# Efficient Antagonistic k-plex Enumeration in Signed Graphs

Lantian Xu, Rong-Hua Li, Dong Wen, Qiangqiang Dai, Guoren Wang, Lu Qin

Abstract—A signed graph is a graph where each edge receives a sign, positive or negative. The signed graph model has been used in many real applications, such as protein complex discovery and social network analysis. Finding cohesive subgraphs in signed graphs is a fundamental problem. A k-plex is a common model for cohesive subgraphs in which every vertex is adjacent to all but at most k vertices within the subgraph. In this paper, we propose the model of size-constrained antagonistic k-plex in a signed graph. The proposed model guarantees that the resulting subgraph is a k-plex and can be divided into two sub-k-plexes, both of which have positive inner edges and negative outer edges. This paper aims to identify all maximal antagonistic k-plexes in a signed graph. Through rigorous analysis, we show that the problem is NP-Hardness. We propose a novel framework for maximal antagonistic k-plexes utilizing set enumeration. Efficiency is improved through pivot pruning and early termination based on the color bound. Preprocessing techniques based on degree and dichromatic graphs effectively narrow the search space before enumeration. Extensive experiments on real-world datasets demonstrate our algorithm's efficiency, effectiveness, and scalability.

Index Terms—Signed graph, k-plex, Antagonistic communities.

# I. INTRODUCTION

S IGNED graphs serve as effective tools for representing the polarity of relationships between entities, employing positive and negative symbols to denote the associations between the respective vertices. These graphs find applications in diverse domains, such as capturing friend-foe relationships in social networks [1], expressing support-dissent opinions within opinion networks [2], characterizing trust-distrust relationships in trust networks [3], and depicting activation-inhibition dynamics in protein-protein interaction networks [4].

Structural balance theory is an essential and foundational theory in the analysis of signed graphs. According to this theory, a signed graph denoted as G is considered balanced if it can be partitioned into two distinct subgraphs, where edges within each subgraph are positive, and the edges connecting vertices from different subgraphs are negative [5]. That is, "The friend (resp. enemy) of my friend(resp. enemy) is my friend, the friend (resp. enemy) of my enemy (resp. friend) is my enemy". We can find the antagonistic sub-communities as the localized effect of social balance theory [6]. Consider the graph G shown in Figure 1, solid (resp. dashed) lines

Lantian Xu and Lu Qin are with the Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, Australia.

Rong-Hua Li, Qiangqiang Dai and Guoren Wang are with Beijing Institute of Technology (BIT), Beijing, China.

Dong Wen is with the University of New South Wales, Sydney, Australia. Manuscript received April 19, 2021; revised August 16, 2021.

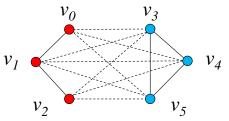


Fig. 1. Balanced graph

represent positive (resp. negative) edges. G can be divide into two parts, one part is  $\{v_0, v_1, v_2\}$ , the other part is  $\{v_3, v_4, v_5\}$ . Although there are no egdes between  $v_0$  and  $v_2$ ,  $v_0$  and  $v_2$  have the common friend  $v_1$  and common enemies  $v_3, v_4$  and  $v_5$ . According to the structural balance theory,  $v_0$  and  $v_2$  can also be regarded as friends.

Some cohesive subgraph models in signed graphs have also been investigated in the literature. [7] proposed the definition of balanced clique and gave the maximal balanced clique search algorithm. Based on this, [8] further proposed a search algorithm for a maximum balanced clique. Due to data noise, clique, where vertices are pairwise connected, can rarely appear in real data [9], [10]. It may be too strict to use clique to find cohesive subgraphs. In order to more accurately mine cohesive subgraphs in signed graphs, a relaxed model of cliques is expected.

The k-plex is a crucial cohesive subgraph model adopted extensively in graph analysis. However, its relevant definitions and algorithms for signed networks are less studied. We introduce the maximal antagonistic k-plex model for cohesive subgraphs in signed graphs to address this gap and draw inspiration from structural balance theory [5]. Formally, given a signed network G, a maximal antagonistic k-plex C is a maximal subgraph of G such that (1) C is k-plex without considering the symbols of edges. (2) C is antagonistic, i.e., C can be divided into two parts such that the edges in the same part are positive, and the edges connecting two parts are negative. To derive large k-plexes, we specify the minimum size of each antagonistic part as a parameter t. This definition incorporates both density and balance considerations. We prove that the maximal antagonistic k-plex problem is NPhard. The primary objective of our paper is to develop efficient algorithms for enumerating all maximal antagonistic k-plex subgraphs within a given signed network.

**Applications.** We show several applications of maximal antagonistic k-plex enumeration as follows.

- Antagonistic group detection. People often share or argue with each other on social networks such as Facebook and Quora. Users with agreements can be represented by positive edges, while negative edges can represent those with disagree-

ments. Consequently, social network data can be modeled as a signed graph [11]–[13]. The main objective of the antagonistic k-plex is to identify two opposing opinion groups in a social network, where users within each group tend to agree with one another and disagree with users from the other group. By computing maximal antagonistic k-plex, we can efficiently discover user groups with contrasting perspectives. These groups are typically tightly interconnected and hold significant influence within the network.

- Protein complex detection. To reconstruct the signaling pathway from the PPI network, we need the signatures of the PPIs, which represent whether the interaction has a positive or negative effect. A signed graph can represent activationinhibition relationships among proteins in the PPI network [14], [15]. Protein complexes can be defined as a group of proteins in which there is a dense population of positively interacting (i.e., activating) proteins, and a dense population of negatively interacting (i.e., inhibiting) [8], [16]. Therefore, identifying antagonistic k-plex in signed PPI networks can serve as a valuable method for detecting protein complexes. - Synonyms and antonymic phrases discovery. Signed graphs provide a natural way to represent synonym and antonym relationships between words [17]. Utilizing maximal antagonistic k-plex, we can discover synonym groups that are antonymous with each other, such as {sonant, voiced, loud, hard} and {surd, soft, voiceless, unvoiced}. These discovered clusters can be further utilized in applications such as automatic question generation [18] and semantic expansion [19].

**Contributions.** We make the following contributions:

- A new k-plex model for signed graphs. We formalize the antagonistic plex model in signed networks based on the structural balance theory. As far as we know, the paper is the first work considering the maximal antagonistic k-plex in signed networks. We prove the NP-Hardness of the problem. - A new framework tailored for maximal antagonistic plex enumeration. In conjunction with the definition of antagonistic k-plex, we introduce antagonistic k-plex expansion conditions. We employ set enumeration techniques to correctly discover of all eligible maximal antagonistic k-plexes, facilitating a comprehensive exploration of the solution space.

- Novel optimization strategies to improve the enumeration performance. We present early termination conditions to speed up set enumeration. Using the pivot technique, we reduce the search branch during enumeration. Moreover, the color bound in the signed graph allows premature termination of unpromising searches, improving the efficiency of identifying maximal antagonistic k-plex in signed networks.
- Preprocessing before set enumeration. In the preprocessing phase, we employ degree pruning in the signed graph, followed by converting it into a dichromatic graph and applying the plex pruning rules. We demonstrate that plexes infeasible in the dichromatic graph cannot form antagonistic k-plex in the signed graph. Utilizing this insight, we propose a method to remove unqualified vertices from the dichromatic graph, thereby reducing the size of the signed graph before set enumeration.
- Extensive performance studies on real datasets. We perform comprehensive experiments to assess the performance of our

proposed algorithms on various real datasets. The results demonstrate that our optimized approach achieves significantly faster execution, nearly two orders of magnitude quicker than the basic algorithm.

**Outline.** Section 2 provides preliminaries including the definition of antagonistic k-plex model and problem statement. Section 3 introduces the basic algorithm. Section 4 shows several optimization techniques. Section 5 reports the results of experimental studies. Section 6 shows some related works. Section 7 concludes our paper.

### II. Preliminaries

In this paper, we focus on the undirected signed graph G=(V,E), where V is the set of vertices and E is the set of signed edges. Each edge in E is either positive or negative. We use  $E^+$  and  $E^-$  to denote the positive and negative edges, respectively, where  $E=E^+\cup E^-$ . There are no multi-edges in G. We denote the number of vertices and the number of edges by n and m, respectively, i.e., n=|V| and  $m=|E|=|E^+|+|E^-|$ . Let  $N_G^+(v)$  represent the positive neighbors of v, i.e.,  $N_G^+(v)=\{u|(v,u)\in E^+\}$ . Let  $N_G^-(v)$  represent the negative neighbors of v, i.e.,  $N_G^-(v)=\{u|(v,u)\in E^-\}$ . We use  $d_G^+=|N_G^+(v)|$  and  $d_G^-=|N_G^-(v)|$  to denote the positive degree and negative degree of v, respectively. We denote  $N_G(v)=N_G^+(v)\cup N_G^-(v)$  by  $N_G(v)$  and  $d_G(v)=d_G^+(v)+d_G^-(v)$  by  $d_G(v)$ . Let  $N_G^2(v)$  be the 2-hop neighbor of v, i.e.,  $N_G^2(v)=\{u|N_G(u)\cap N_G(v)\neq\emptyset, u\not\in N_G(v)\}$ .

We denote v's positive neighbors' positive neighbors as  $N_G^{++}(v)$ . Let  $w_1,w_2...w_n$  be the vertices in  $N_G^+(v)$ ,  $N_G^{++}(v) = \bigcup_1^n N_G^+(w_i)$ . In the same way, we define  $N_G^{+-}(v)$ ,  $N_G^{-+}(v)$ , and  $N_G^{--}(v)$ , which represent v's positive neighbors' negative neighbors, v's negative neighbors' positive neighbors, and v's negative neighbors' negative neighbors respectively. Given two vertices u,v, if  $(u,v) \in E^+$ , we think they are friends and belong to the same group. If  $(u,v) \in E^-$ , we think they are opponents and belong to the antagonistic groups. Further, the 2-hop neighbors of v also can be put into two groups, i.e., the candidate group of v's friends and the candidate group of v's opponents. We use  $N_G^{2+}(v)$  to denote v's 2-hop neighbors which may join the same group with v, i.e.,  $N_G^{2+}(v) = N_G^{++}(v) \cup N_G^{--}(v)$ . We use  $N_G^{2-}(v)$  to denote v's 2-hop neighbors which may join the different group with v, i.e.,  $N_G^{2-}(v) = N_G^{-+}(v) \cup N_G^{--}(v)$ . Note that a vertex can be contained in both  $N_G^{2+}(v)$  and  $N_G^{2-}(v)$ .

**Definition 1** (Antagonistic Graph [7]). A signed graph  $G = (V, E^+, E^-)$  is antagonistic if V can be split into two subsets  $C_L$  and  $C_R$ , s.t.,  $\forall (u, v) \in E^+ \to u, v \in C_L$  or  $u, v \in C_R$ ; and  $\forall (u, v) \in E^- \to u \in C_L, v \in C_R$  or  $u \in C_R, v \in C_L$ .

A k-plex is a subgraph in which every vertex connects to at least s-k vertices in the subgraph where s is the number of vertices in the subgraph [20]. It is clear that any subgraph of a k-plex is also a k-plex.

**Definition 2** (Maximal Antagonistic k-plex). Given a signed network  $G = (V, E^+, E^-)$ , a maximal antagonistic k-plex C is a maximal subgraph of G that satisfies: (1) C is k-plex; and (2) C is antagonistic.

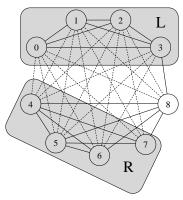


Fig. 2. Maximal antagonistic k-plex

To guarantee the cohesiveness of resulting k-plexes, several existing works [21], [22] only consider k-plexes with at least 2k-1 vertices. We also follow this setting.

**Lemma 1** (Bounded Diameter [20]). Given a k-plex C with the size of s, C is connected and the diameter of C is at most 2 if  $s \ge 2k - 1$ .

The bounded diameter guarantees that any two vertices in the k-plex are close and at least share a common neighbor. We allow users to control the size of resulting k-plexes by setting a size threshold of at least 2k-1 for  $C_L$  and  $C_R$ . The research problem is formally presented as follows.

**Problem statement.** Given a signed network G and two integers k and  $t \geq 2k-1$ , we aim to compute all maximal connected antagonistic k-plex C in G s.t.  $|C_L| \geq t$  and  $|C_R| \geq t$ .

For each resulting maximal antagonistic k-plex, we guarantee the cohesiveness of  $C_L$ ,  $C_R$ , and the whole subgraph. Note that the number of all vertices in each resulting subgraph is at least 4k-2, and the diameter is also bounded by 2.

**Example 1.** Figure 2 is a signed network. The solid/dashed lines denoted positive/negative edges. If we set t = 4 and k=2, there is a maximal antagonistic k-plex in this graph. This maximal antagonistic k-plex C can be divide into two parts,  $C_L = \{v_0, v_1, v_2, v_3\}$  and  $C_R = \{v_4, v_5, v_6, v_7\}$ , where vertices in  $C_L$  and  $C_R$  are marked with different markings.  $C_L$  is complete as any two vertices in  $C_L$  have an positive edge. It is clear that  $C_L$  is a positive k-plex. In  $C_R$ , only  $v_6$  and  $v_7$  are not adjacent. So  $C_R$  is still a positive k-plex. Consider the negative edges between  $C_L$  and  $C_R$ , only  $v_3$ and  $v_7$  do not have a negative edge. It meets the requirement of k-plex. Though  $v_8$  is adjacent to all the other vertices, it has positive edges with  $V_3$  and all the vertices in  $C_R$ . That means  $v_8$  cannot be added into the maximal antagonistic kplex.  $C \cup \{v_8\}$  is not qualified for an antagonistic community. Therefore, the plex C is maximal.

**Theorem 1** (Problem Hardness). The maximal antagonistic k-plex enumeration problem is NP-Hard.

*Proof.* It can be proved following the NP-Hardness of maximal k-plex enumeration problem [23]. According to definition 2, all the maximal antagonistic k-plex is k-plex, if we treat the positive and negative edges as the same. In other words, maximal antagonistic k-plex is a kind of more complex k-

plex. We can even list all of the k-plex first and test each of them to judge if they can be maximal antagonistic k-plex. Although in the research problem of this paper we are given a size constraint, finding all eligible maximal antagonistic k-plexes cannot be solved in non-deterministic polynomial-time. It is clear that maximal antagonistic k-plex is NP-hard.

### III. A BASIC ALGORITHM

We first propose a basic algorithm based on existing techniques for maximal k-plex enumeration in unsigned networks [24]–[26] and maximal balanced clique enumeration [7] in signed networks. Our framework maintains an antagonistic kplex C by two vertex sets  $C = \{C_L, C_R\}$  based on Definition 2. Let  $P_L$  be the set of candidate vertices that can be added into  $C_L$ , and  $P_R$  be the set of candidate vertices that can be added into  $C_R$ . In each step, we expand C by adding vertices from  $P_L$  and  $P_R$  into  $C_L$  and  $C_R$ , respectively. When a new vertex is added into C, we should update  $P_L$  and  $P_R$ . Until no more vertices can join C, we can stop and output  $C_L$  and  $C_R$  as a maximal antagonistic k-plex. Moreover, we use  $Q_L$ and  $Q_R$  to record candidate vertices that have been processed to avoid outputting duplicate maximal antagonistic k-plex. If  $Q_L$  or  $Q_R$  is not empty, the corresponding antagonistic k-plex  $C = C_L \cup C_R$  is contained in an earlier result.

Our basic algorithm for maximal antagonistic k-plex enumeration is presented in Algorithm 1. We process vertices  $v_0, v_1, ..., v_n$  (Line 2) in the ascending order of  $min(d_G^+, d_G^-)$ . We say a vertex u ranks higher than v if u is in front of v in the order. For each vertex  $v_i$ , we enumerate all maximal antagonistic k-plexes containing  $v_i$  (Line 2–8).  $C_L$  and  $C_R$  are initialized by  $v_i$  and  $\emptyset$ , respectively (Line 3). We initialize  $P_L$  and  $P_R$  with vertices ranking lower than  $v_i$  in the candidate set. We initialize  $Q_L$  and  $Q_R$  with vertices ranks higher than  $v_i$  in the candidate set. After initializing these six sets, we invoke procedure BAPEUTIL to enumerate all maximal antagonistic k-plexes containing  $v_i$  (Line 8).

**Expanding the subgraph.** Given a k-plex C and a vertex v,  $S \cup \{v\}$  is also a k-plex if and only if (1) v is adjacent to all the vertices in S which satisfies  $d_S(v) = |S| - k$ ; and (2)  $|N_G(v) \cap S| \ge |S| + 1 - k$ . We extend the conditions to the context of signed networks.

**Lemma 2.** Given an antagonistic k-plex  $C = \{C_L, C_R\}$  and a vertex v,  $C' = C \cup \{v\}$  is also an antagonistic k-plex if v satisfies ①③ or ②③ as follows.

- ②  $\forall u \in \{u \in C_R | d_C(u) = |C| k\} \rightarrow (u, v) \in E^+;$   $\forall u \in C_R \rightarrow (u, v) \notin E^-;$   $\forall u \in \{u \in C_L | d_C(u) = |C| - k\} \rightarrow (u, v) \in E^-;$  $\forall u \in C_L \rightarrow (u, v) \notin E^+.$
- ③  $|N_G(v) \cap C| \ge |C| + 1 k$ . Specifically, if v satisfies ①③,  $C' = \{C_L \cup \{v\}, C_R\}$ . If v satisfies ②③,  $C' = \{C_L, C_R \cup \{v\}\}$ .

*Proof.* According to Definition 2, the whole subgraph is a k-plex. So, the new vertex v must satisfy  $\Im v$  must also simul-

### **Algorithm 1** BAPE $(G = (V, E^+, E^-), k, t)$ **Input:** a signed graph G,k,tOutput: All maximal antagonistic k-plex 1: $Flag \leftarrow true$ 2: **for** $v_i \in \{v_0, v_1, ..., v_{n-1}\}$ **do** $\begin{aligned} & v_i \in \{v_0, v_1, ..., v_{n-1}\} \text{ at } \\ & C_L \leftarrow \{v_i\}, C_R \leftarrow \emptyset \\ & P_L \leftarrow (N_G^+(v_i) \cup N_G^2^+(v_i)) \cap \{v_{i+1}, ..., v_{n-1}\} \\ & P_R \leftarrow (N_G^-(v_i) \cup N_G^2^-(v_i)) \cap \{v_{i+1}, ..., v_{n-1}\} \\ & Q_L \leftarrow (N_G^+(v_i) \cup N_G^2^+(v_i)) \cap \{v_0, ..., v_{i-1}\} \\ & Q_R \leftarrow (N_G^-(v_i) \cup N_G^2^-(v_i)) \cap \{v_0, ..., v_{i-1}\} \\ & \text{BAPEUTIL}(C_L, C_R, P_L, P_R, Q_L, Q_R, k, t) \end{aligned}$ 3: 4: 5: 6: 7: 8: 9: procedure $\mathtt{BAPEUTIL}(C_L,C_R,P_L,P_R,Q_L,Q_R,k,t)$ $P_L \leftarrow \texttt{update}(P_L, C_L, C_R, k)$ $Q_L \leftarrow \texttt{update}(Q_L, C_L, C_R, k)$ 10: 11: $P_R \leftarrow \text{update}(P_R, C_L, C_R, k)$ 12: 13: $Q_R \leftarrow \text{update}(Q_R, C_L, C_R, k)$ if $P_L=\emptyset$ and $P_R=\emptyset$ and $Q_L=\emptyset$ and $Q_R=\emptyset$ then 14: 15: if $|C_L| \ge t$ and $|C_R| \ge t$ then return $C = \{C_L, C_R\}$ 16: $Flag \leftarrow !Flag$ 17: if Flag then 18: 19: for $v \in P_L$ do 20: $P_L \leftarrow P_L \setminus \{v\}$ 21: $\mathtt{BAPEUTIL}(\hat{C}_L \cup \left\{v\right\}, C_R, P_L, P_R, Q_L, Q_R)$ 22: $Q_L \leftarrow Q_L \cup \{v\}$ 23: for $v \in P_R$ do 24: $P_R \leftarrow P_R \setminus \{v\}$ 25: $\mathtt{BAPEUTIL}(C_L, C_R \cup \{v\}, P_L, P_R, Q_L, Q_R)$ 26: $Q_R \leftarrow Q_R \cup \{v\}$ 27: else 28: Line 23-26; Line 19-22

taneously connect all vertices u in C with  $d_v(u) = |C| - k$ . At the same time, v must be attributed to either  $C_L$  or  $C_R$ . Then, the edges between v and the vertex u with  $d_v(u) = |C| - k$  must also conform to the antagonistic principle of positive edges between vertices in the same group and negative edges between vertices in different groups. It is also possible that v is connected to other vertices in C with  $d_v(u) \neq |C| - k$ , and then these edges must likewise not violate the antagonistic principle. Therefore, at least one of v0 and v0 is satisfied. v1

Based on Lemma 2, we can locate a set of candidate vertices that can be added to the current antagonistic k-plex. When  $C_L$ or  $C_R$  expands, we need to refine the candidate set. Algorithm 2 presents the update procedure given the new  $C_L$  and  $C_R$ . The input set X can be  $P_L$ ,  $P_R$ ,  $Q_L$  or  $Q_R$ . Supported by the update (Algorithm 2) procedure, BAPEUtil performs the maximal antagonistic k-plex enumeration based on the given six sets. If  $P_L$ ,  $P_R$ ,  $Q_L$  and  $Q_R$  are empty (Line 14), the current antagonistic plex  $C = \{C_L, C_R\}$  cannot be enlarged and is a maximal antagonistic k-plex. BAPEUtil further checks whether  $C_L$  and  $C_R$  satisfy the size constraint. If so, it outputs the current antagonistic plex (Line 15–16). Otherwise, BAPEUtil adds a vertex v from  $P_L$  to  $C_L$  and recursively invokes itself for further expansion (Line 21). When v is processed, v is removed from  $P_L$  and added in  $Q_L$  (Line 22). Similar processing steps are applied on vertices in  $P_R$  (Line 23–26). In order to balance the size of  $C_L$  and  $C_R$  when searching, we first expand  $C_R$  in the next call by judging the Flag (Line 28).

**Theorem 2** (Correctness). Algorithm 1 finds all maximal antagonistic k-plexes correctly without redundancy. Proof. We show the correctness of Algorithm 1 from three aspects: (1) The antagonistic k-plex output in Algorithm 1 is

# **Algorithm 2** update $(X, C_L, C_R, k)$

```
1: Xnew = \emptyset
2: if X is P_L or Q_L then
3:
       for v in X do
4:
           if v meet requirement @ and @ then
5:
               Xnew \xleftarrow{\cdot} Xnew \cup \{v\}
6: if X is P_R or Q_R then
7:
       for v in X do
           if v meet requirement @ and @ then
8:
9.
               Xnew \leftarrow Xnew \cup \{v\}
10: Return Xnew
```

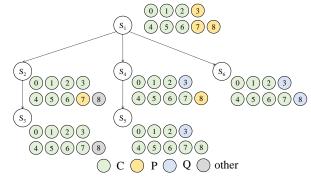


Fig. 3. Running example

maximal. Assume that an antagonistic k-plex C output in Line 16 is not maximal. The candidate set updated by Algorithm 2 would not be empty simultaneously. For a vertex v can be added into C and make  $C \cup \{v\}$  be a larger plex, if v is behind the current enumeration vertex u, v should be stored in  $P_L$  or  $P_R$ . C cannot be output in this round. Otherwise, v should be stored in  $Q_L$  or  $Q_R$ . It cannot satisfy the conditions in Line 14, and C would not be output, either. (2) Algorithm 1 will output all the qualified maximal antagonistic k-plex. In Line 2, Algorithm 1 visits each vertex  $v_i$ . Based on the recursive structure of BAPE, all the maximal antagonistic kplex containing  $v_i$  are explored. (3) Algorithm 1 will not output the same maximal antagonistic k-plex more than once. If the current enumeration vertex is  $v_i$  and the current plex has been output in the previous round, the current plex must contain a vertex  $v_i(j < i)$ . In the previous round,  $v_i$  would be added into  $Q_L$  or  $Q_L$ . Therefore, in the current round,  $Q_L$ or  $Q_L$  will not be empty, and the plex will not be output. Combining all the above three aspects, the correctness of Algorithm 1 is proved. 

**Example 2.** The enumeration procedure of BAPE can be illustrated as a search tree. Figure 3 shows part of the search tree when we conduct the BAPE (k = 2, t = 4) on G in Figure 2 through BAPE.  $S_1, S_2, \ldots$  represent different search states during the enumeration. At  $S_1$ , we assume that we have an antagonistic k-plex  $C = \{C_L = \{v_0, v_1, v_2\}, C_R = \{v_4, v_5, v_6\}\}$  at this state. First, we add  $v_3$  from  $P_L$  to  $C_L$  in  $S_2$ . Due to the negative edge between  $v_3$  and  $v_8$ , we should delete  $v_8$  from  $P_R$ . Then,  $v_7$  is added into  $C_R$ . At  $S_3$ , there are no vertices in P or Q. So  $C = \{C_L = \{v_0, v_1, v_2, v_3\}, C_R = \{v_4, v_5, v_6, v_7\}\}$  can be output as a maximal antagonistic k-plex. In  $S_4$ , we add  $v_7$  into  $C_R$  first. Now,  $v_3$  is in  $Q_L$ . Though  $v_8$  is added into  $C_R$  in  $S_5$ , the current plex does not qualify due to the size of  $C_L$  being too small. Similarly, no plex will be output in  $S_6$ .

We analyze the complexity of our basic approach and start from the time complexity of update as follows.

**Theorem 3.** The Algorithm update  $(X, C_L, C_R, k)$  runs in  $O(|C|^2 + |X|(|C_L| + |C_R|))$  time.

*Proof.* Algorithm 2 can be divided into two parts. First, according to the expansion conditions, we should find all the vertices u in  $C_L$  and  $C_R$  such that  $d_C(u) = |C| - k$ . This step can be finished within running time  $O(|C|^2)$ . Second, for each vertex v in X, we should test its adjacency to all vertices in C, and we can compute  $|N_G(v) \cap C|$  simultaneously. This step can be done in  $O(|X|(|C_L| + |C_R|))$ .

We consider the size of  $P_L$ ,  $Q_L$ ,  $P_R$  and  $Q_R$  which are the inputs of BAPEUTIL. The total size of the four sets is bounded by  $N_G^2(v_i)$ . Let  $\Delta$  be the maximum degree of  $v \in G$ . We have  $|N_G^2(v_i)| \leq \Delta^2$ . Based on Theorem 3, update runs in  $O(\Delta^4)$ . For every  $v_i$  in G, BAPE invokes BAPEUTIL once.

**Theorem 4.** The worst-case time complexity of Algorithm 1 is  $O(n\Delta^4 2^{\Delta^2})$ .

*Proof.* Consider the case where the input subgraph of the function BAPEUTIL is an antagonistic k-plex itself. Due to the hereditary property of k-plex, the input subgraph of BAPEUTIL is k-plex in each iteration. Therefore, the update process cannot remove vertices from P. In this case, BAPEUTIL will be called  $|P_{jL}| + |P_{jR}|$  times in j-th iteration,  $P_{jL}$  and  $P_{jR}$  are the symbol of  $P_{L}$  and  $P_{R}$  in j-th iteration. Due to the property of set enumeration, the total times is  $O(2^{|P_{L}|+|P_{R}|}) \leq O(2^{\Delta^{2}})$ . Combining with Theorem 3, the running time of the update can be assumed as  $O(\Delta^{4})$ . Hence, the whole running time of Algorithm 1 in the worst case is  $O(\Delta^{4}\sum_{i=0}^{n}2^{\Delta^{2}})\leq O(n\Delta^{4}2^{\Delta^{2}})$ .

It is worth mentioning that although we do not introduce pruning on degree in this section, degree-based VR pruning (see Section IV-B for details) is included in our experiments with the baseline algorithm(BAPE).

## IV. OPTIMIZATION

# A. Enumeration Optimization

We present several novel optimization techniques to improve the efficiency of the basic approach. We improve the key recursive procedure BAPEUTIL. The updated algorithm is called SAPEUTIL, and the pseudocode is presented in Algorithm 3.

**Pivoting-based pruning technique.** We utilize the pivot technique to prune the unnecessary branches in the search tree of Algorithm 1.

**Lemma 3** (k-plex pivoting [22]). In an undirected unsigned graph, let C be a k-plex, P is the candidate set, i.e.,  $P = \{v \notin K | K \cup \{v\} \text{ is a k-plex}\}$ , and u is a vertex in P. Any maximal k-plex containing C contains either u, a nonneighbor of u, or a neighbor v of u such that v and u have a common non-neighbor in K.

As shown in lemma 3, the existing pivot method only works for unsigned graphs. We extend it to the antagonistic k-plex in signed graphs.

```
Algorithm 3 SAPEUTIL(C_L, C_R, P_L, P_R, Q_L, Q_R, k, t)
```

```
1: P_L \leftarrow \text{update}(P_L, C_L, C_R, k)
 2: Q_L \leftarrow \text{update}(Q_L, C_L, C_R, k)
 3: P_R \leftarrow \text{update}(P_R, C_L, C_R, k)
4: Q_R \leftarrow \text{update}(Q_R, C_L, C_R, k)

5: if P_L = \emptyset and P_R = \emptyset and Q_L = \emptyset and Q_R = \emptyset then

6: if |C_L| \ge t and |C_R| \ge t then
                return C = \{C_L, C_R\}
 8: if |C_L| + |P_L| < t or |C_R| + |P_R| < t then return
9: Partition vertices of P_L, P_R, P_L \cup P_R by greedy coloring heuristic. They colornums are cd^L, cd^R, cd^A respectively.
10: if cd^L < t or cd^R < t or cd^A < 2t then continue
11: Flag \leftarrow !Flag
12: if Flag then
13:
           choose a pivot u from P_L \cup Q_L
           A_{L} = \left\{ c \in P_{L} \cap N_{G}(u) \mid C_{L} \setminus (N_{G}^{+}(u) \cup N_{G}^{+}(c)) = \emptyset \right\}
14:
           B_L = \left\{ c \in P_L \cap N_G(u) \mid C_R \setminus (N_G^-(u) \cup N_G^-(c)) = \emptyset \right\}
15:
           for v \in P_L \setminus N_G(u) \cap A_L \cap B_L do
16:
17:
                 P_L \leftarrow P_L \setminus \{v\}
                if cd_v^L < t or cd_v^A < 2t then continue
18:
19:
                \texttt{SAPEUTIL}(C_L \cup \left\{v\right\}, C_R, P_L, P_R, Q_L, Q_R)
20:
                Q_L \leftarrow Q_L \cup \{v\}
           A_R = \left\{ c \in P_R \cap N_G(u) \mid C_R \setminus (N_G^+(u) \cup N_G^+(c)) = \emptyset \right\}
21:
           B_R = \left\{ c \in P_R \cap N_G(u) \mid C_L \setminus (N_G^-(u) \cup N_G^-(c)) = \emptyset \right\}
22:
23:
           for v \in P_R \setminus N_G(u) \cap A_R \cap B_R do
24:
                P_R \leftarrow P_R \setminus \{v\}
                if cd_v^R < t or cd_v^A < 2t then continue
25:
26:
                \mathtt{SAPEUTIL}(C_L,C_R \cup \left\{v\right\},P_L,P_R,Q_L,Q_R)
27:
                 Q_R \leftarrow Q_R \cup \{v\}
28: else
           choose a pivot u from P_R \cup Q_R
29:
           Line 21-27; Line 14-20
```

**Lemma 4** (Antagonistic k-plex pivoting). In a signed graph, let  $C = \{C_L, C_R\}$  be a k-plex, candidate set be  $P = \{v \notin C | C \cup \{v\} \text{ is a } k\text{-plex}\}$ , and u be a vertex in P. Any maximal k-plex containing  $C_L$  contains either u, a nonneighbor of u, or a neighbor v of u such that v and u have a common non-neighbor in  $C_L$ . Any maximal k-plex containing  $C_R$  contains either u, a non-neighbor of u, or a neighbor v of u such that v and u have a common non-neighbor in  $C_R$ .

*Proof.* In the signed graph, if we want to select a vertex from the candidate set and put it into the current antagonistic plex, the vertex we choose should be able to form two plexes simultaneously. Specifically, if we want to move v from  $P_L$  to  $C_L$ , we must meet two conditions. First, considering only the positive side,  $C_L \cup \{v\}$  is a k-plex. Second, considering only v's negative edges and the positive edges in  $C_R$ ,  $C_R \cup \{v\}$  is also a k-plex.

To apply Lemma 4, we choose u as a pivot. In order to maximize the effectiveness of this cut, we adopt the philosophy of Chen et al [7], [7], and select a vertex u with maximum  $|N^{+(-)}(u)\cap P_L|+|N^{-(+)}(u)\cap P_R|$ . Then, we compute all neighbors of u in the candidate set  $P_L$  with no common nonneighbor with u in  $C_L$ . The result is denoted as  $A_L$ . We also compute all neighbors of u in  $P_L$  that have no common nonneighbor with u in  $C_R$ . The result is denoted as  $B_L$ . The pivoting technique replaces Line 19 in Algorithm 1 with the following operation:

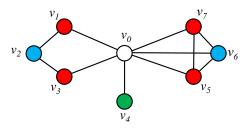


Fig. 4. Color bound example

Similarly, we compute all neighbors of u in the candidate set  $P_R$  with no common non-neighbor with u in  $C_R$ . The result is denoted as  $A_R$ . We also compute all neighbors of u in  $P_R$  that have no common non-neighbor with u in  $C_L$ . The result is denoted as  $B_R$ . We replace Line 23 in Algorithm 1 with the following operation:

For 
$$v \in P_R \setminus N_G(u) \cap A_R \cap B_R$$

By pivoting, we reduce the number of recursions and the search scope. In pivoting-based pruning, we must identify if an edge exists between a vertex in  $P_L$  (or  $P_R$ ) and a vertex in  $C_L$  (or  $C_R$ ). Therefore, the pruned candidate set can be computed in  $O((|P_L| + |P_R|)(|C_L| + |C_R|))$ .

**Early termination.** For an antagonistic k-plex  $C = \{C_L, C_R\}$ , if all vertices in  $P_L$  and  $P_R$  can be moved to C, the maximal possible size of  $C_L$  and  $C_R$  for the final maximal antagonistic k-plex are  $|C_L| + |P_L|$  and  $|C_R| + |P_R|$ , respectively. Therefore, we can terminate the search if  $|C_L| + |P_L| < k$  or  $|C_R| + |P_R| < k$ . We apply the rule in Line 8 of Algorithm 3, which can be done in constant time.

**Colorbound-based pruning technique.** We extend the color bound method for k-plex search in ordinary graphs and propose color-bound-based pruning on signed graphs.

**Lemma 5** (Color-bound [27]). Given a graph G = (V, E), if V can be partitioned into c disjoint independent sets  $I_1...I_c$ , then the upper bound of the size of maximum k-plex in G is  $\sum_{i=1}^{c} min\{|I_i|, k\}$ , a.k.a. color-bound.

According to the size limit in the problem definition, we guarantee three plexes satisfy the size constraint simultaneously in a signed graph. The three plexes are the plex in  $C_L \cup P_L$  only containing positive edges, the plex in  $C_R \cup P_R$  only containing positive edges and the whole plex containing all edges. We apply the lemma 5 for three sets to prune unnecessary search space.

**Definition 3** (Colornum in Signed Graph). In an intermediate state of the enumeration of maximal antagonistic k-plex,  $P_L$  can be partitioned into c disjoint independent sets  $pl_1,...pl_c$  (only containing positive edges),  $P_R$  can be partitioned into c disjoint independent sets  $pr_1,...pr_c$  (only containing positive edges), P can be can be partitioned into c disjoint independent sets  $a_1,...a_c$  (containing positive and negative edges). Colornums in signed graph can be computed as follows, when  $L = C_L \cup P_L$ ,  $R = C_R \cup P_R$  and  $A = C_L \cup P_L \cup C_R \cup P_R$ .

$$cd^{L} \leftarrow \sum_{j=1}^{c} \min(|pl_{j}|, k) + |C_{L}| \\ cd^{R} \leftarrow \sum_{j=1}^{c} \min(|pr_{j}|, k) + |C_{R}| \\ cd^{A} \leftarrow \sum_{j=1}^{c} \min(|a_{j}|, k) + |C_{L} \cup C_{R}|$$

```
Algorithm 4 VertexReduction(G = (V, E^+, E^-), k, t)
```

```
\begin{array}{ll} \text{1: while } \exists v \in V, d^+_G(v) < t-k \text{ or } d^-_G(v) < t-k+1 \text{ or } d_G(v) < 2t-k \\ \textbf{do} \\ \text{2: } & \textbf{for } u \in N^+_G(v) \textbf{ do} \\ \text{3: } & d^+_G(u) - = 1 \\ \text{4: } & \textbf{for } u \in N^-_G(v) \textbf{ do} \\ \text{5: } & d^-_G(u) - = 1 \\ \text{6: } & G \leftarrow G \setminus v \\ \text{7: Return } G \end{array}
```

**Lemma 6.** Colornums in signed graph are the upper bound of the sizes of maximum k-plex of L, R and A.

*Proof.* According to Definition 2, there are three k-plexes in an antagonistic k-plex. Two small k-plexes that consider only positive edges, and a large k-plex that considers both positive and negative edges. Then, in the signed graph, we can compute three colornum upper bounds for the three k-plexes. Any eligible antagonistic k-plex should satisfy all three upper bounds simultaneously.

**Example 3.** Figure 4 shows the example of color bound reduction. In a intermediate state of the enumeration,  $C_L = \{v_0\}$ ,  $P_L = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ . According to lemma 6, the vertices in  $P_L$  are divided into three independent sets and colored with different colors, i.e.,  $pl_1 = \{v_1, v_3, v_4, v_5\}$ ,  $pl_2 = \{v_2, v_6\}$  and  $pl_3 = \{v_7\}$ . Suppose k = 2, then the colornum of  $P_L$  is 5.

Algorithm Implementation. We use the rule of color bound in Algorithm 3 (Lines 9–10). After each update of the candidate set, we greedily color the vertices of all remaining candidate sets. Then, we calculate colornums of  $P_L$ ,  $P_R$  and A based on Lemma 6. We can skip the iteration if the upper bound of the maximum plex that these candidate sets produce is less than our requirement.

**Lemma 7** (Color-degree of One Vertex [27]). In an intermediate state of the enumeration with a growing k-plex C, a candidate set P, assume  $I_1...I_c$  is a coloring of P. For  $u \in P$ , the size of k-plex S that  $u \in S$  and  $C \subseteq S$  is bounded by  $\sum_{j=1,u\notin I_j}^c \min(|I_j| \cap N_G(u), k) + (k - |(C \setminus N_G(u)|) + |C|$ .

Similarly, if we want to add a vertex u into  $C_L$  or  $C_R$ , we can test the color-degree of u first.

**Theorem 5** (Color-degree in Signed Graph). In an intermediate state of the enumeration with two growing k-plex  $C_L$  and  $C_R$ , two candidate sets  $P_L$  and  $P_R$ , assume  $pl_j$  is a coloring of  $P_L$ , assume  $pr_j$  is a coloring of  $P_R$ , assume  $a_j$  is a coloring of  $A_L$ . For  $v \in P_L(P_R)$ , the size of k-plex S that  $v \in S$  and  $C \subseteq S$  is bounded by following:

$$cd_v^L \leftarrow F\left(pl_j, N_G^+(v)\right) + (k - |C_L \setminus N_v^+(v)|) + |C_L|$$

$$cd_v^R \leftarrow F\left(pr_j, N_G^+(v)\right) + (k - |C_R \setminus N_v^+(v)|) + |C_R|$$

$$cd_v^A \leftarrow F\left(a_j, N_G(v)\right) + (k - |(C_L \cup C_R) \setminus N_G(v)|) + |C|$$
In all the above expression, we set  $F\left(X, Y\right) = \sum_{j=1, u \notin X}^c \min(|X| \cap Y, k)$ .

*Proof.* Similarly to Lemma 6, we can similarly compute three corresponding color-degrees on three k-plexes for each candidate vertex v. Any eligible v should satisfy the constraints on all three of its color-degrees simultaneously.

**Example 4.** In Figure 4, suppose k = 2 and try to add  $v_7$  into  $C_L$ . We can see  $min(|pl_1| \cap N_G^+(v_7), k) = 1$ ,  $min(|pl_2| \cap N_G^+(v_7), k) = 1$ ,  $(k - |(C_L \cup C_R) \setminus N_G(v)|) = 0$ , so  $cd_{v_7}^L = 5$ . If t > 5,  $v_7$  can be skipped.

We use this rule in Algorithm 3 (Lines 18 and 25). When we finish updateing, we test if the candidate set can build a qualified plex by Lemma 6. This step can be done in  $O(\Delta|P|)$ . Then, every time we try to add a vertex v int  $C_L(C_R)$ , we can calculate color-degree of v based on lemma 5. If the upper bound of the maximum plex that v can build in this iteration is less than our requirement, we can skip v. For all vertices in P, we need to iterate over its neighbor vertices. Therefore, the time is also  $O(\Delta|P|)$ .

# B. Preprocessing Optimization

In preprocessing optimization, we aim to remove worthless vertices and edges not contained in any maximal antagonistic k-plex according to the constraints.

**Vertex Reduction (VR).** We first consider the neighbors of each vertex. According to the definition 2, for each vertex v in a maximal antagonistic k-plex C, the number of v's neighbors is at least |C| - k. Given that |C| is at least 2t, we have  $d_G(v) \ge 2t - k$ .

 $C_L$  and  $C_R$  also should be k-plex. The edges between vertices in  $C_L$   $(C_R)$  are positive, which implies the positive degree is at least t-k, i.e.,  $d_G^+(v) \geq t-k$ . Further, we consider the negative degree. The negative edge only exists between vertices from  $C_L$  and  $C_R$ , respectively. Each vertex v in  $C_L$  can build a plex with  $C_R$  if we only consider the negative edges from v and the positive edge among  $C_R$ . Similarly, each vertex v in  $C_R$  can build a plex with  $C_L$  if we only consider the negative edges from v and the positive edge among  $C_L$ . That means the negative degree should be at least t-k+1, i.e.,  $d_G^-(v) \geq t-k+1$ .

Then, we use the three constraints to reduce the graph. VertexReduction is shown as Algorithm 4. First, we delete those vertices whose degree does not meet these requirements. Then, we can iterate over the neighbors of the deleted vertex and reduce their corresponding degrees by one. By doing so, some new vertices that do not satisfy the condition will appear. We recursively delete the vertices that do not satisfy the conditions until all the remaining vertices satisfy the conditions.

In Algorithm 4, a queue is used to store vertices that should be removed. Since each vertex is pushed in and popped from the queue at most once, the total processing time is O(n). After that, if a vertex is removed, we need to update the degrees for its neighbors. The total time cost is O(m). Therefore, the time complexity of Algorithm 4 is O(n+m).

**Dichromatic Reduction (DR).** VR can cut the size of the graph, but it is not enough. We also propose a new reduction for each specific vertex.

**Lemma 8** (Pruning Rule in Unsigned Graph [28]). In the undirected unsigned graph G = (V, E), if K is a k-plex and  $|K| \ge q$   $(q \ge 2k - 2)$ , any other vertex u which satisfies any of the following conditions is not in the K.

# **Algorithm 5** DichromaticOnehop(nv, k, t)

```
1: q is the Dichromatic-network of N_G(nv)
2: for u \in N_G^+(nv) do
         L_1 = L_1 \cup \{v\}
4: for u \in N_G^-(nv) do 5: R_1 = R_1 \cup \{v\}
6: while \exists v \in g, d_g^+(v) < t - 2k or d_g^+(v) + d_g^-(v) < 2t - 2k do
         for u \in N_g^+(v) do
7:
8:
             d_q^+(u) - = 1
9:
         for u \in N_g^-(v) do
10:
             d_q^-(u) - = 1
11:
         if v \in L_1 then
12:
             L_1 \leftarrow L_1 \setminus \{v\}
13:
             R_1 \leftarrow R_1 \setminus \{v\}
14:
15:
         update g
16: Return DichromaticTwohop(nv, L_1, R_1, k, t)
```

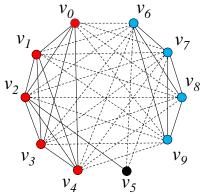


Fig. 5. DichromaticOnehop example

```
① u \in N_G(v) and |N_G(u) \cap N_G(v)| < q - 2k
② u \in N_G^2(v) and |N_G(u) \cap N_G(v)| < q - 2k + 2
```

We apply the pruning rule in the Dichromatic-network [8]. For a vertex v, the ego network consists of v and all its neighbor vertices, as well as all the edges between these vertices. For a vertex v, its Dichromatic-network is its Egonetwork after removing all conflicting edges. The conflicting edges are negative edges between vertices of  $N_G^+(u)$ , negative edges between vertices of  $N_G^-(u)$  and positive edges between a vertex of  $N_G^-(u)$  and a vertex of  $N_G^-(u)$ .

We use Dichromatic-network instead of ego-network for pruning. Those conflicting edges are impossible to add into an antagonistic k-plex. If we use ego-network, it would be difficult to judge whether a vertex should be added to  $C_L$ or  $C_R$ . For example, if v is the two-hop neighbor of u,  $\exists w \in N_G^+(v), u \in N_G^+(w) \text{ and } \exists x \in N_G^+(v), u \in N_G^-(x),$ it is clear that w is positive neighbor of u and they should be in the same group. x is negative neighbor of u, and they should be in different groups. However, it is hard to say which group v should be assigned. On the one hand, v is a positive neighbor of w, so they should be in the same group. u and w are in the same group, so u and v should be in the same group. On the other hand, v is a positive neighbor of x, so they should be in the same group. u and x are in different groups, so u and v should be in different groups. In the ego-network, analyzing from different directions will lead to opposite conclusions about the relationship between u and v. However, we can judge the vertex easily in Dichromaticnetwork. This is because there are no conflicting edges and

no misunderstandings.

**Lemma 9.** Finding an antagonistic k-plex in the Dichromaticnetwork is a necessary but insufficient condition for finding it on the original graph.

*Proof.* If we can find an antagonistic k-plex in Dichromatic-network, it does not mean that we can find this plex in the original graph due to conflicting edges. A conflicting edge can make a whole plex in Dichromatic-network unqualified. Fortunately, we find that if a vertices group cannot build a plex in Dichromatic-network, it cannot build a plex in ego-network. We can use this property to delete vertices on the Dichromatic-network and map them to the original graph.

Next, we will show how to prune in the Dichromaticnetwork. First, we consider the one-hop neighbor. We have the following lemma:

**Theorem 6** (Pruning Rule for One-hop). In the undirected signed graph G = (V, E), if C is a maximal antagonistic k-plex, for a given  $v \in C$ , any other vertex u which satisfies either of the following conditions is not in the C.

```
① u \in N_G^+(v) and |N_G^+(u) \cap N_G^+(v)| < t - 2k
② u \in N_G^-(v) and |N_G^+(u) \cap N_G^-(v)| < t - 2k
③ u \in N_G(v) and |N_G(u) \cap N_G(v)| < 2t - 2k
```

*Proof.* According to Lemma 8, each one-hop neighbor v of a given u should meet the requirements of the two small positive plexes and the whole antagonistic plex. For the small positive plex  $C_s$  that v belongs to, the number of positive neighbors of v in  $C_s$  is at least  $|C_s|-2k$ . For the whole antagonistic plex  $C_w$  that v belongs to, the number of neighbors of v in  $C_w$  is at least  $|C_w|-2k$ . Finally, according to Definition 2,  $C_s$  is at least t, t is at least t.

Algorithm Implementation. Reduction on one-hop neighbors is shown as Algorithm 5. First, we save all the positive and negative neighbors of nv in L and R, respectively (Lines 2–5). Then, we delete all unqualified vertices in g. However, when a vertex is deleted, it will affect the neighbor numbers of its neighbors. Some of its neighbors may become unqualified after deleting some vertices. At that time, we can add these vertices to the delete queue. Utill the delete queue is empty, we can stop, and all the remaining vertices in L and R are qualified one-hop neighbors.

**Example 5.** Figure 5 shows the example of Dichromatic-onehop reduction. We set k=2, t=4 and set  $v_0$  as the current nv in Algorithm 5. The other vertices in the graph are divided into two groups according to their adjacency to vertex  $v_0$ . The positive neighbours of  $v_0$  are  $v_1, v_2, v_3, v_4, v_5$ . Negative neighbours are  $v_6, v_7, v_8, v_9$ . However, the  $|N_G^+(v_5) \cap N_G^+(v_0)| = 3 < 2t - 2k = 4$ .  $v_5$  do not satisfy  $\circledast$  in lemma 6, so  $v_5$  is removed from  $L_1$ . All the remaining vertices form the Dichromatic-onehop neighbors of  $v_0$ .

This process is similar to Algorithm 4. It can be seen as VR in the Dichromatic-network. However, Dichromatic-network is smaller than G. There are at most  $\Delta$  vertices and  $\Delta^2$  edges in Dichromatic-network. So the time complexity of Algorithm 5 is  $O(\Delta + \Delta^2)$ , when  $\Delta$  is the maximum degree of  $v \in G$ .

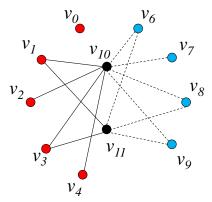


Fig. 6. DichromaticTwohop example

**Algorithm 6** DichromaticTwohop $(nv, L_1, R_1, k, t)$ 

```
1: Ln = \emptyset, Rn = \emptyset
 2: for u \in L_1 do
           for v \in N_G^+(u) do L_2 = L_2 \cup \{v\}
           for v \in N_G^-(u) do R_2 = R_2 \cup \{v\}
 5:
 6:
 7: for u \in R_1 do
 8:
           for v \in N_C^+(u) do
 9:
                 R_2 = R_2 \cup \{v\}
10:
            \begin{array}{c} \text{for } v \in N_G^-\left(u\right) \text{ do} \\ L_2 = L_2 \cup \{v\} \end{array} 
11:
12: for v \in L_2 \setminus N_G(nv) do 13: a = |N_G^+(u) \cap L_1|
           b=|N_G^{\square}\left(u\right)\cap R_1| if a\geq t-2*k+2 and a+b\geq 2*t-2*k+2 then
14:
15:
16:
                 Ln = Ln \cup \{v\}
17: for v \in R_2 \setminus N_G(nv) do
18:
           a = |N_G^+(u) \cap R_1|
           b=|N_G^-(u)\cap L_1| if a\geq t-2*k+2 and a+b\geq 2*t-2*k+2 then
19:
20:
                 Rn=Rn\cup\{v\}
21:
22: Return Ln = Ln \cup L_1, Rn = Rn \cup R_1
```

**Theorem 7** (Pruning Rule for Two-hop). In the undirected signed graph G = (V, E), if C is a maximal antagonistic k-plex, all qualified one-hop neighbors are in  $L_1 \cup R_1$ , w is the neighbor of a vertex in  $L_1 \cup R_1$ . For all the following cases, w is not in C, if a < t - 2k + 2 or a + b < 2t - 2k + 2.

①  $v \in L_1$ ,  $w \in N_G^+(v)$ . let a be the number of w's positive neighbors in  $L_1$ , b be the number of w's negative neighbors in  $R_1$ .

②  $v \in L_1$ ,  $w \in N_G^-(v)$ . let a be the number of w's positive neighbors in  $R_1$ , b be the number of w's negative neighbors in  $L_1$ .

③  $v \in R_1$ ,  $w \in N_G^+(v)$ . let a be the number of w's positive neighbors in  $R_1$ , b be the number of w's negative neighbors in  $L_1$ .

 $\textcircled{\$}v \in R_1$ ,  $w \in N_G^-(v)$ . let a be the number of w's positive neighbors in  $L_1$ , b be the number of w's negative neighbors in  $R_1$ .

*Proof.* First, we have identified all Dichromatic-onehop neighbors. Two-hop neighbors are neighbors of one-hop neighbors. After Algorithm 5 is completed, we have  $L_1$  as the qualified one-hop positive neighbors and  $R_1$  as the qualified one-hop negative neighbors. We can divide Dichromatic-twohop neighbors into four parts according to the adjacency with the Dichromatic-onehop neighbors. They are positive neighbors of vertices in  $L_1$ , negative neighbors of vertices in  $L_1$ , positive

# **Algorithm 7** SAPE $(G = (V, E^+, E^-), k, t)$

neighbors of vertices in  $R_1$ , and negative neighbors of vertices in  $R_1$ . According to Lemma 8, each Dichromatic-twohop neighbor v of a given u should also meet the requirements of two small positive plexes and the whole antagonistic plex. For the small positive plex  $C_s$  that v belongs to, v's positive neighbors in  $C_s$  is at least  $|C_s|-2k+2$ . For whole antagonistic plex  $C_w$  that v belongs to, v's neighbors in  $C_w$  is at least  $|C_w|-2k+2$ .

Algorithm Implementation. Reduction on two-hop neighbors is shown as Algorithm 6. We use Ln and Rn to store the final candidate sets of the given vertex nv. We put the positive neighbors of vertices in  $L_1$  and the negative neighbors of vertices in  $R_1$  into  $L_2$  as candidate sets for 2-hop vertices in  $L_n$ . Similarly, we put the positive neighbors of the vertices of  $R_1$  and the negative neighbors of the vertices of  $L_1$  into  $R_2$  as the candidate set of 2-hop vertices in  $R_n$  (Line 2–11). Then, we traverse  $L_2$  and  $R_2$  respectively and put the vertices meeting the above conditions into  $L_n$  and  $R_n$  respectively (Line 12–21). Then, we union the two-hop candidate Ln and Rn with one-hop candidate  $L_1$  and  $R_1$ , respectively (Line 22). Finally, we get Ln and Rn as the candidate vertices sets for the given vertex nv.

**Example 6.** Figure 6 shows an example of Dichromatic-twohop reduction. We set k=2, t=4 and set  $v_0$  as the current nv in Algorithm 6.  $L_1$  and  $R_1$  are the Dichromatic-onehop neighbor sets of nv. This figure omits the lines among  $v_0$  and Dichromatic-onehop neighbor of  $v_0$ .  $v_{10}$  is the two-hop neighbor of  $v_0$ . It has four positive neighbors in  $L_1$  and negative neighbors in  $R_1$ , which satisfy the requirements of Dichromatic-twohop. However,  $|N_G^+(v_{11}) \cap L_1| + |N_G^-(v_{11}) \cap R_1| = 5 < 2t - 2k + 2 = 6$ . Therefore,  $v_{11}$  is not Dichromatic-twohop neighbor of  $v_0$ .

In DR, we need to iterate over all the neighbors of the two-hop neighbors of nv. So the time complexity of Algorithm 6 is  $O(\Delta^3)$ . As the DR is called for each vertex in G, the total time complexity of DR is  $O(\Delta^3 n)$ .

Algorithm Implementation Summary. We summarize the preprocessing algorithm and show it as Algorithm 7. In the first step, we use VR (Line 1). Then, for each vertex we use Dichromatic Reduction to reduce the number of the vertices in the candidate set (Line 5). After that,  $P_L$ ,  $P_R$ ,  $Q_L$  and  $Q_R$  is computed (Line 6–9) and SAPEUTIL is called (Line 10).

It is worth noting that although a variety of optimization methods are incorporated in Algorithm 7 compared to Algorithm 1, Algorithm 7 still uses the structure of set enumeration

TABLE I DATASETS

Dataset	V	$ E^+ $	$ E^- $	$max(d_G^+)$	$max(d_G^-)$
Slashdot	77,357	396,378	120,197	2,507	598
Epinions	131,828	717,667	123,705	3,334	1,590
Super	567,301	82,547	632,023	2,598	11,696
WiKi	1,140,149	450,467	2,337,500	17,092	124,859

and its worst time complexity does not change. However, it can significantly reduce the number of iterations of the set enumeration. Algorithm 7 will be more efficient than Algorithm 1 in actual computation.

### V. EXPERIMENTS

In this section, we present our experimental results. All the experiments are performed on a machine with Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz and 96GB RAM and Ubuntu system. All algorithms are implemented in C++, using g++ complier with -O3. The time cost is measured as the amount of wall-clock time elapsed during the program's execution. If an algorithm cannot finish in 12 hours, we denote the processing time as INF. We evaluate our algorithms in some real and synthetic signed networks.

**Algorithms.** We evaluate the performance of the following methods:

- Baseline. It is the basic solution shown in Section 3. The baseline execution is too slow and cannot be completed within 12 hours. We use BA instead of baseline.
- BAPE. Apply VR to baseline. For simplicity, we denote it by BA in the following figures.
- SANC. It is the algorithm with all the optimization but color-bound pruning. We denote it by NC in the following figures.
- SAPE. It is the algorithm with the enumeration optimization shown in Section 4. We denote it by SA in the following figures.

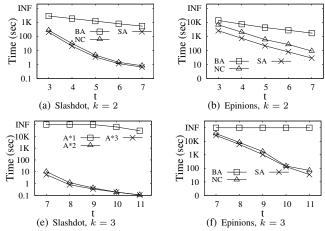
Note that the preprocessing optimization strategies VR can also be used in BAPE. Thus, we apply them to all three algorithms for fairness.

**Datasets.** We use four datasets downloaded from http://snap.stanford.edu to evaluate our algorithms. Slashdot and Epinions are real-world signed networks. Super and WiKi are not signed graphs, but we use the method in [29] to transform them into signed networks. The details of each dataset are shown in Table I.

# A. Efficiency when varying k and t

In this experiment, we evaluate the efficiency of three algorithms when varying t and k. The results are shown in Figure 7.

For each data set, we test five different t-values corresponding to k=2 and k=3. As shown in Figure 7. Among all the test cases, BAPE is the least efficient of the three algorithms. The other two algorithms are considerably more efficient than BAPE. For example, when t=3 and k=3, BAPE cannot finish computing Slashdot in 12h, while BAPE and SANC can complete within minutes. This is mainly because BAPE computes too many hopeless search branches, while the other two algorithms use our proposed enumeration optimization. These optimizations help the algorithm estimate in advance



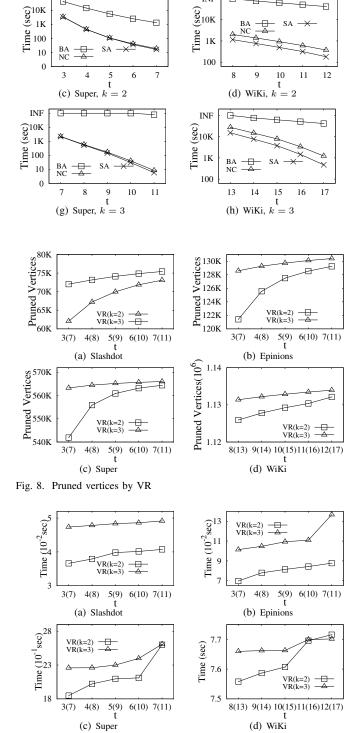
Running time of different algorithms varying t,k

whether the current search branch has any hope of finding maximal antagonistic k-plex in the future. If the current search branch is no longer possible to find the qualified maximal antagonistic k-plex, it is worthless to continue the computation in this time. It can be skipped without affecting the accuracy of the final result. Comparing SAPE and SANC, the only difference is the use of color-bound pruning. Calculating color bound increases the overhead of the algorithm, but it can further reduce the number of search branches and iterations at the same time. The efficiency of these two algorithms is relatively close, but as shown in the Figure 7, the overall efficiency of SAPE is still improved by using color bound. In addition, it can be seen from the figure that the efficiency of all algorithms increases either when the value of k becomes smaller or the value of t becomes larger. This is due to the increased pruning ability of preprocessing optimization at this time. Therefore, the number of vertices that can enter the set enumeration phase is reduced significantly. This reduces the search space of the set enumeration.

# B. Evaluation of Vertex Reduction

In this experiment, we evaluate the effectiveness and efficiency of the VR strategy. Results are shown in Figure 8 and 9. In these figures, given the dataset and value of k, the value of t corresponds to its setting in Figure 7. For example, the corresponding t of Slashdot k = 2 are the value in front of the abscissa axis  $\{3, 4, 5, 6, 7\}$ , and the corresponding t of Slashdot k = 3 are the value behind the abscissa axis  $\{7, 8, 9, 10, 11\}$ . The same is true for other datasets. All subsequent experimental graphs also obey this setting. Figure 8 shows the ability of VR strategy to delete vertices. For example, more than 90 percent of vertices can be pruned in Slashdot and Epinions. As the values of k and tincrease, more and more vertices in the process of VR can be removed.

Figure 9 shows the running time of VR. VR can remove millions of hopeless vertices in a very short period of time. As the value of k decreases and the value of t increases, more vertices need to be removed, and the running time of VR increases slightly, but the change is minimal.



INF

1K

§ 10K

Fig. 9. Running time of VR

INF

# C. Evaluation of Dichromatic Reduction

In this experiment, we evaluate the effectiveness and efficiency of the DR strategies. Figure 10 shows the number of processed vertices after using the DR strategy. After using the DR strategy in Algorithm 7, the sizes of the four sets  $P_L$ ,  $P_R$ ,  $Q_L$  and  $Q_R$  corresponding to each vertex are reduced. The candidate sets of many vertices no longer satisfy the expansion condition. Therefore, these vertices can be skipped during set enumeration. It can be seen from Figure 10 that

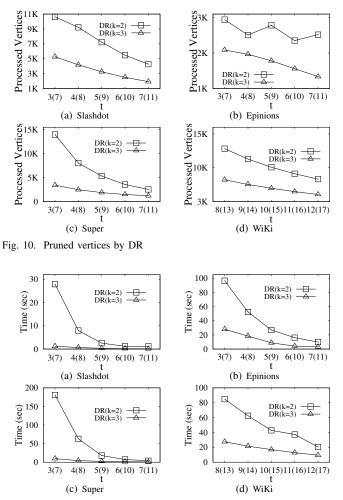


Fig. 11. Running time of DR

the DR strategy can further prune the remaining vertices after using the VR strategy on a large scale. With the increase of k and t, the number of vertices pruned by DR tends to decrease, which is mainly due to the enhanced pruning ability of VR strategy.

Figure 11 shows the time of DR. As k and t increase, the running time of DR also decreases. After VR, the number of vertices becomes smaller and each vertex has fewer two-hop neighbors. It reduces the search space of DR.

# D. Scalability testing

In this experiment, we test the scalability of BAPE and SAPE on the large dataset WiKi by varying their vertices and edges from 20% to 100%. We set t=8 and k=2 as default values. Figure 12 shows the results of the basic algorithm and our modified algorithms.

As shown in Figure 12, the running times of both algorithms increase as graph increases, but SAPE outperforms BAPE in all cases. e.g., when we sample 40% of the vertices, the running times of SAPE and BAPE are 0.9 and 372 seconds, respectively, while when 80% of the vertices are sampled, their runtimes are 232 and 36,458 seconds, respectively. This experiment shows that our modified algorithms have good scalability in practice.

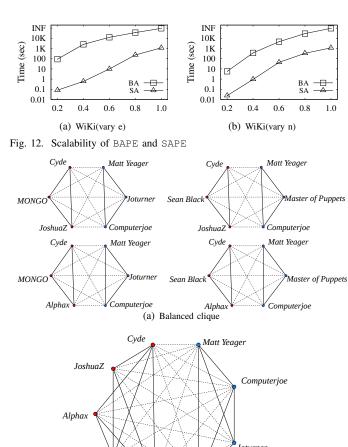


Fig. 13. Case study

# E. Memory Usage

Sean Black

MONGO

In this experiment, we compare the memory usage of the three algorithms and the size of the original dataset used. We set t=3, k=2 for Slashdot, Eponions and Super. Due to the large size of the WiKi, we set t=8, t=2. Figure ?? shows the results.

(b) Antagonistic k-plex

Master of Puppets

The memory usage of the three algorithms is very small, about only six times the size of the dataset. The memory usage of the three algorithms is close. It shows that our optimizations do not result in significant changes in memory usage.

### F. Case study on Wiki-RfA

In this experiment, we conduct a case study of the real dataset Wiki-RfA. Wiki-RfA records the mutual voting results of Wikipedia editors running for administrator, downloaded from https://snap.stanford.edu/. We build an undirected signed graph with wiki-RfA. Each editor can be seen as a vertex. If there is a voting relationship between two editors, then there is an edge between the two editors. If there is only support between two editors, the two editors are on a positive edge. Otherwise, they are on a negative edge. Our goal is to find cohesive antagonistic communities in the graph.

Figure 13 shows an example of search results for the balanced clique model [7] and our model. The red and blue vertices denote the two antagonistic groups in a cohesive antagonistic community. As shown in Figure 13a, [7] can only

find several 3-balanced cliques in this cohesive antagonistic community, which share many vertices. In fact, all the four 3-balanced cliques should be included in a big cohesive antagonistic community. Figure 13b shows the result of our model. For antagonistic k-plex, we set t=4 and k=2. Obviously the antagonistic k-plex gives better results. All the balanced cliques in Figure 13a can be included in the antagonistic k-plex shown in 13b.

This case verifies that the maximal antagonistic k-plex enumeration can be applied to find more generalized antagonistic communities than balanced cliques.

## VI. RELATED WORKS

**Signed network.** A lot of literature on signed graph analysis has appeared in recent years. An excellent survey on signed network analysis can be found in [30]. Among these theories, Structural balance theory is one of the fundamental theory [31] which is first introduced in [32] and extended to graph analysis in [5]. Our work is also based on structural balance theory. [6], [33]–[35] aim to find the antagonistic communities in a signed network. The authors in [33] hope to find a group of k subgraphs such that the edges within each subgraph are dense and cohesive and the edges crossing different subgraphs are dense and oppositive. These works aim to find several groups of dense subgraphs, but they do not propose a clear structural definition of their model.

There is also some research on searching cliques in signed graphs. an  $(\alpha, k)$ -clique model is proposed in [29]. They want to find a maximal clique C in a signed graph such that the vertices in C have no more than k negative neighbors and no less than  $\alpha k$  positive neighbors. However, this model only considers the positive and negative degree of vertices and ignores the model structure. To overcome this problem, [7] propose the definition of maximal balanced clique and a new enumeration algorithm for the maximal balanced clique enumeration problem. Given a signed network G, a balanced clique is complete and can be split into two subcliques. The edges in the same subclique are positive, and the edges between the different subcliques are negative. The author in [8] extended the balanced clique definition and proposed an algorithm for the maximum balanced clique enumeration problem. However, the clique model is too strict to mine real cohesive antagonistic communities on signed graphs fully. Therefore, in this paper, we choose k-plex, the relaxed model of clique.

k-plex model on unsigned networks. It is well-known that the problem of enumerating all maximal k-plexes is NP-hard [10]. Most of the current algorithms for enumerating maximal clique or relaxation models of clique are based on the core of the algorithm in [36], [37]. Many existing algorithms for listing maximal k-plexes, as in [38]–[40] also stem from the Bron-Kerbosch algorithm. Noted that when k=1, k-plex is a clique. In addition, many small size plexes are not applicable in the analysis of real data, so it is proposed in [9] to find extremely large plexes of size at least q. An efficient algorithm using clique and k-core to reduce the search space is also proposed in [9]. Then, [22] further modified the pivot method, which is

commonly used for clique enumeration, to further optimize the efficiency of the maximal k-plex algorithm. They also propose size pruning rules based on the number of common neighbors of two vertices [22]. However, the worst-case time complexity of the previous algorithms is  $O(n^22^n)$ . [28] proposed a new algorithm which can lists all maximal k-plexes with provably worst-case running time  $O(n^2\gamma^n)$  where  $\gamma<2$ . They design a pivot heuristic that always branches on the vertex of the minimum degree in the graph. The time complexity of [41]'s method is similar, but its practical effect is better. [24] lists all maximal k-plexes in  $O(n^2\gamma^D)$  time, where  $\gamma$  is also smaller than 2, and D is the degeneracy of the graph that is far less than the vertices number n. However, these algorithms are difficult to apply to signed graphs due to different structures.

## VII. CONCLUSIONS

In this paper, we study the maximal antagonistic k-plex problem in signed graphs. We propose a new model named antagonistic k-plex in a signed graph. A new maximal antagonistic k-plex enumeration framework is devised based on this model. Then, we design novel optimization strategies based on pivot and color bound to improve the efficiency of the enumeration algorithm. We also use early termination pruning and degree-based pruning to further reduce unnecessary iterative searches. We use the reduction in the dichromatic graph to reduce the search space in the preprocessing stage. Experimental results on real datasets demonstrate the efficiency, effectiveness, and scalability of our modified algorithms.

### REFERENCES

- [1] H. Chen, "Networks, crowds, and markets: Reasoning about a highly connected world [book review]," <u>IEEE Technol. Soc. Mag.</u>, vol. 32, no. 3, p. 10, 2013.
- [2] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: mining a social network with negative edges," in Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009, J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, Eds. ACM, 2009, pp. 741–750.
- [3] C. Giatsidis, B. Cautis, S. Maniu, D. M. Thilikos, and M. Vazirgiannis, "Quantifying trust dynamics in signed graphs, the s-cores approach," in Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014, M. J. Zaki, Z. Obradovic, P. Tan, A. Banerjee, C. Kamath, and S. Parthasarathy, Eds. SIAM, 2014, pp. 668–676.
- [4] L. Ou-Yang, D. Dai, and X. Zhang, "Detecting protein complexes from signed protein-protein interaction networks," <u>IEEE ACM Trans. Comput.</u> <u>Biol. Bioinform.</u>, vol. 12, no. 6, pp. 1333–1344, 2015.
- [5] F. Harary, "On the notion of balance of a signed graph." <u>Michigan Mathematical Journal</u>, vol. 2, no. 2, pp. 143–146, 1953.
- [6] M. Gao, E. Lim, D. Lo, and P. K. Prasetyo, "On detecting maximal quasi antagonistic communities in signed graphs," <u>Data Min. Knowl.</u> Discov., vol. 30, no. 1, pp. 99–146, 2016.
- [7] Z. Chen, L. Yuan, X. Lin, L. Qin, and J. Yang, "Efficient maximal balanced clique enumeration in signed networks," in <u>Proceedings of The</u> Web Conference 2020, 2020, pp. 339–349.
- [8] K. Yao, L. Chang, and L. Qin, "Computing maximum structural balanced cliques in signed graphs," in 38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022. IEEE, 2022, pp. 1004–1016.
- [9] A. Conte, D. Firmani, C. Mordente, M. Patrignani, and R. Torlone, "Fast enumeration of large k-plexes," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017. ACM, 2017, pp. 115-124.
- [10] B. Balasundaram, S. Butenko, and I. V. Hicks, "Clique relaxations in social network analysis: The maximum <u>k</u>-plex problem," <u>Oper. Res.</u>, vol. 59, no. 1, pp. 133–142, 2011.

- [11] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky, "Community interaction and conflict on the web," in <u>Proceedings of the 2018 world</u> wide web conference, 2018, pp. 933–943.
- [12] H. Xiao, B. Ordozgoiti, and A. Gionis, "Searching for polarization in signed graphs: a local spectral approach," in <u>Proceedings of The Web</u> <u>Conference 2020</u>, 2020, pp. 362–372.
- [13] F. Bonchi, E. Galimberti, A. Gionis, B. Ordozgoiti, and G. Ruffo, "Discovering polarized communities in signed networks," in <u>Proceedings</u> of the 28th acm international conference on information and knowledge management, 2019, pp. 961–970.
- [14] A. Suratanee, M. H. Schaefer, M. J. Betts, Z. Soons, H. Mannsperger, N. Harder, M. Oswald, M. Gipp, E. Ramminger, G. Marcus et al., "Characterizing protein interactions employing a genome-wide sirna cellular phenotyping screen," PLoS computational biology, vol. 10, no. 9, p. e1003814, 2014.
- [15] S. Yim, H. Yu, D. Jang, and D. Lee, "Annotating activation/inhibition relationships to protein-protein interactions using gene ontology relations," <u>BMC systems biology</u>, vol. 12, no. 1, pp. 111–122, 2018.
- [16] L. Ou-Yang, D.-Q. Dai, and X.-F. Zhang, "Detecting protein complexes from signed protein-protein interaction networks," <u>IEEE/ACM transactions on computational biology and bioinformatics</u>, vol. 12, no. 6, pp. 1333–1344, 2015.
- [17] G. A. Miller, "Wordnet: a lexical database for english," <u>Communications</u> of the ACM, vol. 38, no. 11, pp. 39–41, 1995.
- [18] V. Kumar, N. Joshi, A. Mukherjee, G. Ramakrishnan, and P. Jyothi, "Cross-lingual training for automatic question generation," <u>arXiv</u> preprint arXiv:1906.02525, 2019.
- [19] A. Krishnan, D. P, S. Ranu, and S. Mehta, "Leveraging semantic resources in diversified query expansion," <u>World Wide Web</u>, vol. 21, pp. 1041–1067, 2018.
- [20] S. B. Seidman and B. L. Foster, "A graph-theoretic generalization of the clique concept," <u>Journal of Mathematical sociology</u>, vol. 6, no. 1, pp. 139–154, 1978.
- [21] Y. Zhou, J. Xu, Z. Guo, M. Xiao, and Y. Jin, "Enumerating maximal k-plexes with worst-case time guarantee," in The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. AAAI Press, 2020, pp. 2442–2449.
- [22] A. Conte, T. De Matteis, D. De Sensi, R. Grossi, A. Marino, and L. 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, 2018, pp. 1272–1281.
- [23] B. Balasundaram, S. Butenko, and I. V. Hicks, "Clique relaxations in social network analysis: The maximum k-plex problem," <u>Operations</u> Research, vol. 59, no. 1, pp. 133–142, 2011.
- [24] Z. Wang, Y. Zhou, M. Xiao, and B. Khoussainov, "Listing maximal k-plexes in large real-world graphs," in <u>Proceedings of the ACM Web</u> <u>Conference 2022</u>, 2022, pp. 1517–1527.
- [25] Z. Wang, Q. Chen, B. Hou, B. Suo, Z. Li, W. Pan, and Z. G. Ives, "Parallelizing maximal clique and k-plex enumeration over graph data," <u>Journal of Parallel and Distributed Computing</u>, vol. 106, pp. 79–91, 2017.
- [26] B. Wu and X. Pei, "A parallel algorithm for enumerating all the maximal k-plexes," in Pacific-Asia conference on knowledge discovery and data mining. Springer, 2007, pp. 476–483.
- [27] Y. Zhou, S. Hu, M. Xiao, and Z.-H. Fu, "Improving maximum k-plex solver via second-order reduction and graph color bounding," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 14, 2021, pp. 12453–12460.
- [28] Y. Zhou, J. Xu, Z. Guo, M. Xiao, and Y. Jin, "Enumerating maximal k-plexes with worst-case time guarantee," in <u>Proceedings of the AAAI</u> <u>Conference on Artificial Intelligence</u>, vol. 34, no. 03, 2020, pp. 2442– 2449
- [29] R.-H. Li, Q. Dai, L. Qin, G. Wang, X. Xiao, J. X. Yu, and S. Qiao, "Efficient signed clique search in signed networks," in 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE, 2018, pp. 245–256.
- [30] N. Girdhar and K. Bharadwaj, "Signed social networks: a survey," in Advances in Computing and Data Sciences: First International Conference, ICACDS 2016, Ghaziabad, India, November 11-12, 2016, Revised Selected Papers 1. Springer, 2017, pp. 326–335.
- [31] X. Zheng, D. D. Zeng, and F. Wang, "Social balance in signed networks," Inf. Syst. Frontiers, vol. 17, no. 5, pp. 1077–1095, 2015.

- [32] F. Heider, "Attitudes and cognitive organization," The Journal of psychology, vol. 21, no. 1, pp. 107–112, 1946.
- [33] L. Chu, Z. Wang, J. Pei, J. Wang, Z. Zhao, and E. Chen, "Finding gangs in war from signed networks," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds. ACM, 2016, pp. 1505–1514.
- [34] D. Lo, D. Surian, K. Zhang, and E. Lim, "Mining direct antagonistic communities in explicit trust networks," in <u>Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011, C. Macdonald, I. Ounis, and I. Ruthven, Eds. ACM, 2011, pp. 1013–1018.</u>
- [35] Y. Su, B. Wang, F. Cheng, L. Zhang, X. Zhang, and L. Pan, "An algorithm based on positive and negative links for community detection in signed networks," <u>Scientific reports</u>, vol. 7, no. 1, pp. 1–12, 2017.
- [36] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph (algorithm 457)," Commun. ACM, vol. 16, no. 9, pp. 575–576, 1973.
- [37] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, "A new algorithm for generating all the maximal independent sets," <u>SIAM J. Comput.</u>, vol. 6, no. 3, pp. 505–517, 1977.
- [38] M. Bentert, A. Himmel, H. Molter, M. Morik, R. Niedermeier, and R. Saitenmacher, "Listing all maximal k-plexes in temporal graphs," ACM J. Exp. Algorithmics, vol. 24, no. 1, pp. 1.13:1-1.13:27, 2019.
- [39] Z. Wang, Q. Chen, B. Hou, B. Suo, Z. Li, W. Pan, and Z. G. Ives, "Parallelizing maximal clique and k-plex enumeration over graph data," J. Parallel Distributed Comput., vol. 106, pp. 79–91, 2017.
- [40] B. Wu and X. Pei, "A parallel algorithm for enumerating all the maximal k-plexes," in Emerging Technologies in Knowledge Discovery and Data Mining, PAKDD 2007, International Workshops, Nanjing, China, May 22-25, 2007, Revised Selected Papers, ser. Lecture Notes in Computer Science, T. Washio, Z. Zhou, J. Z. Huang, X. Hu, J. Li, C. Xie, J. He, D. Zou, K. Li, and M. M. Freire, Eds., vol. 4819. Springer, 2007, pp. 476–483.
- [41] Q. Dai, R. Li, H. Qin, M. Liao, and G. Wang, "Scaling up maximal k-plex enumeration," in Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022, M. A. Hasan and L. Xiong, Eds. ACM, 2022, pp. 345–354.