiment, we evaluate the effectiveness of the balanced  $k$ -plex by detecting synonym and antonym groups on **AdjWordNet**. This dataset is the same as the one used for efficiency evaluation (see Table I), where each vertex represents an adjective and the edge between a pair of synonyms (resp. antonyms) is positive (resp. negative). Using our algorithm MBPE\*, we identify maximal balanced  $k$ -plexes with  $\tau = 6$  and  $k = 3$ . For comparison, we employ MBCEnum [11] to detect maximal balanced cliques under the same parameter  $\tau = 6$ . Our results reveal 8,857 maximal balanced  $k$ -plexes (with sizes ranging from 12 to 64) and 183 maximal balanced cliques (with sizes ranging from 12 to 60). Notably, some synonym and antonym groups are uniquely detected by MBPE\*, as  $k$ -plexes allow for missing connections. Representative results are shown in Table VI. For instance,  $P_1$  (shown in the second row) is a balanced 3-plex with  $|P_L| = 7$  and  $|P_R| = 7$ . Compared to the balanced clique detected by MBCEnum, the adjectives “up” and “pocket-size” (highlighted in gray) newly appear, since they do not connect to all other words. However, these two words fit well within the context of this synonym and antonym group. A similar phenomenon is observed in  $P_2$  (shown in the third row), which is a balanced 3-plex with  $|P_L| = 14$  and  $|P_R| = 16$ . In this case, “heavy” and “light” newly appear compared to the balanced clique detected by MBCEnum. From this experiment, we conclude that balanced  $k$ -plexes

TABLE VI  
CASE STUDY ON ADJWORDNET

$P_L$	$P_R$
big, high, immodest, <b>up</b> , large, proud, tall	humble, modest, <b>pocket-size</b> , lowly, miserable, low, small
dirty, black, cloudy, ill-defined, dark, <b>heavy</b> , opaque, preserved, stale, unclean, unclear, unfair....	clean, blank, clear, fair, fresh, <b>light</b> , neat, unclouded, uncontaminated, uninjected, white....
...	...

can identify more interesting synonym and antonym groups in AdjWordNet than balanced cliques.

**Exp-6: Conflicts discovery on wiki-RfA.** In this experiment, we evaluate the effectiveness of the balanced  $k$ -plex by detecting conflicts on wiki-RfA<sup>3</sup>. Wiki-RfA records the mutual voting results of Wikipedia editors running for administrator, which contains 11,381 Wikipedia members (voters and votees) forming 189,004 distinct voter/votee pairs. We build an undirected signed graph by regarding each user as a node and create a positive (or negative) edge between two nodes if there is an supporting (or opposing) vote between them. Our goal is to find polarized structures in wiki-RfA. First, we apply the MBCEnum algorithm [11], which identifies only maximal balanced cliques of size 6. Figures 6(a) to 6(d) display some of the detected maximal balanced cliques, with blue edges indicating positive connections and red edges indicating negative connections. Next, using our MBPE\* algorithm with  $k = 4$ , we discover a balanced 4-plex of size 13 with  $P_L = \{\text{"Crzrussian"}, \text{"Kusma"}, \text{"Blnguyen"}, \dots\}$  and  $P_R = \{\text{"KyLu"}, \text{"The Halo"}, \text{"Siva1979"}, \dots\}$  (Figure 6(e)). This balanced 4-plex is a more significant polarized structure and includes maximal balanced cliques shown in Figure 6. Interestingly, we observe three positive edges connecting vertices from opposite sets, such as the positive edge between “Kusma” and “Terence Ong”. This is allowed by our balanced  $k$ -plex model, and a limited number of such “conflicting edges” does not compromise the overall polarized structure. This highlights the advantages of our balanced  $k$ -plex model; by allowing slight deviations from structural balanced theory, our model can uncover larger and more intriguing patterns.

## VI. RELATED WORKS

Signed graph was firstly studied by Harary et al. [22], where the notion of structural balanced theory is introduced. The problem of finding the largest (in terms of vertex number) vertex-induced subgraph that is structural balanced, known as the *maximum balanced subgraph problem*, is studied by Figueiredo et al. [28] and Ordozgoiti et al. [6]. The problem is NP-hard. A branch-and-cut exact algorithm, which only works for graphs with up-to a few thousand vertices, is proposed by Figueiredo et al. [28]. Heuristic algorithms without any guarantee on the solution optimality are investigated in [6], [28]. As the maximum balanced subgraph problem does not limit the non-neighbors of each vertex in the identified subgraph, these techniques cannot be applied to our problem.

The notion of structural balanced clique is formulated by Chen et al. [11] and the detection of balanced cliques receives

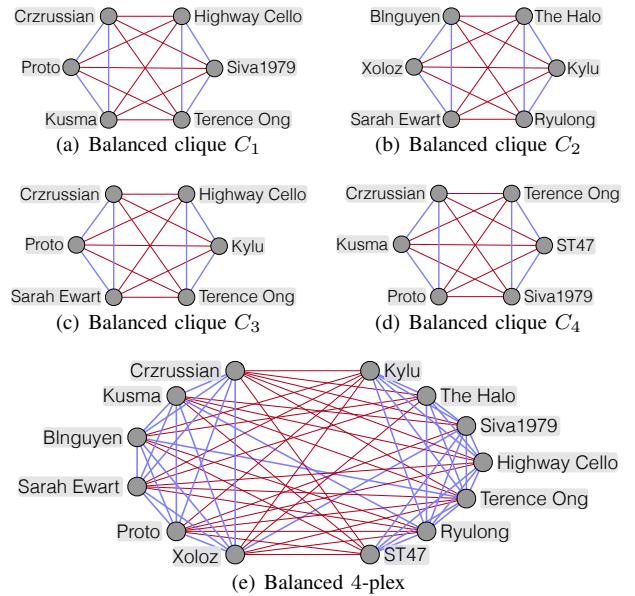


Fig. 6. Polarized structures detected in wiki-RfA

an increasing attention recently [11], [12], [14], [15]. However, balanced clique requires each pair of vertices are linked by an edge. Thus, their algorithms cannot be applied to our problem. Hao et al. [29] introduced the notion of  *$k$ -balanced trusted clique* for signed graphs, which is essentially the same problem as the traditional  $k$ -clique problem over unsigned graphs. Thus, the techniques proposed by Hao et al. [29] cannot be applied to solve our problem. The notion of  $(\alpha, k)$ -clique is defined by Li et al. [26] for signed graphs, which is a clique such that each vertex has at most  $k$  negative neighbors and at least  $\alpha k$  positive neighbors in the clique. As the structural balanced constraint is ignored, the techniques proposed by Li et al. [26] cannot be applied to our problem.

The problem of enumerating maximal  $k$ -plexes in unsigned graph has received extensive attention [21], [30]–[36]. Most of the maximal  $k$ -plex enumeration algorithms follow the framework of the Bron-Kerbosch algorithm, such as [30], [33], [34]. Aside from the Bron-Kerbosch variants, a polynomial delay algorithm is proposed by Berlowitz et al. [37]. Zhou et al. [35] proposed a novel branch scheme with a worst-case time complexity analysis. With their branch scheme, the running time of the algorithm is improved from  $\mathcal{O}^*(2^n)$  to  $\mathcal{O}^*(\gamma_k^n)$  in a graph with  $n$  vertices, where  $\gamma_k$  is related to  $k$  but strictly smaller than 2. Recently, Dai et al. [36] introduced a different branch strategy, which shares the same time complexity as the algorithm presented in [35]. Wang et al. [21] further improve the worst-case time complexity to  $\mathcal{O}^*(\gamma_k^\delta)$ , where  $\delta$  is the degeneracy of the graph. However, these techniques cannot be directly applied to our problem due to the existence of edge signs and the enforcement of structural balanced constraint.

## VII. CONCLUSION

In this paper, we formulated the balanced  $k$ -plex model in signed graphs and proved that the problem of enumer-

ating maximal balanced  $k$ -plexes is #P-hard. To solve this problem, we first introduced the basic backtracking algorithm MBPE-BK and proposed several optimization techniques to enhance its practical performance. Then, we proposed a novel algorithm MBPE with better time complexity than MBPE-BK. We also proposed subgraph sparsification and partition-based vertex reduction to improve its efficiency. Finally, we adopted the minimum-degree branching strategy to speed up the enumeration procedure, which improves the worst-case time complexity of MBPE. Experimental results on real-world and synthetic signed graphs demonstrated the efficiency of our algorithms and the effectiveness of our model.

## REFERENCES

- [1] Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. A survey of signed network mining in social media. *ACM Computing Surveys (CSUR)*, 49(3):1–37, 2016.
- [2] David A. Easley and Jon M. Kleinberg. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. Cambridge University Press, 2010.
- [3] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. The slashdot zoo: mining a social network with negative edges. In *Proc. of WWW’09*, pages 741–750, 2009.
- [4] Le Ou-Yang, Dao-Qing Dai, and Xiao-Fei Zhang. Detecting protein complexes from signed protein-protein interaction networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 12(6):1333–1344, 2015.
- [5] Lingyang Chu, Zhefeng Wang, Jian Pei, Jiannan Wang, Zijin Zhao, and Enhong Chen. Finding gangs in war from signed networks. In *Proc. of SIGKDD’16*, pages 1505–1514, 2016.
- [6] Bruno Ordozgoiti, Antonis Matakos, and Aristides Gionis. Finding large balanced subgraphs in signed networks. In *Proceedings of The Web Conference 2020*, pages 1378–1388, 2020.
- [7] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proc. of WWW’10*, pages 641–650, 2010.
- [8] Jihang Ye, Hong Cheng, Zhe Zhu, and Minghua Chen. Predicting positive and negative links in signed social networks by transfer learning. In *Proc. of WWW’13*, pages 1477–1488, 2013.
- [9] Chien Chin Chen, Yu-Hao Wan, Meng-Chieh Chung, and Yu-Chun Sun. An effective recommendation method for cold start new users using trust and distrust networks. *Information Sciences*, 224:19–36, 2013.
- [10] Jiliang Tang, Charu Aggarwal, and Huan Liu. Recommendations in signed social networks. In *Proceedings of the 25th International Conference on World Wide Web*, pages 31–40, 2016.
- [11] Zi Chen, Long Yuan, Xuemin Lin, Lu Qin, and Jianye Yang. Efficient maximal balanced clique enumeration in signed networks. In *Proceedings of The Web Conference 2020*, pages 339–349, 2020.
- [12] Zi Chen, Long Yuan, Xuemin Lin, Lu Qin, and Wenjie Zhang. Balanced clique computation in signed networks: Concepts and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- [13] Kai Yao, Lijun Chang, and Lu Qin. Computing maximum structural balanced cliques in signed graphs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1004–1016. IEEE, 2022.
- [14] Renjie Sun, Yanping Wu, Xiaoyang Wang, Chen Chen, Wenjie Zhang, and Xuemin Lin. Clique identification in signed graphs: A balance theory based model. *IEEE Transactions on Knowledge and Data Engineering*, 2023.
- [15] Kai Yao, Lijun Chang, and Lu Qin. Identifying large structural balanced cliques in signed graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2023.
- [16] Francesco Bonchi, Edoardo Galimberti, Aristides Gionis, Bruno Ordozgoiti, and Giancarlo Ruffo. Discovering polarized communities in signed networks. In *Proc. of CIKM’19*, pages 961–970, 2019.
- [17] Han Xiao, Bruno Ordozgoiti, and Aristides Gionis. Searching for polarization in signed graphs: a local spectral approach. In *Proceedings of The Web Conference 2020*, pages 362–372, 2020.
- [18] Apichat Suratanee, Martin H Schaefer, Matthew J Betts, Zita Soons, Heiko Mannsperger, Nathalie Harder, Marcus Oswald, Markus Gipp, Ellen Rammingen, Guillermo Marcus, et al. Characterizing protein interactions employing a genome-wide sirna cellular phenotyping screen. *PLoS Comput Biol*, 10(9):e1003814, 2014.
- [19] Kenneth E Read. Cultures of the central highlands, new guinea. *Southwestern Journal of Anthropology*, 10(1):1–43, 1954.
- [20] Coenraad Bron and Joep Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.
- [21] Zhengren Wang, Yi Zhou, Mingyu Xiao, and Bakhadyr Khoussainov. Listing maximal k-plexes in large real-world graphs. In *Proceedings of the ACM Web Conference 2022*, pages 1517–1527, 2022.
- [22] Frank Harary et al. On the notion of balance of a signed graph. *The Michigan Mathematical Journal*, 2(2):143–146, 1953.
- [23] Stephen B Seidman and Brian L Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology*, 6(1):139–154, 1978.
- [24] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983.
- [25] Mirko Lai, Viviana Patti, Giancarlo Ruffo, and Paolo Rosso. Stance evolution and twitter interactions in an italian political debate. In *Natural Language Processing and Information Systems: 23rd International Conference on Applications of Natural Language to Information Systems, NLDB 2018, Paris, France, June 13–15, 2018, Proceedings 23*, pages 15–27. Springer, 2018.
- [26] Rong-Hua Li, Qiangqiang Dai, Lu Qin, Guoren Wang, Xiaokui Xiao, Jeffrey Xu Yu, and Shaojie Qiao. Efficient signed clique search in signed networks. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 245–256. IEEE, 2018.
- [27] Yansen Su, Bangju Wang, Fan Cheng, Lei Zhang, Xingyi Zhang, and Linqiang Pan. An algorithm based on positive and negative links for community detection in signed networks. *Scientific reports*, 7(1):10874, 2017.
- [28] Rosa Maria Videira de Figueiredo and Yuri Frota. The maximum balanced subgraph of a signed graph: Applications and solution approaches. *Eur. J. Oper. Res.*, 236(2):473–487, 2014.
- [29] Fei Hao, Stephen S Yau, Geyong Min, and Laurence T Yang. Detecting k-balanced trusted cliques in signed social networks. *IEEE internet computing*, 18(2):24–31, 2014.
- [30] Bin Wu and Xin Pei. A parallel algorithm for enumerating all the maximal k-plexes. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 476–483. Springer, 2007.
- [31] Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *Journal of Computer and System Sciences*, 74(7):1147–1159, 2008.
- [32] Alessio Conte, Donatella Firmani, Caterina Mordente, Maurizio Patrignani, and Riccardo Torlone. Fast enumeration of large k-plexes. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 115–124, 2017.
- [33] Zhuo Wang, Qun Chen, Boyi Hou, Bo Suo, Zhanhuai Li, Wei Pan, and Zachary G Ives. Parallelizing maximal clique and k-plex enumeration over graph data. *Journal of Parallel and Distributed Computing*, 106:79–91, 2017.
- [34] Alessio Conte, Tiziano De Matteis, Daniele De Sensi, Roberto Grossi, Andrea Marino, and Luca Versari. D2k: scalable community detection in massive networks via small-diameter k-plexes. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1272–1281, 2018.
- [35] Yi Zhou, Jingwei Xu, Zhenyu Guo, Mingyu Xiao, and Yan Jin. Enumerating maximal k-plexes with worst-case time guarantee. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2442–2449, 2020.
- [36] Qiangqiang Dai, Rong-Hua Li, Hongchao Qin, Meihao Liao, and Guoren Wang. Scaling up maximal k-plex enumeration. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 345–354, 2022.
- [37] Devora Berlowitz, Sara Cohen, and Benny Kimelfeld. Efficient enumeration of maximal k-plexes. In *Proceedings of the 2015 ACM SIGMOD International conference on management of data*, pages 431–444, 2015.
- [38] Leslie G Valiant. The complexity of enumeration and reliability problems. *siam Journal on Computing*, 8(3):410–421, 1979.
- [39] Vladimir Batagelj and Matjaz Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.
- [40] Fedor V Fomin and Petteri Kaski. Exact exponential algorithms. *Communications of the ACM*, 56(3):80–88, 2013.

## APPENDIX A. PROOFS OF LEMMAS AND THEOREMS

**Proof of Theorem 1.** We prove the theorem by reducing from the problem of enumerating all maximal  $k$ -plexes that are of size at least  $\tau$  in an unsigned graph, which is  $\#P$ -complete [38]. Given an unsigned input graph  $G^u = (V^u, E^u)$  and an integer  $\tau$  of the maximal  $k$ -plex enumeration problem, we construct a signed graph  $G = (V, E^+, E^-)$  as follows. Firstly,  $V = V^u \cup \{v_1, \dots, v_\tau\}$  where  $v_i \notin V^u, \forall 1 \leq i \leq \tau$ . Denote  $\{v_1, \dots, v_\tau\}$  by  $V^a$ . Then,  $E^+ = E^u \cup \{(v_i, v_j) \mid v_i \in V^a, v_j \in V^a, v_i \neq v_j\}$  and  $E^- = \{(u, v) \mid u \in V^u, v \in V^a\}$ . We first prove by contradiction that every balanced  $k$ -plex  $(P_L, P_R)$  in  $G$  with  $\min\{|P_L|, |P_R|\} \geq \tau \geq k$  must satisfy either  $P_L \subseteq V^u \wedge P_R = V^a$  or  $P_L = V^a \wedge P_R \subseteq V^u$ .

- 1) Suppose  $P_L \subseteq V^u$  and  $P_R \subseteq V^u$ . Consider any vertex  $u \in P_L$ , we have  $d_{P_R}^-(u) = 0 \leq |P_R| - k$ , contradicting that  $(P_L, P_R)$  is a balanced  $k$ -plex.
- 2) Suppose  $P_L \cap V^a \neq \emptyset$  and  $P_R \cap V^a \neq \emptyset$ . Let  $X_1 = P_L \cap V^a$ ,  $X_2 = P_L \cap V^u$ ,  $Y_1 = P_R \cap V^a$  and  $Y_2 = P_R \cap V^u$ . Then, none of  $X_1, X_2, Y_1, Y_2$  is empty. Let's consider any vertex  $u \in X_2$ , then  $d_{P_L}^+(u) + d_{P_R}^-(u) \geq |P_L| + |P_R| - k$  and thus  $|X_1| + |Y_2| \leq k - 1$ . Similarly we have  $|X_2| + |Y_1| \leq k - 1$ , contradicting that  $|P_L| \geq \tau$  and  $|P_R| \geq \tau$ .
- 3) Suppose  $P_L \cap V^a \neq \emptyset$ ,  $P_L \cap V^u \neq \emptyset$  and  $P_R \subseteq V^u$ . Consider any vertex  $u \in P_L$ , we have  $d_{P_R}^-(u) = 0 \leq |P_R| - k$ , contradiction (similar to the first case).
- 4) Suppose  $P_R \cap V^a \neq \emptyset$ ,  $P_R \cap V^u \neq \emptyset$  and  $P_L \subseteq V^u$ . Similar to the third case above.

Without loss of generality, assume  $P_R = V^a$ . Then, it is easy to see that  $(P_L, P_R)$  is a maximal balanced  $k$ -plex in  $G$  with  $\min\{|P_L|, |P_R|\} \geq \tau$  if and only if  $P_L$  is a maximal  $k$ -plex in  $G^u$  with  $|P_L| \geq \tau$ .  $\square$

**Proof of Lemma 1.** Suppose that the partitioning is not unique, and let  $(P_L, P_R)$  and  $(P'_L, P'_R)$  be two possible ways to partition it. Denote  $X_1 = P_L \cap P'_L$ ,  $X_2 = P_L \cap P'_R$ ,  $Y_1 = P_R \cap P'_L$  and  $Y_2 = P_R \cap P'_R$ . Without loss of generality, assume  $X_1 \neq \emptyset$  and let  $u$  be an arbitrary vertex in  $X_1$ . Then, by the definition of balanced  $k$ -plex, it must be the case that:

- $d_{X_1 \cup X_2}^+(u) + d_{Y_1 \cup Y_2}^-(u) \geq |X_1| + |X_2| + |Y_1| + |Y_2| - k$
- $d_{X_1 \cup Y_1}^+(u) + d_{X_2 \cup Y_2}^-(u) \geq |X_1| + |X_2| + |Y_1| + |Y_2| - k$

By adding up both inequalities, we have

$$2d_{X_1}^+(u) + 2d_{Y_2}^-(u) + d_{X_2}^+(u) + d_{Y_1}^-(u) + d_{X_2}^-(u) \geq 2|X_1| + 2|X_2| + 2|Y_1| + 2|Y_2| - 2k \quad (1)$$

This is because  $d_{X_1 \cup X_2}^+(u) = d_{X_1}^+(u) + d_{X_2}^+(u)$ ,  $d_{Y_1 \cup Y_2}^-(u) = d_{Y_1}^-(u) + d_{Y_2}^-(u)$ ,  $d_{X_1 \cup Y_1}^+(u) = d_{X_1}^+(u) + d_{Y_1}^-(u)$ ,  $d_{X_2 \cup Y_2}^-(u) = d_{X_2}^-(u) + d_{Y_2}^-(u)$  since all sets are disjoint. Moreover, the LHS of Equation 1 satisfies:

$$LHS \leq 2(|X_1| - 1) + 2|Y_2| + |X_2| + |Y_1| \quad (2)$$

Combining Equations 1 and 2, we have

$$|X_2| + |Y_1| \leq 2k - 2. \quad (3)$$

Note that  $X_2$  and  $Y_1$  cannot be empty at the same time; otherwise the two ways of partitioning are the same. Without

loss of generality, we pick a vertex  $v \in X_2$ , following the same logic, we derive

$$|X_1| + |Y_2| \leq 2k - 2 \quad (4)$$

Combining Equations 3 and 4, we have

$$|X_1| + |X_2| + |Y_1| + |Y_2| \leq 4k - 4.$$

Hence we conclude the proof.  $\square$

**Proof of Lemma 2.** Since  $(P_L, P_R)$  is a balanced  $k$ -plex, we know the maximal balanced subgraph of  $G$  induced by  $(P_L, P_R)$ , i.e.,  $g(P_L, P_R)$ , must be a  $k$ -plex (after removing edges that violate structural balance), whose diameter is bounded by 2 [34]. Thus, we can easily know the diameter of the subgraph  $G[P_L \cup P_R]$  must also be bounded by 2.  $\square$

**Proof of Lemma 3.** We first prove that  $P_L \subseteq \{v\} \cup N_G(v) \cup N_G^{++}(v) \cup N_G^{--}(v)$ . For any other vertex  $u \in P_L$  such that  $u \neq v$  and  $u \notin N_G(v)$ ,  $u$  and  $v$  must have a common neighbor  $w$  in  $g(P_L, P_R)$ . This is because the diameter of this maximal structural balanced subgraph is bounded by 2 (Lemma 2). Moreover,  $w$ 's links to  $u$  and  $v$  must be of the same sign (i.e.,  $(u, w) \in E^+ \wedge (v, w) \in E^+$  or  $(u, w) \in E^- \wedge (v, w) \in E^-$ ) since  $g(P_L, P_R)$  is structural balanced. Thus, we derive that  $u \in N_G^{++}(v) \cup N_G^{--}(v)$ .  $P_R \subseteq N_G(v) \cup N_G^{+-}(v) \cup N_G^{-+}(v)$  can be proved in a similar way except the difference that since the vertices in  $P_R$  are at opposite side with  $v$ , then  $N_G^{++}(v) \cup N_G^{--}(v)$  should be replaced with  $N_G^{+-}(v) \cup N_G^{-+}(v)$ . Hence we conclude the proof.  $\square$

**Proof of Theorem 2.** Firstly, the time cost of computing degeneracy order of  $V$  is bounded by  $\mathcal{O}(m)$  [39]. Secondly, the for loop at Line 2 processes each vertex  $v_i$  by constructing the partial solution, candidate sets and exclusive sets, which can be done in  $\mathcal{O}(\delta D)$ . Then the set of  $v_i$ -lead maximal balanced  $k$ -plexes are computed by invoking `Enum`, which then recursively invokes itself (Lines 18 and 22). Note that the number of vertices in  $C_L, C_R, X_L$  and  $X_R$  is bounded by  $\delta D$  since we only consider the forward two-hop neighbors of each vertex  $v_i$ . For each  $v_i$ , the total number of invocations to `Enum` is at most  $3^{\delta D}$ , since in each `Enum` we reduce the size of  $C_L$  or  $C_R$  by at least 1 (Lines 17 and 21). The running time of `Enum` without considering Lines 18 and 22 is dominated by the refinement of  $C_L, C_R, X_L$  and  $X_R$  at Lines 9-12, whose time complexity is bounded by  $\mathcal{O}((\delta D)^2)$ . Thus, the total time complexity of Algorithm 1 is  $\mathcal{O}(m + \sum_{v_i \in V} (\delta D + (\delta D)^2 3^{\delta D})) = \mathcal{O}(n(\delta D)^2 3^{\delta D})$ .  $\square$

**Proof of Lemma 4.** For each vertex  $v$  in a balanced  $k$ -plex  $(P_L, P_R)$ , its positive degree and negative degree in  $g(P_L, P_R)$  should be at least  $\tau - k$  and  $\tau - k + 1$ , according to the definition of balanced  $k$ -plex. Thus,  $v$ 's positive degree and negative degree should also satisfy these conditions in  $G$ . In a similar way, we can also prove the degree of each vertex  $v$  in a balanced  $k$ -plex  $(P_L, P_R)$  should be at least  $2\tau - k$  in  $G$ . We complete the proof.  $\square$

**Proof of Lemma 5.** Suppose  $(u, v)$  is a positive edge in  $g(P_L, P_R)$ . Without loss of generality, we assume  $u, v \in P_L$ . In this case,  $u$  and  $v$  has at most  $k - 1$  non-neighbors from  $P_L$  in  $g(P_L, P_R)$ , respectively. Since  $|P_L| \geq \tau$ , we know that the number of vertices in  $P_L$  that positively link to both  $u$  and  $v$  is at least  $\tau - 2k$ . Thus, in  $G$ , which is a super graph of  $g(P_L, P_R)$ , the number of vertices that positively link to both  $u$  and  $v$  is also at least  $\tau - 2k$ . We have  $\Delta_G^{++}(u, v) \geq \tau - 2k$ . Also,  $u$  and  $v$  has at most  $k - 1$  non-neighbors from  $P_R$  in  $g(P_L, P_R)$ , respectively. Since  $|P_R| \geq \tau$ , we know that the number of vertices in  $P_R$  that negatively link to both  $u$  and  $v$  is at least  $\tau - 2k + 2$ . Thus, we have  $\Delta_G^{--}(u, v) \geq \tau - 2k + 2$ . Moreover,  $u$  and  $v$  has at most  $k - 1$  non-neighbors in  $g(P_L, P_R)$  respectively, and the size of  $g(P_L, P_R)$  is at least  $2\tau$ . Thus we have  $\Delta_G^{++}(u, v) + \Delta_G^{--}(u, v) \geq 2\tau - 2k$ . In a similar way, we can prove the conditions for the negative edge  $(u, v)$ . We omit the details.  $\square$

**Proof of Lemma 6.** We prove the lemma by contradiction. Suppose there exists a maximal balanced  $k$ -plex  $P'$  that does not satisfy the given condition:  $P'$  contains  $P$  but

- (1)  $P'$  does not contain  $u$ ,
- (2)  $P' \setminus P$  does not contain any  $v \in C$  such that  $v \neq u$  and  $u$  and  $v$  are non-neighbors in  $g(P \cup \{u, v\})$ ,
- (3)  $P' \setminus P$  does not contain any  $v \in C$  such that  $u$  and  $v$  are neighbors in  $g(P \cup \{u, v\})$  and they have common non-neighbors in  $g(P \cup \{u, v\})$ .

We will show that  $u$  can be added to  $P'$  to form a larger balanced  $k$ -plex. From (2), we know that  $P' \setminus P$  contains only neighbors of  $u$  in  $g(P' \cup \{u\})$ . Given that  $u$  is a valid candidate vertex and  $P \cup \{u\}$  is a balanced  $k$ -plex, the only thing left to prove is that  $u$ 's non-neighbors in  $g(P \cup \{u\})$  have at most  $k - 1$  non-neighbors in  $g(P' \cup \{u\})$ . This can be derived from (3), since it states that any vertex  $v \in P' \setminus P$  does not have common non-neighbors with  $u$  in  $g(P \cup \{u, v\})$ . Thus, the presence of  $u$  in  $P'$  would not cause its non-neighbors in  $g(P \cup \{u\})$  to have more than  $k$  non-neighbors. Hence,  $P' \cup \{u\}$  is also a balanced  $k$ -plex, which contradicts the claim that  $P'$  is maximal.  $\square$

**Proof of Lemma 7.** Firstly, let us consider that the vertex  $u \in N_{\succ_\eta}^+(v_i)$ . Suppose  $P' = (P'_L, P'_R)$  is a maximal  $v_i$ -lead balanced  $k$ -plex derived from this instance and  $u \in P'$ . There are two cases (1)  $u \in P'_L$  and (2)  $u \in P'_R$ . For the first case, the size of  $P'_L$  is at most  $d_L^+(u) + k + |S|$  since  $u$  has at most  $k$  non-neighbors or wrong-signed neighbors in  $P'_L$ . Also, at most  $|S|$  vertices in  $P'_L$  may come from  $R$  and  $T$ , and most ideally all of them are positive neighbors of  $u$ . Combining with the size constraint  $\tau$ , we have  $d_L^+(u) \geq \tau - k - |S|$ . Similarly, we can derive that  $d_R^-(u) \geq \tau - k - |S| + 1$ . For the second case that  $u$  is in the opposite side with  $v_i$ , i.e.,  $u \in P'_R$ .  $u$  and  $v_i$  are wrong-signed neighbors with each other. In this case, the size of  $P'_L$  is at most  $d_L^-(u) + k - 1 + |S| - 1$  since  $u$  has at most  $k - 1$  non-neighbors or wrong-signed neighbors in  $P'_L$ . Also, at most  $|S| - 1$  vertices in  $P'_L$  may come from  $R$  and  $T$ , and most ideally all of them are negative neighbors of  $u$ . Thus, we can derive that  $d_L^-(u) \geq \tau - k - |S| + 2$ . Similarly, we can also derive that  $d_R^+(u) \geq \tau - k - |S| + 2$ . The degree

constraints of the vertex  $u \in N_{\succ_\eta}^-(v_i)$  and  $u \in N_{\succ_\eta}^2(v_i)$  can be derived similarly. We omit the details.  $\square$

**Proof of Theorem 3.** Firstly, the time cost of computing degeneracy order of  $V$  is bounded by  $\mathcal{O}(m)$  [39]. For each  $v_i$  in the degeneracy order, we reduce the search space to its 2-hop neighbors  $N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$ . Then we enumerate all subsets  $S \subseteq N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)$  with size at most  $k - 1$ .  $S$  is then added to  $P_L$  and  $P_R$  to generate a partial solution. This will give in total  $|N_{\succ_\eta}(v_i) \cup N_{\succ_\eta}^2(v_i)|^{k-1} = \mathcal{O}((\delta D)^{k-1})$  instances. For each instance, we will partition it at most another  $2^{|S|} = \mathcal{O}(2^{k-1})$  times and invoke **Enum**. Here, the size of  $C_L$  and  $C_R$  is bounded by  $\delta$  and the size of  $X_L$  and  $X_R$  is bounded by  $\delta D$ . As the analysis in Theorem 2, for each instance the time complexity of **Enum** is bounded by  $\mathcal{O}((\delta D)^2 2^\delta)$ . Thus, the total time complexity is  $\mathcal{O}(m + \sum_{v_i \in V} ((\delta D)^{k-1} 2^{k-1} (\delta D)^2 2^\delta)) = \mathcal{O}(n 2^{k-1} (\delta D)^{k+1} 2^\delta)$ .  $\square$

**Proof of Theorem 4.** We only need to prove the time complexity of **Enum-MDP**. When  $C_L \cup C_R = \emptyset$ , it means the candidate sets have been exhausted and no branching is needed. Then we can analyze from the perspective of the number of vertices removed from  $C_L \cup C_R$ . When the minimum degree vertex  $u$  is from  $P_L \cup P_R$ , the algorithm will generate  $k' + 1$  branches. In the first branch, one vertex (i.e.,  $\{v_1\}$ ) is removed from  $C_L \cup C_R$ . In the second branch, two vertices (i.e.,  $\{v_1, v_2\}$ ) are removed from  $C_L \cup C_R$ . In the  $i$ -th branch ( $3 \leq i \leq k'$ ),  $i$  vertices (i.e.,  $\{v_1, \dots, v_i\}$ ) are removed from  $C_L \cup C_R$ . Finally, in the last branch,  $q_2$  vertices (i.e.,  $\{v_1, \dots, v_{q_2}\}$ ) are removed from  $C_L \cup C_R$ . Note that we have  $q_2 > k'$  since  $q_1 + q_2 > k$ . If we use  $T(c)$  to represent the number of branches generated by **Enum-MDP** on the instance  $C_L \cup C_R$  of size  $c$ , then we can obtain the following recurrence:

$$T(c) \leq T(c-1) + \dots + T(c-k') + T(c-q_2).$$

When  $u \in C_L \cup C_R$ , the algorithm will generate two branches. The first branch removes  $u$  from  $C_L \cup C_R$  to  $X_L \cup X_R$ . The second branch removes  $u$  from  $C_L \cup C_R$  to  $P_L \cup P_R$ , which follows with the above recurrence. Combining them together, we have:

$$T(c) \leq T(c-1) + \dots + T(c-k'-1) + T(c-q_2-1).$$

Given  $k' \leq k - 1$  and  $q_2 \geq k' + 1$ , the worst case is when  $k' = k - 1$  and  $q_2 = k' + 1 = k$ . After substituting into the above recurrence, we obtain:

$$T(c) \leq T(c-1) + \dots + T(c-k) + T(c-k-1).$$

Based on the theoretical result in [40], the branching factor  $\alpha_k$  of the above recurrence is the largest real root of function  $x^{k+2} - 2x^{k+1} + 1 = 0$ . Note that  $\alpha_k$  is strictly smaller than 2 and only depends on  $k$ . For example,  $\alpha_2 = 1.839$ ,  $\alpha_3 = 1.928$ , and  $\alpha_4 = 1.966$ . We also have  $c \leq \delta$  since **Enum-MDP** starts with  $C_L \cup C_R$  being  $N_{\succ_\eta}^+(v_i) \cup N_{\succ_\eta}^-(v_i)$ . In addition, the running time of **Enum-MDP** without considering the recursive invocations of itself is dominated by the refinement of  $C_L$ ,  $C_R$ ,  $X_L$  and  $X_R$  and the minimum-degree vertex selection, whose

time complexity can be bounded by  $\mathcal{O}((\delta D)^2)$ . Thus, the running time of `Enum-MDP` is  $\mathcal{O}((\delta D)^2 \alpha_k^\delta)$ . As the analysis in Theorem 3, at most  $(2\delta D)^{k-1}$  instances will be generated for

each vertex  $v_i$ . Thus, the time complexity of `MBPE*` is  $\mathcal{O}(m + \sum_{v_i \in V} ((2\delta D)^{k-1} (\delta D)^2 \alpha_k^\delta)) = \mathcal{O}(n 2^{k-1} (\delta D)^{k+1} \alpha_k^\delta)$ .  $\square$