# Computing Maximum Structural Balanced Cliques in Signed Graphs

Kai Yao
*The University of Sydney*
kyao8420@uni.sydney.edu.au

Lijun Chang
*The University of Sydney*
lijun.chang@sydney.edu.au

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

***Abstract*—Signed graphs have been used to capture the polarity of relationships between entities through positive and negative edge signs, indicating friendly and antagonistic relationships, respectively. In this paper, we focus on (structural) balanced cliques in signed graphs, where a clique, denoted by its vertex set $C$, is (structural) balanced if it can be uniquely partitioned into two sets $C_L$ and $C_R$ such that all negative edges in the clique are between $C_L$ and $C_R$. We study the maximum balanced clique problem that aims to find the balanced clique $C^*$ such that $\min\{|C_L^*|, |C_R^*|\} \geq \tau$ for a user-given threshold $\tau$ and $|C^*|$ is the largest possible. We propose a novel graph reduction technique by transforming the maximum balanced clique problem over a signed graph $G$ to a series of maximum dichromatic clique problems over small subgraphs of $G$. That is, for a vertex $u$ in $G$, we first extract the subgraph $G_u$ of $G$ induced by vertex set $V_L \cup V_R$, where $V_L$ is the union of $u$ and its positive neighbors and $V_R$ is $u$'s negative neighbors. Then, we remove from $G_u$ all negative edges between vertices of the same set (i.e., $V_L$ or $V_R$) as well as remove all positive edges between $V_L$ and $V_R$; denote the resulting graph of discarding edge signs as $g_u$. We show that the maximum balanced clique containing $u$ in $G$ is the same as the maximum dichromatic clique (i.e., it has at least $\tau$ vertices from each of $V_L$ and $V_R$) containing $u$ in $g_u$. Due to the small size and no edge signs in $g_u$, the maximum dichromatic clique containing $u$ in $g_u$ can be efficiently computed by exploiting the existing pruning and bounding techniques that are designed for the classic maximum clique problem on unsigned graphs. Furthermore, we extend our techniques to the polarization factor problem which aims to find the largest $\tau$ such that there is a balanced clique $C$ with $\min\{|C_L|, |C_R|\} \geq \tau$, and to the generalized maximum balanced clique problem that reports a maximum balanced clique for each $\tau \geq 0$. Experimental studies on large real signed graphs demonstrated the efficiency and effectiveness of our techniques.***

## I. Introduction

Signed graphs enhance the representation capability of traditional graphs, by capturing the *polarity* of relationships between entities/vertices through *positive* and *negative* edge signs [1]. For example, signed graphs capture the friend-foe relationship in social networks [2], support-dissent opinions in opinion networks [3], trust-distrust relationship in trust networks [4], and activation-inhibition in protein-protein interaction networks [5]. One prominent and fundamental theory in signed graph analysis is the *structural balance theory* [6], which states that a signed (sub)graph is structural balanced if its vertices can be partitioned into two sets such that all edges inside each partition have positive signs and all cross-partition edges have negative signs. That is, "the friend of my friend is my friend", and "the friend of my enemy is my enemy".

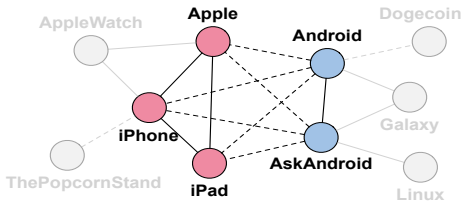



Fig. 1. A toy signed graph of Reddit

Many interesting problems, such as community detection [7], [8], link prediction [9], [10] and recommendation systems [11], [12], have been formulated and studied for signed graphs based on the structural balance theory.

Recently, the problem of enumerating all maximal structural balanced cliques in a signed graph is formulated and studied in [13]. A vertex set $C$ is a structural balanced clique if (1) it is a clique (i.e., every pair of its vertices is connected by an edge), and (2) it is structural balanced according to the structural balance theory (i.e., $C$ can be partitioned into two sets $C_L$ and $C_R$ such that all negative edges are between $C_L$ and $C_R$). For presentation simplicity, we refer to structural balanced cliques as *balanced cliques*. For example, for the toy signed graph in Figure 1 that captures the sentiment among different subreddits on Reddit, the three "Apple" groups (in red) and the two "Android" groups (in blue) together form a balanced clique; here, solid (resp. dashed) lines represent positive (resp. negative) sentiment. However, signed graphs may have an enormous number of maximal balanced cliques, as the enumeration problem is #P-complete. For example, the Douban dataset used in our experiments has more than $10^9$ maximal balanced cliques. Enumerating all of them may overwhelm end-users. To remedy this, a threshold $\tau$ is adopted in [13] such that only maximal balanced cliques $C$ satisfying $|C_L| \geq \tau$ and $|C_R| \geq \tau$ are enumerated. Nevertheless, the number of such cliques could still be large.

Motivated by this, we in this paper investigate the *maximum balanced clique problem*, aiming to find the *largest* balanced clique $C^* \subseteq V$ that satisfies $|C_L^*| \geq \tau$ and $|C_R^*| \geq \tau$ for a user-given threshold $\tau$. We prove that the maximum balanced clique problem is NP-hard. Detecting the maximum balanced clique has many applications; some examples are as follows.

- **Conflict Discovery.** Users actively interact with each other (both positively and negatively) on online platforms such as Facebook and Reddit, and these interactions can

be modeled as a signed graph [14]–[16]. Users in the maximum balanced clique are actively involved in conflicting groups, who have clear and firm standpoints on each other. Thus, they may represent the core members of two polarized structures, and detecting actively involved core members could help discover and prevent potential conflicts on the web.

- **Protein Complexes Detection.** Protein-protein interaction (PPI) networks can be modeled as signed networks to capture the activation-inhibition relations between proteins [17], [18]. As argued in [5], [19], protein complexes are ideally defined as groups of proteins within which are densely positively interacted (i.e., activation), and between which are densely negatively interacted (i.e., inhibition). Thus, detecting balanced cliques can help find protein complexes in signed PPI networks.
- **Synonym and Antonym Groups Discovery.** The synonyms and antonyms relationships between words can be naturally captured by a signed graph [20]. Thus, we can use maximum balanced clique to identify significant synonym groups that are antonymous with each other, which can be further used in applications such as semantic expansion [21] and automatic question generation [22].

Although the enumeration algorithm of [13] can be easily adapted to report the maximum balanced clique, this is inefficient. We propose an efficient branch-and-bound algorithm MBC$^*$ in this paper. Our algorithm is based on the following observation. Suppose we know that vertex $u$ is in the maximum balanced clique $C^*$. Then, according to the structural balance theory, it must satisfy $C^*_L \subseteq \{u\} \cup N^+_G(u)$ and $C^*_R \subseteq N^-_G(u)$, and therefore $C^*$ can be found in the subgraph $G_u$ of $G$ induced by $\{u\} \cup N^+_G(u) \cup N^-_G(u)$; here, $N^+_G(u)$ (resp. $N^-_G(u)$) is the set of positive (resp. negative) neighbors of $u$ in $G$. Moreover, based on the information of $C^*_L$ and $C^*_R$, we can sparsify $G_u$ by removing all *conflicting edges*: 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 $N^-_G(u)$. In addition, after removing the conflicting edges from $G_u$, we no longer need to explicitly consider the structural balance theory and thus edge signs, as every clique of $G_u$ will now be structural balanced. Let $g_u$ be the resulting graph of $G_u$ by discarding edge signs, and define $V_L = \{u\} \cup N^+_G(u)$ and $V_R = N^-_G(u)$. We call a clique in $g_u$ that has at least $\tau$ vertices from each of $V_L$ and $V_R$ as a *dichromatic clique*. It can be verified that $C^*$ is the maximum dichromatic clique in $g_u$ that includes $u$. As we do not know which vertex is in $C^*$, we need to enumerate each vertex $u$ of $G$ and suppose that $u$ is in $C^*$. That is, we transform the maximum balanced clique problem over $G$ to a series of maximum dichromatic clique problems over small subgraphs of $G$. Due to the small size and no edge signs in $g_u$, the maximum dichromatic clique containing $u$ in $g_u$ can be efficiently computed by exploiting the existing pruning and bounding techniques that are designed for the classic maximum clique problem on unsigned graphs. Moreover, our empirical study shows that most, if not all, of the instances of maximum dichromatic clique problem are directly pruned by the bounding techniques.

Both our maximum balanced clique problem and the maximal balanced clique enumeration problem studied in [13] require a user-given threshold $\tau$. However, it is unclear how to choose the appropriate $\tau$ for an application. Choosing a too large $\tau$ may lead to no result, while a too small $\tau$ may lead to skewed results as well as an enormous number of results for the enumeration problem of [13]. We provide two alternative ways to resolve this issue. Firstly, we formulate the *polarization factor problem*, which computes the largest $\tau^*$ such that $G$ has a balanced clique $C$ satisfying $|C_L| \geq \tau^*$ and $|C_R| \geq \tau^*$. We call this $\tau^*$ the *polarization factor* of $G$, denoted $\beta(G)$. It is immediate that there is no balanced clique for $\tau > \beta(G)$. Our empirical study shows that $\beta(G)$ varies from 3 to 201 for the graphs tested in our experiments. Thus, it would be interesting to know the polarization factor of a graph. We show that we can directly adapt our techniques of MBC$^*$ for computing $\beta(G)$. Secondly, instead of requiring users to input $\tau$, we report a maximum balanced clique for every $\tau \geq 0$; we term this problem as the *generalized maximum balanced clique problem*. It is easy to see that the maximum balanced clique for $\tau$ must be no smaller than that for $\tau + 1$, since an optimal solution to the latter is a feasible solution to the former. Thus, we vary $\tau$ from $\beta(G)$ to 0, and use the optimal solution to the problem for $\tau + 1$ as an initial solution to the problem for $\tau$, for the purpose of computation sharing.

**Contributions.** Our main contributions are as follows.

- We propose an efficient branch-and-bound algorithm MBC$^*$ to solve the maximum balanced clique problem. Our main idea is based on a novel graph reduction technique that transforms the problem over a large signed graph $G$ to a series of maximum dichromatic clique problems over small subgraphs of $G$; this not only removes edge signs but also sparsifies the edge set.
- We formulate the polarization factor problem, and modify MBC$^*$ to efficiently solve the polarization factor problem. The polarization factor $\beta(G)$ provides a rough guidance for choosing the threshold $\tau$ for our maximum balanced clique problem as well as the maximal balanced clique enumeration problem of [13].
- We also extend our techniques for the generalized maximum balanced clique problem that reports a maximum balanced clique for each $\tau \geq 0$.

Extensive empirical studies on large real signed graphs demonstrate that maximum balanced clique is more polarized than the community reported by PolarSeeds [15], and that our algorithms outperform baseline algorithms by up-to three orders of magnitude. For example, on the Douban dataset, MBC$^*$ finds the maximum balanced clique within 5 seconds, while the baseline algorithm takes more than 4 hours.

**Organization.** The rest of the paper is organized as follows. Section II provides preliminaries and defines the problems studied in this paper. Section III studies the maximum balanced clique problem, Section IV investigates the polariza-

tion factor problem, and Section V studies the generalized maximum balanced clique problem. Experimental results are reported in Section VI, and related works are discussed in Section VII. Finally, Section VIII concludes the paper. All the proofs for lemmas and theorems can be found in Appendix A.

## II. Preliminaries

In this paper, we focus on an *undirected* and *signed* graph $G = (V, E)$, where $V$ is the set of vertices and $E$ is the set of signed edges that is further partitioned into *positive* edges $E^+$ and *negative* edges $E^-$. We also denote a signed graph as $G = (V, E^+, E^-)$. We assume that the signed graph $G$ is *simple*, i.e., $E^+ \cap E^- = \emptyset$ and there is no self-loops. We denote the number of vertices and the number of edges by $n$ and $m$, respectively, i.e., $n = |V|$ and $m = |E| = |E^+| + |E^-|$. For each vertex $v \in V$, let $N_G^+(v) = \{u \mid (v,u) \in E^+\}$ be the set of *positive neighbors* of $v$ and $N_G^-(v) = \{u \mid (v,u) \in E^-\}$ be the set of *negative neighbors* of $v$. We use $d_G^+(v) = |N_G^+(v)|$ and $d_G^-(v) = |N_G^-(v)|$ to denote the *positive degree* and *negative degree* of $v$, respectively. We also use $N_G(v)$ and $d_G(v)$ to denote the (entire set of) neighbors and (total) degree of $v$, i.e., $N_G(v) = N_G^+(v) \cup N_G^-(v)$ and $d_G(v) = |N_G(v)| = d_G^+(v) + d_G^-(v)$. For ease of presentation, we omit the subscript $G$ in the notations when the context is clear. Given a vertex subset $S \subseteq V$, we use $G[S]$ to denote the *vertex-induced* subgraph of $G$ that consists of all edges between vertices of $S$, i.e., $G[S] = (S, \{(u,v) \in E \mid u \in S, v \in S\})$.

**Definition 1** (Structural Balanced Group [6])**.** Given a signed graph $G = (V, E^+, E^-)$, a group of vertices $C \subseteq V$ is *structural balanced* if it can be split into two subgroups $C_L$ and $C_R$ such that all edges between vertices in the same subgroup are positive and all edges between vertices from different subgroups are negative.

**Definition 2** (Structural Balanced Clique [13])**.** Given a signed graph $G$, a group of vertices $C \subseteq V$ is a *structural balanced clique* if (1) it is structural balanced and (2) it induces a clique, i.e., $(u,v) \in E^+ \cup E^-, \forall u, v \in C$ with $u \neq v$.

For simplicity, we refer to a structural balanced clique as a *balanced clique*. Consider the signed graph in Figure 2 where positive (resp. negative) edges are represented by solid (resp. dashed) lines. $C = \{v_1, v_2, v_3, v_4\}$ is a balanced clique with $C_L = \{v_1, v_2\}$ and $C_R = \{v_3, v_4\}$, or $C_L = \{v_3, v_4\}$ and $C_R = \{v_1, v_2\}$. We call $C_L$ and $C_R$ the two sides (e.g., left and right) of the balanced clique $C$. It is easy to verify that the splitting of $C$ into $C_L$ and $C_R$ is unique; nevertheless, the roles of $C_L$ and $C_R$ can be swapped. Thus, in the following, we directly use $C_L$ and $C_R$ to denote the two sides without formally defining them. Note that if all edges between vertices of $C$ are positive edges, then $C$ is also regarded as structural balanced. That is, one of $C_L$ and $C_R$ can be empty.

**Problem Statements.** We study two related problems.

**Problem 1** (Maximum Balanced Clique Problem)**.** Given a signed graph $G$ and a non-negative polarization threshold $\tau$, the maximum balanced clique problem aims to find the largest balanced clique $C^*$ in $G$ that satisfies the *polarization constraint* $\tau$ (i.e., $|C_L^*| \geq \tau$ and $|C_R^*| \geq \tau$).

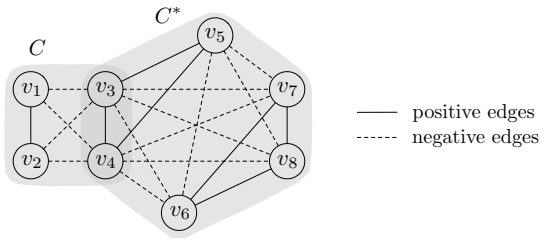



Fig. 2. A toy signed graph

In the problem definition, the polarization constraint $\tau$ requires each side (i.e., $C_L$ and $C_R$) to be of size at least $\tau$, in the same way as [13]. This is to make sure that the sides are not extremely small, and to avoid reporting too skewed results. Consider the signed graph in Figure 2 and suppose $\tau = 2$. Both $C = \{v_1, v_2, v_3, v_4\}$ and $C^* = \{v_3, v_4, v_5, v_6, v_7, v_8\}$ (also some subsets of $C^*$) are balanced cliques and satisfy the polarization constraint $\tau$, while $C^*$ is the largest one.

**Problem 2** (Polarization Factor Problem)**.** Given a signed graph $G = (V, E^+, E^-)$, the polarization factor problem aims to compute the largest $\tau^*$ such that $G$ has a balanced clique satisfying the polarization constraint $\tau^*$.

We call this largest $\tau^*$ the *polarization factor* of $G$, and denote it as $\beta(G)$. For example, the polarization factor of the signed graph in Figure 2 is 3.

Based on $\beta(G)$, we also extend our techniques to solve the maximum balanced clique problem for every $0 \leq \tau \leq \beta(G)$, and thus do not require end-users to input a threshold $\tau$. Note that, there will be no result for $\tau > \beta(G)$.

**NP-hardness.** We prove in the theorem below that both problems are NP-hard.

**Theorem 1.** *Both the maximum balanced clique problem and the polarization factor problem are NP-hard.*

All the proofs for lemmas and theorems can be found in Appendix A.

## III. Maximum Balanced Clique for a Fixed $\tau$

In this section, we study the maximum balanced clique problem. We first introduce an enumeration-based baseline algorithm MBC in Section III-A, and then propose a dichromatic clique-based efficient algorithm MBC$^*$ in Section III-B. We also present in Section III-C a heuristic algorithm for computing a large balanced clique, which is invoked by MBC$^*$.

### A. An Enumeration-based Baseline Approach MBC

The problem of enumerating all maximal balanced cliques has been studied in [13], and the enumeration algorithm MBCEnum proposed in [13] can be easily modified to find the maximum balanced clique. In the following, we first describe MBCEnum, and then present our enumeration-based baseline algorithm MBC for the maximum balanced clique problem.

**Algorithm 1:** MBC

**Input:** A signed graph $G = (V, E^+, E^-)$ and a threshold $\tau$
**Output:** The maximum balanced clique $C^*$

1 Reduce $G$ by VertexReduction and EdgeReduction of [13];
2 $C^* \leftarrow \emptyset$;
3 Enum$(\emptyset, \emptyset, V(G), V(G))$;
4 **return** $C^*$;

**Procedure** Enum$(C_L, C_R, P_L, P_R)$
5 **if** $|C_L| \geq \tau$ **and** $|C_R| \geq \tau$ **and** $|C_L| + |C_R| > |C^*|$ **then**
6 $\quad C^* \leftarrow C_L \cup C_R$;
7 **for each** $v \in P_L$ **do**
8 $\quad C'_L \leftarrow C_L \cup \{v\}$; $C'_R \leftarrow C_R$;
9 $\quad P'_L \leftarrow N^+(v) \cap P_L$; $P'_R \leftarrow N^-(v) \cap P_R$;
10 $\quad$ **if** $|C'_L| + |P'_L| \geq \tau$ **and** $|C'_R| + |P'_R| \geq \tau$ **and** $|C'_L| + |P'_L| + |C'_R| + |P'_R| > |C^*|$ **then**
11 $\quad\quad$ **if** $P'_R \neq \emptyset$ **then** Enum$(C'_R, C'_L, P'_R, P'_L)$;
12 $\quad\quad$ **else** Enum$(C'_L, C'_R, P'_L, P'_R)$;
13 $\quad P_L \leftarrow P_L \setminus \{v\}$;

MBCEnum is an adaptation of the classic BK algorithm, proposed in [24] for enumerating all maximal cliques in unsigned graphs, to enumerating all maximal balanced cliques. The general idea is to iteratively build up two sets $C_L$ and $C_R$ such that $C_L \cup C_R$ is always a balanced clique. In addition, it maintains two candidate sets $P_L$ and $P_R$ of vertices that can be used to grow $C_L$ and $C_R$, respectively. Specifically, $P_L$ (resp. $P_R$) is the set of vertices that are directly connected to every vertex of $C_L$ (resp. $C_R$) via positive edges and are directly connected to every vertex of $C_R$ (resp. $C_L$) via negative edges. Then, it tries each vertex of $P_L$ to be added to $C_L$ and each vertex of $P_R$ to be added to $C_R$, to grow the solution by one vertex and then conducts the recursion. Note that, MBCEnum also maintains two exclusive sets $X_L$ and $X_R$ of vertices that are used for certifying whether a solution $C_L \cup C_R$ is maximal or not; we do not discuss them here as they are not needed when computing the maximum balanced clique.

To improve the efficiency, MBCEnum [13] also proposed a vertex reduction method VertexReduction and an edge reduction method EdgeReduction to reduce the input graph based on the polarization threshold $\tau$. The general idea of VertexReduction is that every vertex in a balanced clique satisfying the polarization constraint must have a positive degree at least $\tau - 1$ and a negative degree at least $\tau$; thus, all vertices violating these degree constraints can be removed. The general idea of EdgeReduction is that every edge in a balanced clique satisfying the polarization constraint must participate in a certain number of triangles of each type; we omit the details, which can be found in [13]. VertexReduction can be conducted in $\mathcal{O}(n+m)$ time, while EdgeReduction takes $\mathcal{O}(m^{3/2})$ time.

Based on MBCEnum, we have a baseline algorithm MBC for the maximum balanced clique problem. The pseudocode of MBC is shown in Algorithm 1, which is self-explanatory. It can be easily verified that all calls to Enum$(C_L, C_R, P_L, P_R)$, except the first one, guarantee that $C_L \cap C_R = \emptyset$, $P_L \cap P_R = \emptyset$ and $(C_L \cup C_R) \cap (P_L \cup P_R) = \emptyset$. Note that at Line 11, we swap the roles of $C_L$ (resp. $P_L$) and $C_R$ (resp. $P_R$) such that we are adding vertices to the two sides of the growing balanced clique in alternating order; this is to avoid generating too skewed intermediate results.

### *B. A Maximum Dichromatic Clique-based Approach* MBC*

Our empirical studies in Section VI show that the MBC algorithm is inefficient, due to lack of advanced pruning and bounding techniques. That is, only size-based upper bound is used for pruning (see Line 10 of Algorithm 1). To improve efficiency, we aim to utilize (some of) the advanced pruning and bounding techniques that have been shown to be successful for the classic maximum clique problem in unsigned graphs [25]–[27]. Specifically, we aim to utilize the *degree-based pruning* and *graph coloring-based upper bounding* for computing maximum balanced clique. To illustrate, let's consider an *unsigned* graph, and let $lb$ be a lower bound of the maximum clique size which is set as the largest size of the enumerated cliques. The degree-based pruning is as follows.

**Lemma 1** (Degree-based pruning)**.** *If the degree of a vertex $u$ is less than $lb$, then we can remove $u$ from the graph without affecting the maximum clique computation.*

Degree-based pruning is correct because we have already found a clique of size $lb$, and we are now searching for cliques of size larger than $lb$. A *coloring* of a graph is to assign a color to each vertex of the graph such that no two adjacent vertices have the same color. The smallest number of colors needed to color a graph is called its *chromatic number*.

**Lemma 2** (Graph coloring-based upper bounding)**.** *The maximum clique size of a graph is at most its chromatic number.*

The correctness is easy to see as each vertex of a clique requires a different color. However, computing the chromatic number is an NP-hard problem [28]. Thus, heuristic techniques (e.g., see [27]) are usually used in practice to compute an upper bound of the chromatic number, which then is also an upper bound of the maximum clique size.

**Ineffectiveness of A Naive Strategy.** However, it is nontrivial to effectively apply these pruning and bounding techniques to balanced clique computation on signed graphs. This is because we now have both positive edges and negative edges, and we also need to satisfy the structural balanced constraint. One possible way to utilize these techniques for signed graphs is ignoring the edge signs and the structural balanced constraint when conducting pruning and bounding. This is correct, but is ineffective as verified by our experiments. First, the structural balanced constraint is not considered due to ignoring edge signs. Second, the number of edges is abundant which makes the pruning and bounding ineffective. Consider the signed graph in Figure 3 as an example. The number of colors needed to color its vertices after ignoring edge signs is 6 as there is an edge between every pair of vertices, and thus the coloring-based upper bound is 6. But it is easy to see that the maximum balanced clique size is 3 for $\tau = 0$, and is 2 for $\tau = 1$.

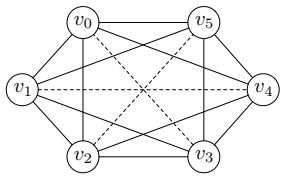
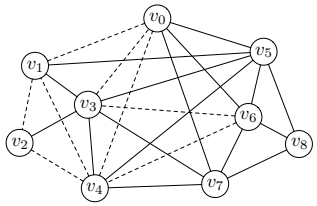
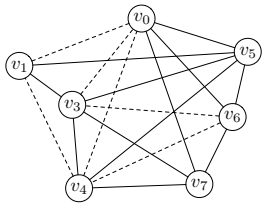


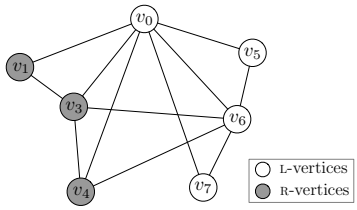

Fig. 3. Ineffective of coloring-based upper bound by ignoring edge signs




(a) A signed graph




(b) Ego-network $G_{v_0}$ of $v_0$




(c) Dichromatic-network $g_{v_0}$ of $v_0$

Fig. 4. Illustration of our transformation

**A Novel Graph Reduction Technique.** To resolve the above drawbacks, we propose a novel graph reduction technique. The general idea is based on the following observation that removes edges from a subgraph. Suppose we know that vertex $u$ is in the maximum balanced clique $C^*$. Then, according to the structural balance theory, it must satisfy $C_L^* \subseteq \{u\} \cup N_G^+(u)$ and $C_R^* \subseteq N_G^-(u)$; here, without loss of generality, we assume $u \in C_L^*$. This is because by the structural balance theory, each vertex of $N_G^+(u)$ has a positive edge to $u$ and thus cannot be on the opposite side of $u$, and similarly vertices of $N_G^-(u)$ cannot be on the same side as $u$. Therefore, $C^*$ can be found in the subgraph $G_u$ of $G$ induced by vertices $\{u\} \cup N_G^+(u) \cup N_G^-(u)$. Moreover, based on the information of $C_L^*$ and $C_R^*$, we can sparsify the subgraph $G_u$ by removing all *conflicting edges*:

- 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 $N_G^-(u)$.

This is because, if $v, v' \in N_G^+(u)$ are connected by a negative edge, then $v$ and $v'$ cannot be both in $C^*$; recall that $C_L^* \subseteq \{u\} \cup N_G^+(u)$. Similarly, if $v \in N_G^+(u)$ and $v'' \in N_G^-(u)$ are connected by a positive edge, then $v$ and $v''$ cannot be both in $C^*$. As a result, all the conflicting edges are not in $G[C^*]$ and thus can be safely removed. In addition, after removing the conflicting edges from $G_u$, we no longer need to explicitly consider the structural balance theory and thus edge signs; this is because every clique of $G_u$ will now be structural balanced in $G$. Let $g_u$ be the resulting graph of $G_u$ by discarding edge signs, and define $V_L = \{u\} \cup N_G^+(u)$ and $V_R = N_G^-(u)$. It is easy to verify that $C^*$ is the maximum clique in $g_u$ that includes $u$ and has at least $\tau$ vertices from each of $V_L$ and $V_R$. Thus, our problem becomes finding such a maximum clique in $g_u$; we term this problem as maximum dichromatic clique problem, which is formally defined as follows.

**Problem 3** (Maximum Dichromatic Clique Problem)**.** The input of the maximum dichromatic clique problem consists of a *dichromatic graph* $g = (V(g), E(g))$ — where the vertices $V(g)$ is further partitioned into the set of L-vertices $V_L$ and the set of R-vertices $V_R$ with $V_L \cap V_R = \emptyset$ — and a non-negative threshold $\tau$. It aims to find the largest clique $C^* \subseteq V(g)$ such that $|C^* \cap V_L| \geq \tau$ and $|C^* \cap V_R| \geq \tau$. We call a clique $C$ satisfying $|C \cap V_L| \geq \tau$ and $|C \cap V_R| \geq \tau$ a *dichromatic clique* satisfying the constraint $\tau$.

The above discussion is based on the assumption that we know a vertex $u$ in the maximum balanced clique. However, in practice, we do not know such a vertex. To remedy this, we enumerate each vertex $u \in V(G)$ and compute a maximum balanced clique containing $u$; the largest one among all the enumerated balanced cliques then is the result. Actually, we can do better by giving a total ordering $\prec$ to the vertices, and only considering higher-ranked neighbors in constructing $G_u$ and $g_u$. In summary, we transform the maximum balanced clique problem over a large signed graph $G$ to a series of maximum dichromatic clique problems over small subgraphs of $G$, as follows. Given a signed graph $G = (V, E^+, E^-)$ and any total ordering $\prec$ of $V$, for each vertex $u \in V$,

1) We first extract the *ego-network* of $u$, denoted $G_u$, which is the subgraph of $G$ induced by $u$ and $u$'s *higher-ranked* neighbors according to the total ordering $\prec$.
2) We then transform $G_u$ into a *dichromatic-network* of $u$, denoted $g_u$, which ***not only removes edge signs but also sparsifies the edge set***. Specifically, let $V_L$ be the union of $u$ and $u$'s positive neighbors in $G_u$, and $V_R$ be the set of $u$'s negative neighbors in $G_u$. We label vertices of $V_L$ as L-vertices, and vertices of $V_R$ as R-vertices. Then, we remove all *conflicting edges* from $G_u$: all negative edges between L-vertices, all negative edges between R-vertices, and all positive edges between L-vertices and R-vertices. The resulting graph after discarding edge signs is a dichromatic graph $g_u$.

**Example 1.** *For example, consider the signed graph in Figure 4(a) and assume $v_0$ has the lowest rank according to the total ordering $\prec$. Figure 4(b) illustrates the ego-network $G_{v_0}$ of $v_0$ which does not include vertices $\{v_2, v_8\}$, and Figure 4(c) shows the dichromatic-network $g_{v_0}$ of $v_0$ which removes edges $\{(v_1, v_4), (v_1, v_5), (v_3, v_5), (v_4, v_5), (v_3, v_7), (v_4, v_7)\}$. Note that, in our implementation, we actually exclude $u$ and its adjacent edges from $G_u$ and $g_u$. Then, the effect of edge reduction becomes more evident; for example, $G_{v_0}$ has 12 edges while $g_{v_0}$ only has 6 edges, after excluding $v_0$.*

We prove in the theorem below that the maximum balanced clique in $G$ can be obtained by computing maximum dichromatic cliques in the dichromatic networks $g_u$ for all $u \in V$.

**Theorem 2.** *The size of the maximum balanced clique in $G = (V, E^+, E^-)$ that satisfies the constraint $\tau$ is equal to $\max_{u \in V} \delta(g_u, \tau)$, where $\delta(g_u, \tau)$ denotes the size of the maximum dichromatic clique in $g_u$ satisfying the constraint $\tau$.*

There are two main benefits of transforming the maximum balanced clique problem over $G$ to a series of maximum dichromatic clique problems over dichromatic networks of $G$.

- Firstly, each dichromatic network is small as discussed above. Although we need to solve $n$ instances of the maximum dichromatic clique problem in the worst case, our empirical studies in Section VI show that we only need to solve a small number of such instances (e.g., at most hundreds) in practice. This is because most of the instances are directly pruned by the degree-based pruning and coloring-based upper bounding.
- Secondly, by transforming the maximum balanced clique searching on signed graphs to maximum dichromatic clique searching on dichromatic graphs, degree-based pruning and graph coloring-based upper bounding come into effect, which can significantly reduce the search space and thus speed up the computation.

**Algorithm 2:** MBC*

**Input:** A signed graph $G = (V, E^+, E^-)$, and a threshold $\tau$
**Output:** The maximum balanced clique $C^*$

1 Reduce $G$ by VertexReduction of [13];
2 $C^* \leftarrow$ MBC-Heu$(G, \tau)$;
3 Reduce $G$ to its $|C^*|$-core;
4 DOrder$(\cdot) \leftarrow$ degeneracy ordering of vertices;
5 **for each** *vertex $u$ in reverse order w.r.t.* DOrder$(\cdot)$ **do**
6 $\quad g_u \leftarrow$ the dichromatic-network of $u$;
7 $\quad g \leftarrow$ the $|C^*|$-core of $g_u$;
8 $\quad$ **if** colorUB$(g) > |C^*|$ **then** MDC$(\{u\}, g \setminus \{u\}, \tau - 1, \tau)$;
9 **return** $C^*$;

**Procedure** MDC$(C, g = (V_L, V_R, E), \tau_L, \tau_R)$
10 **if** $|C| > |C^*|$ **and** $\tau_L \leq 0$ **and** $\tau_R \leq 0$ **then** $C^* \leftarrow C$;
11 Reduce $g$ to its $(|C^*| - |C|)$-core;
12 **if** $|V_L(g)| < \tau_L$ **or** $|V_R(g)| < \tau_R$ **or** colorUB$(g) \leq |C^*| - |C|$ **then**
13 $\quad$ **return** ;
14 **if** $\tau_L > 0$ **and** $\tau_R \leq 0$ **then** $\mathcal{B} \leftarrow V_L(g)$;
15 **else if** $\tau_L \leq 0$ **and** $\tau_R > 0$ **then** $\mathcal{B} \leftarrow V_R(g)$;
16 **else** $\mathcal{B} \leftarrow V_L(g) \cup V_R(g)$;
17 **while** $\mathcal{B} \neq \emptyset$ **do**
18 $\quad v \leftarrow$ the vertex of $\mathcal{B}$ with the minimum degree in $g$;
19 $\quad$ **if** $v \in V_L(g)$ **then** $\tau_L' \leftarrow \tau_L - 1$; $\tau_R' \leftarrow \tau_R$;
20 $\quad$ **else** $\tau_L' \leftarrow \tau_L$; $\tau_R' \leftarrow \tau_R - 1$;
21 $\quad$ MDC$(C \cup \{v\}, g[N_g(v)], \tau_L', \tau_R')$;
22 $\quad$ Remove $v$ from $\mathcal{B}$ and $g$;

**Pseudocode of MBC*.** Based on the above discussions, the pseudocode of our dichromatic clique-based algorithm for the maximum balanced clique problem is shown in Algorithm 2, denoted MBC*. We first apply the VertexReduction of [13] to reduce the input signed graph $G$ (Line 1). Note that, we do not apply EdgeReduction of [13]; this is because it has a high time complexity and incurs a large overhead for our efficient algorithm MBC*, as verified by our experiments in Section VI. Next, we heuristically compute a balanced clique by invoking MBC-Heu (Line 2), which will be presented in Section III-C. Then, we reduce $G$ to its $|C^*|$-core and obtain the degeneracy ordering of vertices (Lines 3–4), by ignoring edge signs. Here, the ***$k$-core*** is the maximal subgraph that has minimum degree at least $k$ [29], which can be obtained by iteratively removing vertices of degree smaller than $k$. The ***degeneracy ordering*** (aka smallest-first ordering) is defined recursively as follows: the first vertex has the smallest degree in the graph, the second vertex has the smallest degree in the resulting graph after removing all preceding vertices, and so on [29]. Based on the degeneracy ordering DOrder, we process vertices in the reverse order according to DOrder (Line 5); we say a vertex ranks higher if it appears later in DOrder. For each vertex $u$, we extract the dichromatic-network $g_u$ of $u$ (Line 6), and then compute the maximum dichromatic clique in $g_u$ (Lines 7–8). The intuition of using the degeneracy ordering is that the ego-networks as well as the dichromatic networks then will have fewer vertices.

Efficiently computing the maximum dichromatic clique in the dichromatic network $g_u$ is critical to the performance of our algorithm MBC*. We propose a branch-and-bound algorithm, MDC, to solve this problem efficiently, whose pseudocode is also given in Algorithm 2. The input of MDC consists of a clique $C$, a dichromatic graph $g = (V_L, V_R, E)$, and two thresholds $\tau_L$ and $\tau_R$. It aims to find the largest dichromatic clique $C'$ in $g$ such that $|C' \cap V_L| \geq \tau_L$ and $|C' \cap V_R| \geq \tau_R$; then $C \cup C'$ is used to update $C^*$ (Line 10). Note that, the existing maximum clique solvers for unsigned graphs cannot be directly applied here due to the existence of the two thresholds $\tau_L$ and $\tau_R$. We first reduce $g$ to its $(|C^*| - |C|)$-core at Line 11 by ignoring vertex labels (i.e., L and R); this is because we are looking for a dichromatic clique $C'$ such that $|C \cup C'| > |C^*|$. We prune the instance (i.e., $g$) if there is no feasible dichromatic clique (i.e., $|V_L(g)| < \tau_L$ or $|V_R(g)| < \tau_R$) or a computed coloring-based upper bound of the maximum clique size by ignoring vertex labels is no larger than $|C^*| - |C|$ (Lines 12-13). If the instance is not pruned, then we generate new branches. First, the branching vertices $\mathcal{B}$ are selected based on $\tau_L$ and $\tau_R$ as follows. If $\tau_L > 0$ and $\tau_R \leq 0$, the next vertex to be included into $C$ (i.e., $\mathcal{B}$) should be ideally from $V_L(g)$ (Line 14). Similarly, if $\tau_L \leq 0$ and $\tau_R > 0$, the next vertex to be included into $C$ should be from $V_R(g)$ (Line 15). Otherwise, we can choose any vertex from $V_L(g) \cup V_R(g)$ (Line 16). Then, while $\mathcal{B}$ is not empty (Line 17), we choose the vertex $v$ of $\mathcal{B}$ that has the smallest degree in $g$ (Line 18), to generate a new branch at Line 21 by including $v$ into $C$ and then recursively solve the problem on the subgraph of $g$ induced by $v$'s neighbors (i.e., $g[N_g(v)]$). After processing $v$, we remove $v$ from both $\mathcal{B}$ and $g$ (Line 22).

To reduce the number of MDC instances to be generated at Line 8 of Algorithm 2, we conduct the following prunings for $g_u$. We first reduce $g_u$ to its $|C^*|$-core, denoted by $g$, by ignoring vertex labels (Line 7); this is because we are now searching for a clique of size at least $|C^*| + 1$. Then, we compute a coloring-based upper bound for the maximum clique size of $g$ after ignoring vertex labels. If this upper bound is no larger than $|C^*|$, then we prune the graph $g$ without calling MDC (Line 8).

**Theorem 3.** *The time complexity of MBC* (Algorithm 2) is $O(n \cdot m \cdot 2^{\delta})$ and the space complexity is $O(m)$, where $\delta$ is the degeneracy of $G$.*

### C. A Heuristic Algorithm MBC-Heu

In this subsection, we present a heuristic algorithm MBC-Heu for the maximum balanced clique problem, which is invoked at Line 2 of Algorithm 2 for initialization. Let $u$ be the vertex of maximum degree in the input graph $G$. We extract the dichromatic network $g_u$ of $u$, and then iteratively grow a clique $C$ in $g_u$. That is, $C$ is initialized as $\{u\}$. Let $g$ be the subgraph of $g_u$ induced by the set of vertices that are adjacent to all vertices of $C$. We greedily include into $C$ the vertex that has the maximum degree in $g$, and then update $g$; we repeat this process until $g$ is empty. To balance the sizes of $|C \cap V_L|$ and $|C \cap V_R|$, we add vertices into $C$ from $V_L$ and from $V_R$ in an alternating order.

**Algorithm 3:** MBC-Heu

**Input:** A signed graph $G = (V, E^+, E^-)$, and a threshold $\tau$
**Output:** A balanced clique $C$ satisfying the constraint $\tau$ or $\emptyset$

```
1  u ← maximum-degree vertex in G;
2  g ← the dichromatic network g_u of u;
3  C ← {u};
4  while V_L(g) ≠ ∅ or V_R(g) ≠ ∅ do
5      if V_L(g) = ∅ or (V_R(g) ≠ ∅ and |C_L| ≥ |C_R|) then
6          v ← maximum-degree vertex of V_R(g) in g;
7      else v ← maximum-degree vertex of V_L(g) in g;
8      C ← C ∪ {v}; g ← g[N_g(v)];
9  if |C_L| ≥ τ and |C_R| ≥ τ then return C;
10 else return ∅;
```

The pseudocode of our heuristic algorithm MBC-Heu is shown in Algorithm 3, which is self-explanatory. Note that, we return the clique $C$ only when it satisfies the polarization constraint $\tau$ (Lines 9–10). In our implementation, we run the heuristic algorithm for the vertex $u$ with the largest value of $\min\{d^+(u), d^-(u)\}$.

**Theorem 4.** *The time complexity and space complexity of* MBC-Heu *(Algorithm 3) are both $\mathcal{O}(m)$.*

## IV. Polarization Factor

In this section, we study the polarization factor problem. We first present an enumeration-based baseline algorithm PF-E in Section IV-A. Then we in Section IV-B present a binary search algorithm PF-BS which invokes our MBC* algorithm (proposed in Section III-B) multiple times. Finally, we in Section IV-C propose an algorithm PF* by adapting the MBC* algorithm to directly solve the polarization factor problem.

### A. An Enumeration-based Baseline Algorithm PF-E

Recall that the polarization factor of a signed graph $G = (V, E^+, E^-)$, denoted $\beta(G)$, is the largest $\tau$ such that $G$ has a balanced clique $C$ satisfying $|C_L| \geq \tau$ and $|C_R| \geq \tau$. Similar to Section III-A, we can modify the maximal balanced clique enumeration algorithm MBCEnum [13] to compute $\beta(G)$. We denote this enumeration-based algorithm as PF-E. The correctness of PF-E is immediate. However, the time cost of PF-E is expensive as one would expect.

### B. A Binary Search Algorithm PF-BS

It is obvious that for any $\tau \leq \beta(G)$, we can always find a balanced clique in $G$ satisfying the constraint $\tau$, and for any $\tau > \beta(G)$, we will not be able to find any. Thus, we can iteratively invoke our algorithm MBC* proposed in Section III-B for increasing $\tau$ to compute $\beta(G)$. That is, we stop when MBC* is not able to find any balanced clique satisfying the constraint $\tau$, and then $\tau - 1$ is the result.

A better approach is binary searching the value of $\beta(G)$, which also invokes MBC* for each guessing $\tau$ of $\beta(G)$. The lower bound of $\beta(G)$ is 0, and the upper bound can be set as $\max_{v \in V}\{\min\{d^+(v) + 1, d^-(v)\}\}$. We denote our binary search-based algorithm as PF-BS, and we omit the pseudocode. The correctness of PF-BS is immediate.

**Optimization.** MBC* was designed to find the maximum balanced clique satisfying the polarization constraint $\tau$. However, for the polarization factor problem, we only need to know whether there exists a balanced clique satisfying the polarization constraint $\tau$ or not, i.e., the balanced clique does not need to be *maximum*. Thus, in each invocation of MBC*, we can terminate the execution of MBC* once both $\tau_L$ and $\tau_R$ (see Algorithm 2) decrease to 0.

### C. A Dichromatic Clique Checking-based Algorithm PF*

The PF-BS algorithm invokes MBC* as a black box, and needs to invoke MBC* $\mathcal{O}(\log n)$ times in the worst case. In this subsection, we look into the algorithm MBC*, and directly adapt MBC* for computing the polarization factor $\beta(G)$.

Firstly, similar to Theorem 2, we have the lemma below that transforms the polarization factor problem over $G$ to a series of another problem over small subgraphs of $G$. Here, for each dichromatic network $g_u$, we compute the largest $\tau$ such that $g_u$ has a dichromatic clique satisfying the constraint $\tau$, instead of computing the maximum dichromatic clique for a given $\tau$.

**Lemma 3.** *For any signed graph $G = (V, E^+, E^-)$ and a total ordering $\prec$ of $V$, we have $\beta(G) = \max_{u \in V} \gamma(g_u)$, where $g_u = (V_L, V_R, E)$ is the dichromatic network of $u$ as defined in Section III-B, and $\gamma(g_u)$ denotes the largest $\tau$ such that $g_u$ has a dichromatic clique satisfying the constraint $\tau$.*

Let $(v_1, v_2, \ldots, v_n)$ be any total ordering of $V$. We process vertices in the reverse order according to the total ordering. Following Lemma 3, when processing vertex $v_i$, we need to compute the largest $\tau$ such that $g_{v_i}$ has a dichromatic clique satisfying the constraint $\tau$. Actually, we can do better. The general idea is that let $\tau = \max_{j=i+1}^{n} \gamma(g_{v_j})$, then we only need to verify whether $g_{v_i}$ has a dichromatic clique satisfying the constraint $\tau + 1$; we call this problem the *dichromatic clique checking* problem. This is because $\gamma(g_{v_i})$ cannot be larger than $\tau + 1$, as proved in the lemma below.

**Lemma 4.** *Let $(v_1, v_2, \ldots, v_n)$ be a total ordering of $V$. We have $\gamma(g_{v_i}) \leq \max_{j=i+1}^{n} \gamma(g_{v_j}) + 1$ for all $i \geq 1$.*

The pseudocode of our algorithm is given in Algorithm 4, denoted PF*. First, we heuristically compute a polarization value by invoking MBC-Heu$(G, 0)$ (Line 1). Note that,

**Algorithm 4:** PF*

**Input:** A signed graph $G = (V, E^+, E^-)$
**Output:** The polarization factor $\beta(G)$

1 $\tau^* \leftarrow$ MBC-Heu$(G, 0)$;
2 Reduce $G$ by VertexReduction of [13] based on $\tau^* + 1$;
3 POrder$(\cdot) \leftarrow$ PDecompose(G);
4 **for each** *vertex* $u$ *in reverse order w.r.t.* POrder$(\cdot)$ **do**
5 $\quad g_u \leftarrow$ the dichromatic-network of $u$;
6 $\quad g \leftarrow$ the $(\tau^* + 1, \tau^* + 1)$-core of $g_u$;
7 $\quad$ **if** $u \in V_L(g)$ **then**
8 $\quad\quad$ **if** DCC$(g \setminus \{u\}, \tau^*, \tau^* + 1)$ **then** $\tau^* \leftarrow \tau^* + 1$;
9 **return** $\tau^*$;

**Procedure** DCC$(g = (V_L, V_R, E), \tau_L, \tau_R)$
**Output:** **true** if $g$ contains a valid dichromatic clique for thresholds $\tau_L$ and $\tau_R$; **false** otherwise

10 **if** $\tau_L = 0$ **and** $\tau_R = 0$ **then return true**;
11 Reduce $g$ to its $(\tau_L, \tau_R)$-core;
12 **if** $\tau_L > 0$ **and** $\tau_R = 0$ **then** $\mathcal{B} \leftarrow V_L(g)$;
13 **else if** $\tau_L = 0$ **and** $\tau_R > 0$ **then** $\mathcal{B} \leftarrow V_R(g)$;
14 **else** $\mathcal{B} \leftarrow V_L(g) \cup V_R(g)$;
15 **while** $\mathcal{B} \neq \emptyset$ **do**
16 $\quad v \leftarrow$ the vertex of $\mathcal{B}$ with the minimum degree in $g$;
17 $\quad$ **if** $v \in V_L(g)$ **then** $\tau'_L \leftarrow \tau_L - 1$; $\tau'_R \leftarrow \tau_R$;
18 $\quad$ **else** $\tau'_L \leftarrow \tau_L$; $\tau'_R \leftarrow \tau_R - 1$;
19 $\quad$ **if** DCC$(g[N_g(v)], \tau'_L, \tau'_R)$ **then return true**;
20 $\quad$ Remove $v$ from $\mathcal{B}$ and $g$;
21 **return false**;

MBC-Heu$(G, 0)$ actually returns a balanced clique $C$; we set $\tau^* = \min\{|C_L|, |C_R|\}$. Based on $\tau^*$, we reduce the input graph $G$ by applying VertexReduction of [13] (Line 2). Next, we compute a total ordering POrder of $V$ by invoking PDecompose (Line 3). Based on the total ordering POrder, we process vertices in reverse order according to POrder (Line 4). For each vertex $u$, we extract the dichromatic-network $g_u$ of $u$ (Line 5), and then reduce $g_u$ to its $(\tau^* + 1, \tau^* + 1)$-core (Line 6), denoted $g$. If $u$ is not pruned (Line 7), then we invoke DCC to check whether $g$ has a dichromatic clique $C$ satisfying $|C \cap V_L(g)| \geq \tau^* + 1$ and $|C \cap V_R(g)| \geq \tau^* + 1$; if the answer is yes, then we increase $\tau^*$ by 1 (Line 8). Here, the $(\tau_L, \tau_R)$-core for non-negative integers $\tau_L$ and $\tau_R$ is the (unique) maximal subgraph $g = (V_L, V_R, E)$ such that every vertex of $V_L$ (resp. $V_R$) has at least $\tau_L - 1$ (resp. $\tau_L$) neighbors in $V_L$ and at least $\tau_R$ (resp. $\tau_R - 1$) neighbors in $V_R$. The motivation of computing the $(\tau_L, \tau_R)$-core is that every vertex participating in a dichromatic clique $C$ satisfying $|C \cap V_L(g_u)| \geq \tau_L$ and $|C \cap V_R(g_u)| \geq \tau_R$ must be in the $(\tau_L, \tau_R)$-core. Note that, the $(\tau_L, \tau_R)$-core can be computed in linear time by iteratively removing vertices that violate the conditions.

The pseudocode of DCC is also given in Algorithm 4. It is similar to MDC in Algorithm 2, except that (1) we now do not need to keep track of the growing clique $C$, and (2) we can stop the execution (Lines 10 and 20) once both $\tau_L$ and $\tau_R$ become 0; note that, in DCC, $\tau_L$ and $\tau_R$ will not go below 0, while it is possible in MDC. It is worth pointing out that we first greedily add $u$ to the clique $C$ at Line 8 before invoking DCC; this is because if $u$ is not in the clique, then invoking DCC definitely will return false according to Lemma 4.

**Theorem 5.** *The time complexity of* PF* *(Algorithm 4) is* $O(n \cdot m \cdot 2^{\delta})$ *if we use the degeneracy ordering at Line 3. The space complexity is* $O(m)$.

**Polarization Order.** Any total ordering of $V$ can be used at Line 4 of Algorithm 4 for correctly computing $\beta(G)$. One possibility is to use the degeneracy ordering as used in Algorithm 2. In the following, we propose a new total ordering, called polarization order, to improve the time efficiency. First, we define the polarization core.

**Definition 3** (Polarization Core)**.** Given a signed graph $G = (V, E^+, E^-)$ and an integer $k$, the $k$-polarization core (abbreviated as $k$-polar-core) of $G$ is the maximal subgraph $g$ of $G$ such that $\min\{d_g^+(u) + 1, d_g^-(u)\} \geq k$, $\forall u \in V(g)$.

We define the *polar-core number* of a vertex $u$, denoted $pn(u)$, as the largest $k$ such that $u$ is contained in the $k$-polar-core. The reason of defining these concepts is that $pn(u)$ provides an upper bound of $\gamma(g_u)$ for any total ordering $\prec$, as provided in the lemma below. Consequently, if we process vertices in non-increasing polar-core number, we are likely to obtain a large lower bound of $\beta(G)$ by processing the first few ego-networks (or dichromatic networks), and thus prune a lot of invocations to DCC based on the lower bound of $\beta(G)$.

**Lemma 5.** *For any vertex* $u$ *in* $G$*, we have* $pn(u) \geq \gamma(g_u)$ *for any total ordering* $\prec$.

**Algorithm 5:** PDecompose

**Input:** A signed graph $G = (V, E^+, E^-)$
**Output:** Polarization order POrder

1 **for each** $v \in V$ **do**
2 $\quad d^+(v) \leftarrow$ the positive degree of $v$ in $G$;
3 $\quad d^-(v) \leftarrow$ the negative degree of $v$ in $G$;
4 POrder $\leftarrow \emptyset$;
5 **for** $i \leftarrow 1$ ***to*** $n$ **do**
6 $\quad u \leftarrow \arg\min_{v \in V \setminus \text{POrder}} \min\{d^+(v) + 1, d^-(v)\}$;
7 $\quad pn(u) \leftarrow \min\{d^+(u) + 1, d^-(u)\}$;
8 $\quad$ Add $u$ to the tail of POrder;
9 $\quad$ **for each** *positive neighbor* $v$ *of* $u$ *that not in* POrder **do**
10 $\quad\quad$ **if** $d^+(v) + 1 > pn(u)$ **then** $d^+(v) \leftarrow d^+(v) - 1$;
11 $\quad$ **for each** *negative neighbor* $v$ *of* $u$ *that not in* POrder **do**
12 $\quad\quad$ **if** $d^-(v) > pn(u)$ **then** $d^-(v) \leftarrow d^-(v) - 1$;
13 **return** POrder;

We call the total ordering that ranks vertices in non-decreasing polar-core number the *polarization order*. The polarization order can be computed in a similar way to the peeling algorithm for computing the degeneracy ordering [29]. The pseudocode code of our algorithm, denoted PDecompose, is shown in Algorithm 5, which is self-explanatory. The general idea is to iteratively remove from $G$ the vertex $u$ that has the smallest $\min\{d^+(u) + 1, d^-(u)\}$ in the current graph, and add $u$ to the end of POrder. By using the bin-sort like data structure as discussed in [29], PDecompose can be

implemented to run in $\mathcal{O}(n+m)$ time and $\mathcal{O}(m)$ space; we omit the details.

## V. Maximum Balanced Clique for Every $\tau$

In this section, we investigate the problem of computing a maximum balanced clique for each $\tau \geq 0$, which then alleviate the end-user from specifying the threshold $\tau$. We term this problem as the *generalized maximum balanced clique problem.*

A straightforward method is to invoke MBC* independently for each $\tau$ from 0 to $\beta(G)+1$. Note that, as $\beta(G)$ is not an input to this problem, we need to invoke MBC* for $\tau = \beta(G)+1$ which would return an empty result and thus indicate that this $\tau$ is larger than $\beta(G)$. We denote this algorithm as gMBC. We can do better by sharing computation among the different invocations to MBC*, based on the lemma below.

**Lemma 6.** *Given a signed graph $G$ and thresholds $\tau_1$ and $\tau_2$ s.t. $0 \leq \tau_1 < \tau_2 \leq \beta(G)$, the maximum balanced clique for threshold $\tau_1$ is no smaller than that for $\tau_2$.*

**Algorithm 6:** gMBC*

**Input:** A signed graph $G = (V, E^+, E^-)$
**Output:** A maximum balance clique for each $0 \leq \tau \leq \beta(G)$

```
Set τ as β(G) by invoking PF*;
while τ ≥ 0 do
    if τ = β(G) then g ← reduce G to (2τ − 1)-core;
    else g ← reduce G to |C^{τ+1}|-core;
    C^τ ← MBC*(g, τ);
    if C^τ = ∅ then C^τ ← C^{τ+1};
    τ ← τ − 1;
return {C^0, ..., C^{β(G)}};
```

Based on the result in Lemma 6, we propose to compute maximum balanced cliques for decreasing $\tau$ value order. Suppose we have obtained the maximum balanced clique $C^\tau$. Then we can use $C^\tau$ as the initial solution to problem for threshold $\tau-1$. In this way, the search space of computing the maximum balanced clique for $\tau-1$ will be greatly reduced, since we observe that in practice $|C^\tau| = |C^{\tau-1}|$ for many of the $\tau$ values. Note that, however, we need to first invoke PF* to get the polarization factor $\beta(G)$ and set the initial $\tau$ as $\beta(G)$. We denote this approach as gMBC*. The pseudocode of gMBC* is shown in Algorithm 6. It is immediate that the time complexity of gMBC* is $\mathcal{O}(\beta(G) \cdot n \cdot m \cdot 2^\delta)$ and the space complexity is $\mathcal{O}(m)$.

## VI. Experiments

In this section, we empirically evaluate the efficiency and effectiveness of our algorithms. For the maximum balanced clique problem, we implemented MBC (Algorithm 1) and MBC* (Algorithm 2). In addition, we also implemented another three variants of these two algorithms: MBC-noER which is MBC without EdgeReduction, MBC*-withER which is the variant of MBC* that also applies EdgeReduction at Line 1 of Algorithm 2, and MBC-Adv which is the improved version of MBC that applies the degree-based pruning and coloring-based upper bound by ignoring edge signs.

TABLE I
Statistics of Datasets

| Dataset | $\lvert V\rvert$ | $\lvert E\rvert$ | $\frac{\lvert E^-\rvert}{\lvert E\rvert}$ | $\lvert C^*\rvert$ | $\beta(G)$ | Category |
|---|---|---|---|---|---|---|
| Bitcoin | 5,881 | 21,492 | 0.15 | 11 | 5 | Trade |
| AdjWordNet | 16,259 | 76,845 | 0.32 | 60 | 28 | Language |
| Reddit | 54,075 | 220,151 | 0.08 | 8 | 3 | Social |
| Referendum | 10,884 | 251,406 | 0.05 | 19 | 5 | Political |
| Epinions | 131,828 | 711,210 | 0.17 | 15 | 6 | Social |
| WikiConflict | 116,717 | 2,026,646 | 0.63 | 6 | 3 | Editing |
| Amazon | 176,816 | 2,685,570 | 0.11 | 29 | 7 | Rating |
| BookCross | 63,535 | 3,890,104 | 0.07 | 550 | 118 | Rating |
| DBLP | 2,387,365 | 11,915,023 | 0.72 | 73 | 24 | Coauthor |
| Douban | 1,588,455 | 18,709,948 | 0.25 | 116 | 43 | Social |
| TripAdvisor | 145,315 | 20,569,277 | 0.14 | 1,916 | 201 | Rating |
| YahooSong | 1,000,990 | 30,139,524 | 0.18 | 127 | 21 | Rating |
| SN1 | 2,000,000 | 50,154,048 | 0.41 | 13 | 5 | Synthetic |
| SN2 | 2,000,000 | 111,573,268 | 0.39 | 19 | 7 | Synthetic |

For the polarization factor problem, we implemented PF-E (Section IV-A), PF-BS (Section IV-B) and PF* (Algorithm 4). In addition, we also implemented PF*-DOrder, which is the variant of PF* that replaces POrder with DOrder.

For the generalized maximum balanced clique problem, we implemented gMBC and gMBC*.

All the algorithms are implemented in C++. All the experiments are conducted on a machine with an Intel Core-i7 3.20GHz CPU and Ubuntu system. The time cost is measured as the amount of wall-clock time elapsed during the program's execution. Unless otherwise specified, experiments are conducted with polarization threshold $\tau = 3$ by default.

**Datasets.** We evaluate the algorithms on 14 datasets. Bitcoin, Reddit and Epinions are signed networks downloaded from SNAP [1]. AdjWordNet [2] captures the synonym-and-antonym relation among adjectives in the English language. Referendum and WikiConflict are signed networks used in [16] and are obtained from the authors of [16]. Amazon, BookCross, TripAdvisor and YahooSong are rating networks downloaded from KONECT [3]. We transform them into signed graphs as follows. For each pair of users, if they have enough number of close (resp. opposite) rating scores to a set of items, we assign a positive (resp. negative) edge between them. DBLP is the signed network used in [31], which is downloaded from the *dblp database* [4] and processed in the same way as [31], i.e., assign "+" to an edge $(u, v)$ if the number of papers co-authored by $u$ and $v$ is no less than 2, otherwise assign "−" to $(u, v)$. Douban is a signed network used in [7]. It is a movie-rating based social network in which an edge is negative if two users have very different movie preferences, and is positive otherwise. 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 [32]. Statistics of the graphs are shown in Table I, in which $\frac{|E^-|}{|E|}$ is the ratio of negative edges, $|C^*|$ is the maximum balanced clique size under polarization threshold

[1] https://snap.stanford.edu/
[2] https://wordnet.princeton.edu/
[3] http://konect.cc/networks/
[4] https://dblp.uni-trier.de/

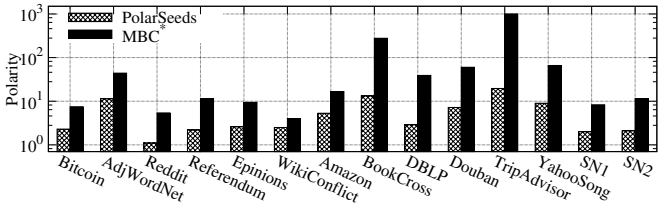



Fig. 5. Polarity (the larger, the better)

TABLE II
CASE STUDY ON REDDIT

| $C_L$ | $C_R$ |
|---|---|
| videos, gaming, mma, thepopcornstand, canada | subredditdrama, truesubredditdrama, drama |

$\tau = 3$, and $\beta(G)$ is the polarization factor.

### A. Effectiveness of Maximum Balanced Clique

**Compare with** PolarSeeds **[15].** We first compare the quality of maximum balanced clique computed by our algorithm MBC* with the polarized community computed by PolarSeeds [15]. As PolarSeeds has query vertices (i.e., seeds), we randomly pick 100 pairs of good seeds in the same way as [15] and report the average result quality. Specifically, vertices $u$ and $v$ are considered to be seeds if $(u, v) \in E^-$, $d_G^+(u) > t$ and $d_G^+(v) > t$, where $t$ is a pre-defined positive integer. The other parameters are set as their default values (i.e., $\epsilon = 10^{-3}$ and $\kappa = 0.9$). The result for the quality metric *Polarity* [15], [16] is shown in Figure 5. *Polarity* counts the number of edges that agree with the polarized structure and penalizes it by its size. That is, a polarized community of [15] is represented as $(C_1, C_2)$ which is similar to our $C_L$ and $C_R$, and $\text{Polarity}(C_1, C_2) = \frac{|E^+(C_1) \cup E^+(C_2)| + 2|E^-(C_1, C_2)|}{|C_1 \cup C_2|}$, where $E^+(C)$ means the set of positive edges in $C$ and $E^-(C_1, C_2)$ means the set of negative edges between $C_1$ and $C_2$. We can see that MBC* produces higher-quality results compared with PolarSeeds on all datasets. This is mainly because balanced cliques ensure that all edges agree with the polarized structure, and maximum balanced clique makes the Polarity even larger. Note that, besides Polarity, there are also other two quality metrics used in [15], i.e., signed bipartiteness ratio (SBR) and harmonic mean of cohesion and opposition measures (HAM). It can be proved that the HAM of balanced cliques is always 1, the highest possible HAM value. Thus, MBC* also outperforms PolarSeeds regarding HAM. In terms of SBR which penalizes the edges going outside $C_1 \cup C_2$, PolarSeeds performs better than MBC*, since MBC* does not penalize the edges going outside $C_L \cup C_R$.

**Case Studies.** We conduct case studies for the maximum balaned cliques on datasets Reddit and AdjWordNet. In Reddit, each vertex represents a subreddit, and positive and negative edges are created according to the sentiment between subreddits. The maximum balanced clique for $\tau = \beta(G) = 3$ as computed by MBC* is shown in Table II. Subreddits in $C_L$ usually interact with each other and share interesting posts about videos, games and pictures. All of them show negative sentiments to the subreddits drama in $C_R$. This shows that maximum balanced clique detects conflict groups in Reddit.

TABLE III
CASE STUDY ON ADJWORDNET

| $C_L$ | $C_R$ |
|---|---|
| good, better, best, wonderful, excellent, great, superior, awesome, brilliant, fabulous, fantastic, outstanding, perfect, superb, splendid, preeminent, exceptional, optimum, terrific.... | bad, worse, worst, terrible, poor, awful, inferior, unwell, horrendous, weak, horrifying, dreadful, despicable, disastrous, horrible, execrable, deplorable, abominable, horrific.... |

In AdjWordNet, each vertex represents an adjective, and a positive (resp. negative) edge indicates synonymous (resp. antonymous) relationship. The maximum balanced clique for $\tau = \beta(G) = 28$ has 60 vertices with $|C_L| = 28$ and $|C_R| = 32$, where a snippt is shown in Table III. We can see that words in the same group, i.e., $C_L$ or $C_R$, have similar meaning and words in different groups are mostly antonymous with each other. This case study shows that maximum balanced clique reveals interesting patterns in AdjWordNet.

We also run the MBCEnum algorithm of [13] to enumerate all maximal balanced cliques. For the above settings of $\tau$, MBCEnum reports 197 cliques for Reddit and 1 clique for AdjWordNet. For Reddit, this huge number of results may overwhelm end-users, and moreover, most of the enumerated cliques (i.e., 93%) are of size only 6 and they heavily overlap with each other. In contrast, our algorithm efficiently detects the largest balanced clique, which can be regarded as the dominating polarized group. For AdjWordNet, the sole clique enumerated by MBCEnum is the same as the one computed by our algorithm MBC*, as one would expect. On the other hand, MBC* runs significantly faster than MBCEnum, e.g., $50\times$ and $200\times$ faster on Reddit and AdjWordNet, respectively.

### B. Efficiency for the Maximum Balanced Clique Problem

**Against Enumeration-based Baseline Algorithm** MBC**.** We first evaluate our algorithm MBC* against the enumeration-based baseline algorithm MBC. The results on all datasets for $\tau = 3$ are reported in Figure 6. We can clearly see that MBC* significantly outperforms MBC on all datasets, and the improvement is up-to three orders of magnitude (e.g., on Douban). This is due to our strategy of transforming the maximum balanced clique problem to a series of maximum dichromatic clique problems which remove edge signs and sparsify the graph. In Figure 6, we also report the running time of MBC-noER and MBC*-withER. We can see that EdgeReduction improves the efficiency for MBC (compared with MBC-noER), but degrades the performance for MBC*-withER (compared with MBC*). This is because MBC is very slow and thus EdgeReduction can improve the efficiency by reducing the graph instance. In contrast, MBC* runs very fast, and thus EdgeReduction incurs a large overhead for MBC* due to having a high time complexity. Thus, we omit MBC-noER and MBC*-withER in the remaining testings.

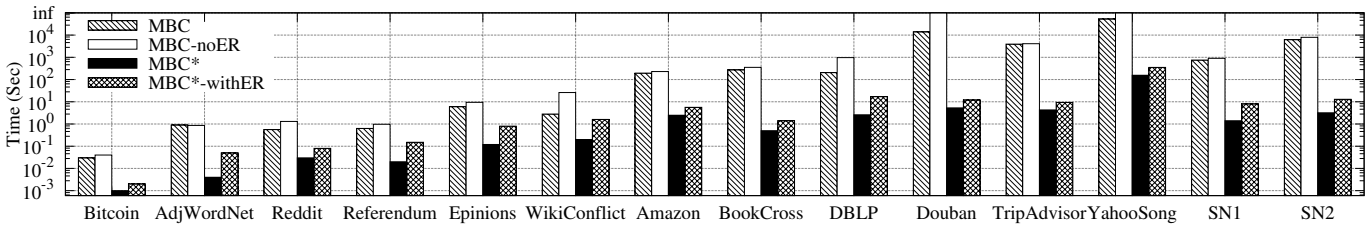
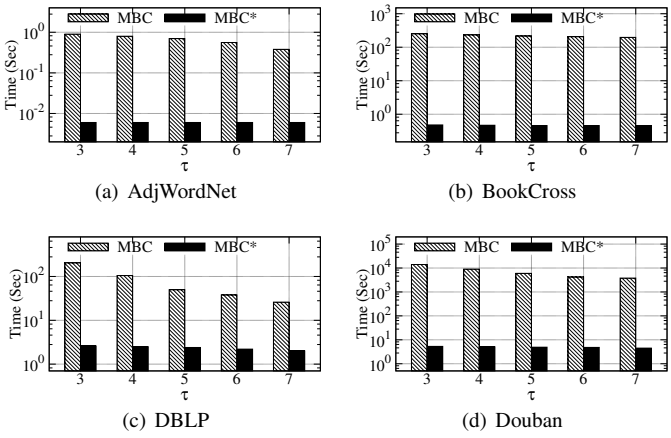



Fig. 6. Running time on all graphs for maximum balanced clique detection ($\tau = 3$)




(a) AdjWordNet (b) BookCross

(c) DBLP (d) Douban

Fig. 7. Varying polarization threshold $\tau$

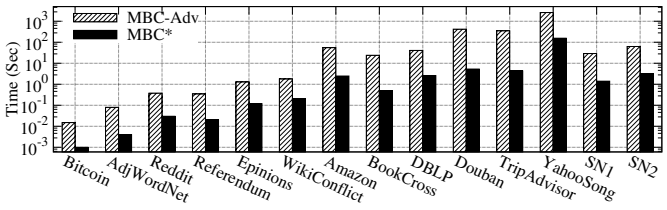



Fig. 8. Influence of our MDC transformation

The results of evaluating MBC* against MBC by varying $\tau$ from 3 to 7 are shown in Figure 7. MBC* consistently outperforms MBC for different $\tau$ values. We also observe that the running time of MBC decreases as $\tau$ increases, while that of MBC* is almost insensitive to $\tau$. This is because for larger $\tau$ values, the pruning power of EdgeReduction that is used in MBC strengthens. Nevertheless, MBC is still orders of magnitude slower than MBC* for $\tau = 7$.

**Evaluate the Influence of Our MDC Transformation.** In this experiment, we evaluate the influence of our MDC transformation — i.e., transform a maximum balanced clique problem to a series of maximum dichromatic clique problems — on the performance of our algorithm MBC*. We compare MBC* against MBC-Adv that does not apply the transformation but conducts the degree-based pruning and coloring-based bounding by simply discarding edge signs. From Figure 8, we can see that MBC* outperforms MBC-Adv by more than one order of magnitude. Thus, our MDC transformation improves the performance.

To get a deeper insight, we also evaluate the average size reduction brought by our MDC transformation. Recall that for each ego-network $G_u$, we first reduce it to $g_u$ by removing all conflicting edges, and then reduce $g_u$ to $g$ by degree-based pruning (Line 7 of Algorithm 2). We use SR1 to denote the average size reduction ratio after stage 1 (i.e., SR1 $= 1 - \frac{|E(g_u)|}{|E(G_u)|}$), and use SR2 to denote the average size reduction ratio after stage 2 (i.e., SR2 $= 1 - \frac{|E(g)|}{|E(G_u)|}$). The results of SR1 and SR2 are shown in the fourth and fifth columns of Table IV, respectively. We see that around $50\%$ edges are removed on average after stage 1, and almost $80\%$ edges are removed on average after the two stages. This demonstrates that the subgraph sizes are significantly reduced before passing to MDC. We also report the number of MDC instances that are generated by our algorithm MBC* in the third column of Table IV. We see that the number of MDC instances is very small compared with $n = |V|$. This confirms the superiority of our MDC transformation. Note that for Bitcoin, AdjWordNet and TripAdvisor, the number of MDC instances is 0. This is because the heuristic solution found by MBC-Heu is guaranteed to be optimal, and thus no MDC instance is generated.

TABLE IV
RUNNING STATISTICS OF MBC* AND PF* FOR $\tau = 3$ (SR1 AND SR2 ARE SIZE REDUCTION RATIOS, THE LARGER THE BETTER)

| Graphs | MBC* | | | | PF* | | | |
|---|---|---|---|---|---|---|---|---|
| | Heu | #MDC | SR1 | SR2 | Heu | #DCC | SR1 | SR2 |
| Bitcoin | 11 | 0 | - | - | 4 | 1 | 20% | 20% |
| AdjWordNet | 60 | 0 | - | - | 28 | 0 | - | - |
| Reddit | 6 | 96 | 41% | 86% | 2 | 14 | 40% | 92% |
| Referendum | 19 | 20 | 6% | 31% | 3 | 2 | 22% | 22% |
| Epinions | 11 | 82 | 30% | 70% | 5 | 10 | 35% | 66% |
| WikiConflict | 0 | 43 | 48% | 94% | 2 | 1 | 54% | 80% |
| Amazon | 15 | 952 | 64% | 81% | 6 | 825 | 73% | 89% |
| BookCross | 150 | 3 | 61% | 85% | 4 | 150 | 56% | 90% |
| DBLP | 31 | 26 | 49% | 80% | 2 | 23 | 51% | 82% |
| Douban | 83 | 10 | 73% | 98% | 27 | 6 | 72% | 95% |
| TripAdvisor | 1916 | 0 | - | - | 138 | 63 | 39% | 49% |
| YahooSong | 36 | 575 | 23% | 31% | 7 | 563 | 29% | 39% |
| SN1 | 10 | 19 | 45% | 90% | 4 | 16 | 49% | 92% |
| SN2 | 12 | 28 | 42% | 85% | 5 | 23 | 46% | 89% |

**Scalability Testing.** We now test the scalability of MBC, MBC-Adv and MBC* on two large datasets DBLP and Douban. We randomly sample vertices from $20\%$ to $100\%$ in the original graph. For each sampled vertex set, we obtain the induced subgraph of the vertex set as the input data. As shown in Figure 10, the running time of all algorithms increases as the graph size increases. This is because a larger graph means a larger search space and thus a higher running time. Nevertheless, MBC* outperforms MBC and MBC-Adv in all the settings and scales better to the graph size.

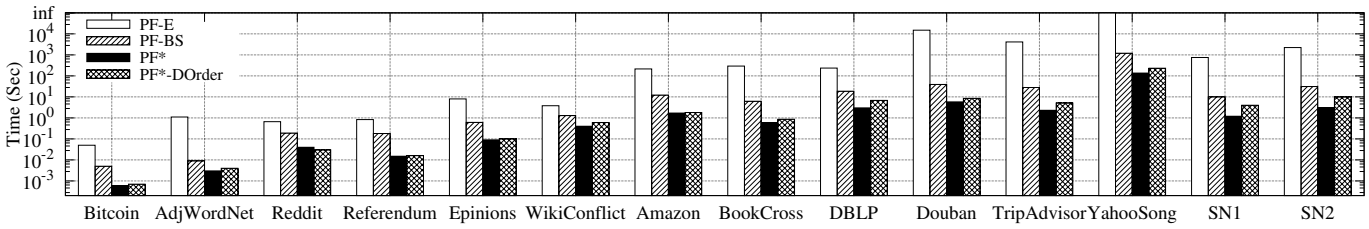
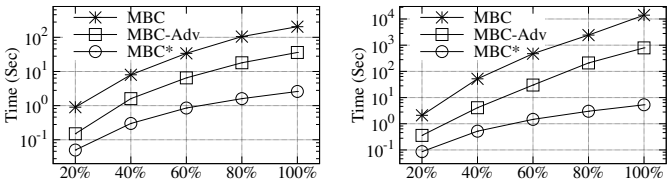
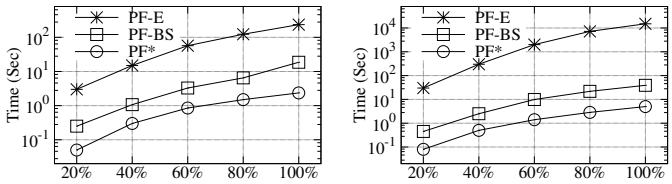



Fig. 9. Running time on all graphs for polarization factor




(a) DBLP (b) Douban

Fig. 10. Scalability testing ($\tau = 3$, vary graph size)




(a) DBLP (b) Douban

Fig. 12. Scalability testing (vary graph size)

**Evaluate Our Heuristic Algorithm.** We now evaluate our heuristic algorithm MBC-Heu. The size of the balanced clique found by MBC-Heu, which is used as the initial clique in MBC*, is shown in the second column of Table IV. On Bitcoin, AdjWordNet, Referendum and TripAdvisor, MBC-Heu finds the optimal solution. On some of the graphs, the balanced clique found by MBC-Heu is far from optimal. For example, on BookCross, the balanced clique found by MBC-Heu is of size 150 while the maximum balanced clique size is 550. In the extreme case, MBC-Heu fails to find a valid initial solution on WikiConflict due to the constraint $\tau$.

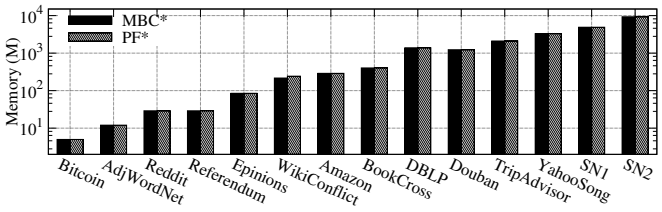



Fig. 11. Memory consumption of MBC* and PF*

**Memory Consumption.** The memory consumption of MBC* on all the datasets is shown in Figure 11, which is "the maximum resident set size of the process during its lifetime" as measured by the Linux command /usr/bin/time[5]. We can see that the memory consumption is small and is almost linear to the number of edges, conforming with our theoretical analysis.

### C. Efficiency for the Polarization Factor Problem

**Against PF-E, PF-BS and PF*-DOrder.** Figure 9 reports the running time of PF-E, PF-BS, PF*-DOrder and PF* on all the graphs. We can see that PF* outperforms the enumeration-based baseline algorithm PF-E by several orders of magnitude on all the graphs. For example, on Douban, PF* computes the polarization factor in 5 seconds, while PF-E takes more than 4 hours. Our binary search-based algorithm PF-BS is significantly faster than PF-E, due to our efficient algorithm MBC* for verifying $\tau$. However, as PF-BS needs to invoke MBC* multiple times, it is almost one order of magnitude slower than PF*. This demonstrates the superiority of PF* that transforms the polarization factor problem to a series of dichromatic clique checking problems.

In Figure 9, we also report the running time of PF*-DOrder, which is the variant of PF* that uses the degeneracy ordering DOrder instead of the polarization ordering. We see that PF*-DOrder is slower than PF*. This demonstrates the superiority of the polarization ordering for computing the polarization factor.

**Evaluate the Heuristic Algorithm and DCC Transformation.** In this experiment, we report the running statistics of PF*. The initial lower bound of the polarization factor $\beta(G)$ computed by MBC-Heu is shown in the sixth column of Table IV. We see that MBC-Heu obtains the exact value of $\beta(G)$ on AdjWordNet. On some of the graphs, the initial lower bound of $\beta(G)$ is much smaller than the exact value. For example, on BookCross, the initial lower bound is 4 while the exact $\beta(G)$ is 118.

The number of DCC instances generated by PF* and the average instance size reduction ratio are shown in last three columns of Table IV, which exhibit similar phenomenon as MBC*. That is, the number of DCC instances is very small compared with $n = |V|$, and the transformation leads to a significant reduction on each dichromatic-network. This confirms the superiority of our DCC transformation.

**Scalability Testing.** In this experiment, we test the scalability of PF-E, PF-BS and PF* on DBLP and Douban by varying their vertices from 20% to 100%. As shown in Figure 12, when the graph scales up, the processing time of all algorithms increases as well because of the increasing search space. However, PF* outperforms other algorithms in all cases and scales well to the graph size.

**Memory Consumption.** The memory consumption of PF*

[5]http://man7.org/linux/man-pages/man1/time.1.html

TABLE V
DISTINCT NUMBER OF MAXIMUM BALANCED CLIQUES FOR DIFFERENT $\tau$ VALUES (I.E., $|\mathbb{C}| = |\{C^0, C^1, \ldots, C^{\beta(G)}\}|$)

| Graph | $\lvert\mathbb{C}\rvert$ | Size Range of the Cliques |
|---|---|---|
| Bitcoin | 2 | $10_{\langle 5\rvert 5\rangle} \rightarrow 11_{\langle 4\rvert 7\rangle}$ |
| AdjWordNet | 1 | $60_{\langle 28\rvert 32\rangle} \rightarrow 60_{\langle 28\rvert 32\rangle}$ |
| Reddit | 4 | $8_{\langle 3\rvert 5\rangle} \rightarrow 17_{\langle 0\rvert 17\rangle}$ |
| Referendum | 6 | $17_{\langle 5\rvert 12\rangle} \rightarrow 35_{\langle 0\rvert 35\rangle}$ |
| Epinions | 7 | $12_{\langle 6\rvert 6\rangle} \rightarrow 93_{\langle 0\rvert 93\rangle}$ |
| WikiConflict | 4 | $6_{\langle 3\rvert 3\rangle} \rightarrow 16_{\langle 0\rvert 16\rangle}$ |
| Amazon | 8 | $15_{\langle 7\rvert 8\rangle} \rightarrow 42_{\langle 0\rvert 42\rangle}$ |
| BookCross | 39 | $240_{\langle 118\rvert 122\rangle} \rightarrow 614_{\langle 1\rvert 613\rangle}$ |
| DBLP | 7 | $49_{\langle 24\rvert 25\rangle} \rightarrow 247_{\langle 1\rvert 246\rangle}$ |
| Douban | 13 | $88_{\langle 43\rvert 45\rangle} \rightarrow 139_{\langle 0\rvert 139\rangle}$ |
| TripAdvisor | 14 | $448_{\langle 201\rvert 247\rangle} \rightarrow 1916_{\langle 45\rvert 1871\rangle}$ |
| YahooSong | 10 | $43_{\langle 21\rvert 22\rangle} \rightarrow 353_{\langle 0\rvert 353\rangle}$ |
| SN1 | 6 | $10_{\langle 5\rvert 5\rangle} \rightarrow 19_{\langle 0\rvert 19\rangle}$ |
| SN2 | 8 | $15_{\langle 7\rvert 8\rangle} \rightarrow 24_{\langle 0\rvert 24\rangle}$ |

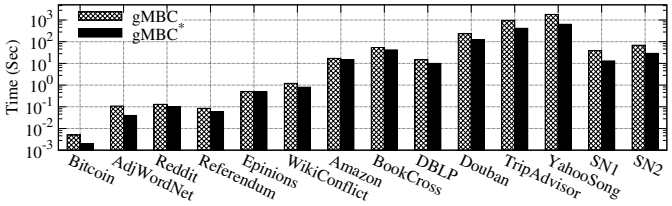



Fig. 13. Running time of gMBC and gMBC* on all graphs

on all the datasets is shown in Figure 11. We can see that the memory consumption of PF* is almost the same as that of MBC*. Thus, the memory consumption of PF* is also small and is linear to the number of edges.

### D. The Generalized Maximum Balanced Clique Problem

In this subsection, we evaluate the algorithms for computing a maximum balanced clique for each $0 \leq \tau \leq \beta(G)$. Firstly, we observe that the number of distinct maximum balanced clique in $\{C^0, C^1, \ldots, C^{\beta(G)}\}$, where $C^i$ is the maximum balanced clique for constraint $\tau = i$, is much smaller than $\beta(G) + 1$. For example, the number is 39 for BookCross which has $\beta(G) = 118$, and the number is 14 for TripAdvisor which has $\beta(G) = 201$; the number of such cliques for all datasets is shown in the second column of Table V, denoted by $|\mathbb{C}|$. This is because the maximum balanced clique is the same for many of the different $\tau$ values. In the third column of Table V, we also report the size of the maximum balanced clique for $\tau = \beta(G)$ and that for $\tau = 0$; the two numbers in the subscripts represent the sizes of the two sides of the clique, i.e., $|C_L|$ and $|C_R|$. For example, for the dataset BookCross, the maximum balanced clique for $\tau = \beta(G) = 118$ has 240 vertices with $|C_L| = 118$ and $|C_R| = 122$, and the maximum balanced clique for $\tau = 0$ has 614 vertices with $|C_L| = 1$ and $|C_R| = 613$. We can see that $C^0$ is highly skewed while $C^{\beta(G)}$ is well balanced, in terms of the sizes of $C_L$ and $C_R$. As the number of distinct maximum balanced cliques for all possible values of the threshold $\tau$ is not large, end-users can easily go through all these maximum balanced cliques to find the ones that suit their needs.

The running time of algorithms gMBC and gMBC* on all the datasets is shown in Figure 13. Recall that both algorithms invoke MBC*. The increasing factor of gMBC and gMBC* over MBC* is generally related to the polarization factor $\beta(G)$ of the graph, since the larger $\beta(G)$, the more rounds they need to invoke MBC*. Nevertheless, gMBC* consistently runs faster than gMBC due to the computation sharing in the former.

## VII. RELATED WORKS

Signed graph was firstly studied by Harary et al. [6], where the notion of structural balanced 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 in [8], [33]. 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 in [33]. Heuristic algorithms without any guarantee on the solution optimality are investigated in [8], [33]. As the maximum balanced subgraph problem does not require the identified subgraph to be complete (i.e., clique), these techniques cannot be applied to our problem. Also note that the identified subgraph may violate the structural balanced constraint after adding the missing edges, if we consider those edges currently not in the signed graph to be missing edges. In contrast, balanced cliques that are studied in this paper have the benefit that *they are guaranteed to be structural balanced even after adding back the edges that are currently missing (i.e., absent) in the signed graph*. Moreover, despite that balanced clique is more restrictive than balanced subgraph, our empirical study shows that it is not uncommon to find large balanced cliques in real signed graphs.

The notion of (structural) balanced clique is formulated in [13], where the problem of enumerating all maximal balanced cliques is studied. We adapt the enumeration algorithm proposed in [13] for our problems, and show in our experiments that this is inefficient and the adapted algorithms are outperformed by our algorithms MBC* and PF* by several orders of magnitude. Hao et al. [34] introduced the notion of *$k$-balanced trusted clique* for signed graphs, which is a clique with $k$ vertices such that all its edges are positive edges. This is essentially the same problem as the traditional $k$-clique problem over unsigned graphs, i.e., by removing all negative edges. Thus, the techniques proposed in [34] cannot be applied to solve our problems. The notion of $(\alpha, k)$-clique is defined in [31] 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 in [31] cannot be applied to our problem.

On the other hand, there is a huge literature on the classic *maximum clique problem* in unsigned graphs, e.g., [25]–[27], [35]. Advanced algorithmic techniques have been developed for the classic maximum clique problem, such as graph recoloring [26] and incremental MaxSAT reasoning [35]. However, these techniques cannot be directly applied to our problem due to the existence of edge signs and the enforcement of structural balanced constraint.

## VIII. Conclusion

In this paper, we first studied the maximum balanced clique problem in signed graphs. We proposed techniques to transform the maximum balanced clique problem over $G$ to a series of maximum dichromatic clique problems over small subgraphs of $G$. The transformation not only removes edge signs (and thus the structural balanced constraint) but also sparsifies the edge set. In addition, we also extended our techniques to the polarization factor problem, and to the problem of reporting a maximum balanced clique for each $\tau \geq 0$. Experimental results on real datasets demonstrated the efficiency, effectiveness and scalability of our algorithms.

## 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 *Proceedings of the 18th International Conference on World Wide Web*, pages 741–750, 2009.

[4] Christos Giatsidis, Bogdan Cautis, Silviu Maniu, Dimitrios M Thilikos, and Michalis Vazirgiannis. Quantifying trust dynamics in signed graphs, the s-cores approach. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 668–676. SIAM, 2014.

[5] 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.

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

[7] Lingyang Chu, Zhefeng Wang, Jian Pei, Jiannan Wang, Zijin Zhao, and Enhong Chen. Finding gangs in war from signed networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1505–1514, 2016.

[8] 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.

[9] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650, 2010.

[10] Jihang Ye, Hong Cheng, Zhe Zhu, and Minghua Chen. Predicting positive and negative links in signed social networks by transfer learning. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1477–1488, 2013.

[11] 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.

[12] 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.

[13] 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.

[14] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. Community interaction and conflict on the web. In *Proceedings of the 2018 world wide web conference*, pages 933–943, 2018.

[15] 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.

[16] Francesco Bonchi, Edoardo Galimberti, Aristides Gionis, Bruno Ordozgoiti, and Giancarlo Ruffo. Discovering polarized communities in signed networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 961–970, 2019.

[17] Apichat Suratanee, Martin H Schaefer, Matthew J Betts, Zita Soons, Heiko Mannsperger, Nathalie Harder, Marcus Oswald, Markus Gipp, Ellen Ramminger, Guillermo Marcus, et al. Characterizing protein interactions employing a genome-wide sirna cellular phenotyping screen. *PLoS Comput Biol*, 10(9):e1003814, 2014.

[18] Soorin Yim, Hasun Yu, Dongjin Jang, and Doheon Lee. Annotating activation/inhibition relationships to protein-protein interactions using gene ontology relations. *BMC systems biology*, 12(1):9, 2018.

[19] Arunachalam Vinayagam, Jonathan Zirin, Charles Roesel, Yanhui Hu, Bahar Yilmazel, Anastasia A Samsonova, Ralph A Neumüller, Stephanie E Mohr, and Norbert Perrimon. Integrating protein-protein interaction networks with phenotypes reveals signs of interactions. *Nature methods*, 11(1):94–99, 2014.

[20] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[21] Adit Krishnan, P Deepak, Sayan Ranu, and Sameep Mehta. Leveraging semantic resources in diversified query expansion. *World Wide Web*, 21(4):1041–1067, 2018.

[22] Vishwajeet Kumar, Nitish Joshi, Arijit Mukherjee, Ganesh Ramakrishnan, and Preethi Jyothi. Cross-lingual training for automatic question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4863–4872, 2019.

[23] full version: https://lijunchang.github.io/pdf/msbc-tr.pdf.

[24] Coenraad Bron and Joep Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.

[25] Evgeny Maslov, Mikhail Batsyn, and Panos M Pardalos. Speeding up branch and bound algorithms for solving the maximum clique problem. *Journal of Global Optimization*, 59(1):1–21, 2014.

[26] Etsuji Tomita. Efficient algorithms for finding maximum and maximal cliques and their applications. In *International Workshop on Algorithms and Computation*, pages 3–15. Springer, 2017.

[27] Lijun Chang. Efficient maximum clique computation over large sparse graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 529–538, 2019.

[28] Richard M. Karp. Reducibility among combinatorial problems. In *Proc. of CCC'72*, pages 85–103, 1972.

[29] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983.

[30] David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM J. Exp. Algorithmics*, 18, 2013.

[31] 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 *34th IEEE International Conference on Data Engineering*, pages 245–256, 2018.

[32] 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):1–12, 2017.

[33] 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.

[34] 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.

[35] Chu-Min Li, Zhiwen Fang, and Ke Xu. Combining maxsat reasoning and incremental upper bound for the maximum clique problem. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 939–946. IEEE, 2013.


## Appendix A. Proofs of Lemmas and Theorems

**Proof of Theorem 1.** We prove the NP-hardness of both problems by reducing from the classic maximum clique problem over unsigned graphs which is NP-hard [28]. Given an unsigned graph instance $G = (V, E)$ of the maximum clique problem and any threshold $\tau \leq |V|$, we construct the signed graph instance $G^s$ for the maximum balanced clique problem as follows. We first set $G^s$ as $G$ with all edges being positive edges. Then, we add into $G^s$ a complete graph with $\tau$ vertices $\{u_1, \ldots, u_\tau\}$ where all edges are positive edges. Finally, we add into $G^s$ a negative edge between each vertex

of $\{u_1, \ldots, u_\tau\}$ and each vertex of $V$. It is easy to verify that $G$ has a clique of size at least $\tau$ if and only if $G^s$ has a balanced clique satisfying the polarization constraint $\tau$. Thus, both the maximum balanced clique problem and the polarization factor problem are NP-hard. □

**Proof of Theorem 2.** Let $\omega(G, \tau)$ be the size of the maximum balanced clique in $G$ that satisfies the polarization constraint $\tau$. First, for any dichromatic clique $C$ in the dichromatic network $g_u$ for $u \in V$, $C$ is a balanced clique in $G$ satisfying the polarization constraint $\tau$. This is because, let $C_L = C \cap V_L(g_u)$ and $C_R = C \cap V_R(g_u)$, then all edges between $C_L$ and all edges between $C_R$ in $g_u$ correspond to positive edges in the ego-network $G_u$, and all edges between $C_L$ and $C_R$ in $g_u$ correspond to negative edges in $G_u$. Thus, $C_L \cup C_R$ is a balanced clique in $G_u$ and thus in $G$, and $\omega(G, \tau) \geq \max_{u \in V} \delta(g_u, \tau)$. Second, let $C^*$ be a maximum balanced clique in $G$ satisfying the polarization constraint $\tau$, and let $u^*$ be the lowest-ranked vertex in $C^*$. Then $C^*$ is a dichromatic clique in $g_{u^*}$, as $C^*_L$ (that contains $u^*$) are L-vertices and $C^*_R$ are R-vertices. Thus, $\omega(G, \tau) \leq \max_{u \in V} \delta(g_u, \tau)$, and the lemma holds. □

**Proof of Theorem 3.** Firstly, each of Lines 1–4 runs in $\mathcal{O}(m)$ time. Specifically, VertexReduction, degree-based pruning and degeneracy ordering can all be computed in $\mathcal{O}(m)$ time. The time complexity of MBC-Heu (i.e., Line 2) is also $\mathcal{O}(m)$, which will be discussed in Section III-C. Secondly, the for loop at Line 5 processes each vertex $u$, by constructing the dichromatic network $g_u$ of $u$ and then computing the maximum dichromatic clique in $g_u$. The maximum dichromatic clique in $g_u$ is computed by invoking MDC, which then recursively invokes itself (Line 21). Note that, the number of vertices in $g_u$ is bounded by $\delta$ [27]. Thus, for a fixed dichromatic network $g_u$, the total number of invocations to MDC is at most $2^\delta$, since each invocation to MDC has a different input $C$. It is easy to see that each invocation to MDC (excluding Line 21) takes $\mathcal{O}(|E(g)|) = \mathcal{O}(m)$ time; note that, the coloring-based upper bound can be computed in linear time to the number of edges. Therefore, the time complexity of Algorithm 2 is $\mathcal{O}(n \cdot m \cdot 2^\delta)$.

For the space complexity, the input graph $G$ takes $\mathcal{O}(m)$ space and the dichromatic network $g_u$ takes $\mathcal{O}(m)$ space. Note that, although MDC takes a graph $g$ as input, we do not store a separate copy of $g$ for each invocation to MDC. Instead, we only store $g_u$ in the main memory and modify $g_u$ before going to the recursion of Line 21, and after returning from the recursion, we restore the graph $g_u$, in the same way as [30]. Moreover, after processing a dichromatic network, we release its memory, and thus there will be at most one dichromatic network stored in the main memory through out the execution of Algorithm 2. Thus, the space complexity is $\mathcal{O}(m)$. □

**Proof of Theorem 4.** Line 1 takes $\mathcal{O}(m)$ time to obtain the maximum-degree vertex $u$. Line 2 takes $\mathcal{O}(m)$ time to construct the dichromatic network $g_u$. The while loop is the most time-critical. To efficiently obtain the maximum-degree vertex $v$ in the while loop (i.e., Lines 6,7), we maintain the degree in $g$ for each vertex of $g$. Without loss of generality, assume $v_1, \ldots, v_l$ are the maximum-degree vertices that are iteratively obtained in the while loop. It is easy to see that the total time complexity of Lines 6,7 for the entire execution of while loop is $\mathcal{O}(n + \sum_{i=1}^{l} d_G(v_i)) = \mathcal{O}(m)$; note that the number of vertices in $g$ is at most $d_G(v_i)$ after $v_i$ is added to $C$, see Line 8. Also, the total time complexity of maintaining the degree information for vertices of $g$ is $\mathcal{O}(m)$, as each vertex is removed from $g$ at most once at Line 8. As a result, the time complexity of Algorithm 3 is $\mathcal{O}(m)$. For the space complexity, it is immediate that it is $\mathcal{O}(m)$, as we iteratively modify $g$; that is, there is only one copy of $g$ during the execution. □

**Proof of Lemma 3.** This lemma can be proved in a similar way to the proof of Theorem 2. We omit the details. □

**Proof of Lemma 4.** Let $C$ be a dichromatic clique in $g_{v_i} = (V_L, V_R, E)$ satisfying $|C \cap V_L(g_{v_i})| \geq \gamma(g_{v_i})$ and $|C \cap V_R(g_{v_i})| \geq \gamma(g_{v_i})$. Let $C' = C \setminus \{u\}$, and suppose $v_j$ is the lowest-ranked vertex in $C'$. It is easy to verify that $j > i$ and $C'$ is a dichromatic clique in $g_{v_j}$ satisfying $|C' \cap V_L(g_{v_j})| \geq \gamma(g_{v_i}) - 1$ and $|C' \cap V_R(g_{v_j})| \geq \gamma(g_{v_i}) - 1$. Thus, the lemma holds. □

**Proof of Theorem 5.** The time and space complexity of Algorithm 4 follows in the same way as that of Algorithm 2, if we use the degeneracy ordering at Line 3 of Algorithm 4. Note that, although we do not explicitly pass the growing clique $C$ as the input to DCC, such a growing clique $C$ is implicitly maintained (i.e., by adding $v$ obtained at Line 16 of Algorithm 4). Thus, the total number of invocations to DCC for each fixed dichromatic network $g_u$ is bounded by $2^\delta$. □

**Proof of Lemma 5.** Recall that $g_u$ is the dichromatic-network of $u$ as defined in Section III-B, and $\gamma(g_u)$ is the largest $\tau$ such that $g_u$ has a dichromatic clique satisfying the constraint $\tau$. Let $C$ be a dichromatic clique in $g_u$ satisfying the constraint $\gamma(g_u)$. Then, for every vertex $v \in C$, we have $\min\{d^+_C(v) + 1, g^-_C(v)\} \geq \gamma(g_u)$; moreover, this also holds for $\{u\} \cup C$. Consequently, $pn(u) \geq \gamma(g_u)$, and the lemma holds. □

**Proof of Lemma 6.** Let $C^{\tau_2}$ be the maximum balanced clique for threshold $\tau_2$. It is obvious that $C^{\tau_2}$ also satisfies the threshold $\tau_1$ since $\tau_1 < \tau_2$. Consequently, the maximum balanced clique for threshold $\tau_1$ will be no smaller than $C^{\tau_2}$. □