Identifying Large Structural Balanced Cliques in Signed Graphs

Kai Yao, Lijun Chang, and Lu Qin

***Abstract*—Signed graphs have been used to capture the polarity of relationships through positive/negative edge signs. In this paper, we consider balanced cliques — a clique is balanced if its vertex set C can be partitioned into C_L and C_R such that all negative edges are between C_L and C_R — and study the problems of maximum balanced clique computation and large balanced clique enumeration. Our main idea is a novel graph reduction that transforms a balanced clique problem over a signed graph G to problems over small subgraphs of G. Specifically, for each vertex u in G, we extract the subgraph G_u of G induced by $V_L \cup V_R$; V_L is u and u's positive neighbors while V_R is u's negative neighbors. Then, we remove from G_u all positive edges between V_L and V_R and all negative edges between vertices of the same set; denote the resulting graph of discarding edge signs as g_u. We show that all balanced cliques containing u in G can be found by processing g_u. Due to the small size and no edge signs, large cliques containing u in g_u can be efficiently identified. Experimental results on real signed graphs demonstrated the advantages of our techniques.**

***Index Terms*—Graph algorithms, signed graphs, structural balanced cliques.**

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]. In the literature, many traditional graph analysis tasks have been extended to signed graphs by treating positive and negative edges differently. For example, Derr et al. [6] conducted a comprehensive investigation on relevance measurements for signed graphs. Rahaman and Hosein [7] extended the DeGrootian opinion dynamics, which was originally developed for unsigned graphs, to signed graphs aiming to minimize polarization under a given budget. Rizi and Granitzer [8] extended classic graph embedding algorithms to signed heterogeneous graphs. Besides, community detection [9], [10], link prediction [11], [12] and recommendation systems [13], [14] have also been formulated and studied for signed graphs.

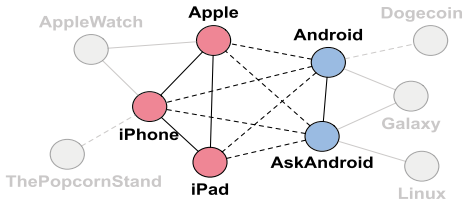


Fig. 1. A toy signed graph of Reddit.

One prominent and fundamental theory in signed graph analysis is the *structural balance theory* [15], 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". The problem of enumerating all maximal structural balanced cliques in a signed graph is recently formulated and studied by Chen et al. [16]. 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 Fig. 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, of varying sizes. 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 partially remedy this, Chen et al. [16] introduced a threshold τ 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, and the efficiency of the algorithms in [16] is not satisfactory.

Motivated by this, we in this paper investigate both the *maximum* balanced clique computation problem and the *large* balanced clique enumeration problem. Given a signed graph

Manuscript received 13 June 2022; revised 3 May 2023; accepted 2 July 2023. Date of publication 17 July 2023; date of current version 6 February 2024. The work of Lijun Chang was supported by ARC under Grants FT180100256 and ARC DP220103731. The work of Lu Qin was supported by ARC under Grants FT200100787 and ARC DP210101347. Recommended for acceptance by B. Glavic. *(Corresponding author: Kai Yao.)*

Kai Yao and Lijun Chang are with the School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: kyao8420@uni.sydney.edu.au; Lijun.Chang@sydney.edu.au).

Lu Qin is with the University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: lu.qin@uts.edu.au).

Digital Object Identifier 10.1109/TKDE.2023.3295803

$G = (V, E^+, E^-)$ and a threshold τ, the maximum balanced clique computation problem aims to find the largest balanced clique C^* in G that satisfies $|C_L^*| \geq \tau$ and $|C_R^*| \geq \tau$. Let $\omega_\tau(G)$ be the size of C^*, i.e., $\omega_\tau(G) = |C^*|$. Given G and thresholds τ and α, the large balanced clique enumeration problem aims to enumerate all maximal balanced cliques $C \subseteq V$ that satisfy $|C_L| \geq \tau$, $|C_R| \geq \tau$ and $|C| \geq \omega_\tau(G) - \alpha$. Our motivation of introducing α into the large balanced clique enumeration problem is to make a trade-off between the size and the number of identified balanced cliques, and also a trade-off between the processing time and the number of identified large balanced cliques, by tuning α. Specifically, the larger the value of α, the smaller the size and the more the number of reported results and also the longer the processing time. For example, by setting $\alpha = 0$, only *maximum* balanced cliques are reported, while for $\alpha = |V|$, all maximal balanced cliques will be reported; the processing time for $\alpha = 0$ could be orders of magnitude smaller than that for $\alpha = |V|$.

The counterpart of balanced cliques in signed graphs is cliques in unsigned graphs. However, unlike clique detection in unsigned graphs that identifies a set of closely connected groups, balanced clique detection in signed graphs aims to capture the polarized connection between groups where entities in the graph exhibit both harmonious and opposing relationships, such as friend-foe relationships in social networks, support-dissent opinions in opinion networks and activation-inhibition relations in protein-protein interaction networks. By detecting balanced cliques in these signed graphs, we can gain new insights into the relationships among entities and the graph structure. Various applications of balanced clique detection are introduced 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 [17], [18], [19]. Users in a large balanced clique are actively involved in conflicting groups, who have clear and firm standpoints on each other. Thus, they may represent 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 [20], [21]. As argued in [5], [22], 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 [23]. Thus, we can use large balanced cliques to identify significant synonym groups that are antonymous with each other, which can be further used in applications such as semantic expansion [24] and automatic question generation [25].

The state-of-the-art algorithm for enumerating all maximal balanced cliques in a signed graph is MBCEnum [16]. It is adapted from the classic BK algorithm designed for enumerating all maximal cliques in an unsigned graph [26]. Due to the existence of edge signs, MBCEnum iteratively builds up two sets C_L and C_R such that $C_L \cup C_R$ is a balanced clique, and maintains two candidate sets P_L and P_R of vertices that are used to grow C_L and C_R, respectively. Specifically, following the structural balance theory, 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. Recall that BK only needs to maintain one solution set C and one candidate set P. Thus, MBCEnum is more complicated than BK. Moreover, the efficiency of MBCEnum is not satisfactory for computing large balanced cliques, due to the lack of advanced pruning and bounding techniques that have been shown to be crucial to the practical efficiency of maximum clique computation on unsigned graphs [27], [28], [29], [30]. One possible way to utilize these techniques for signed graphs is discarding edge signs (and thus the structural balance constraint) when conducting pruning and bounding. This is correct but ineffective as verified by our experiments: (1) the structural balance constraint is not considered due to ignoring edge signs, and (2) the number of edges is abundant which makes the pruning and bounding ineffective.

In this paper, we first propose an efficient branch-and-bound algorithm MBC* to compute a maximum balanced clique C^*. Our main idea is a novel graph reduction technique that transforms the maximum balanced clique problem over a signed graph G to a series of problems over small subgraphs of G. Specifically, for each vertex u in G, we extract the subgraph G_u of G induced by $V_L \cup V_R$ 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 positive edges between V_L and V_R and all negative edges between vertices of the same set; denote the resulting graph of discarding edge signs as g_u. We prove that the maximum balanced clique containing u in G is the same as the maximum clique in g_u that contains u and at least τ vertices from each of V_L and V_R. Due to the small size and no edge signs, g_u can be solved efficiently by exploiting the existing pruning and bounding techniques that are designed for the maximum clique problem on unsigned graphs.

We then propose a two-stage approach BCE* for enumerating large maximal balanced cliques. We first compute the maximum balanced clique size $\omega_\tau(G)$ by invoking MBC* in Stage-I, and then enumerate all large maximal balanced cliques by using the size threshold $\lambda = \max\{\omega_\tau(G) - \alpha, 2\tau\}$ in Stage-II. Our enumeration algorithm follows a similar idea to MBC*, i.e., we transform the enumeration problem over G into a series of enumeration problems over g_u for each u in G. It is worth mentioning that when $\alpha \geq \omega_\tau(G) - 2\tau$, our problem becomes the same as the maximal balanced clique enumeration problem studied by Chen et al. [16]. Our empirical studies show that BCE* is up to two orders of magnitude faster than the algorithm of [16] for this special case, i.e., for large α values. Note also that, our algorithm BCE* will run faster when α become smaller, while the algorithm of [16] does not consider α.

Our main contributions are summarized as follows.

- We propose an efficient branch-and-bound algorithm MBC* to solve the maximum balanced clique computation 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 problems over small subgraphs of G, which not only removes edge signs but also sparsifies the edge set.
- We formulate the large balanced clique enumeration problem for a relative size threshold α, which captures the maximal balanced clique enumeration problem studied by Chen et al. [16] as a special case. We also propose an efficient algorithm BCE* to enumerate all large maximal balanced cliques.
- We conduct extensive empirical studies on large real signed graphs to demonstrate the effectiveness of our models and the efficiency of our algorithms. In particular, for the problem of enumerating all maximal balanced cliques that is studied by Chen et al. [16], our algorithm BCE* is up to two orders of magnitude faster than the algorithm proposed by Chen et al. [16].

A preliminary version of this work has been published in [31]. Our contributions related to the large balanced clique enumeration problem are new, compared with [31].

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 computation problem, and Section IV studies the large balanced clique enumeration problem. Experimental results are reported in Section V, and related works are discussed in Section VI. Finally, Section VII concludes the paper.

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 its set of *positive neighbors* and $N_G^-(v) = \{u \mid (v, u) \in E^-\}$ be its set of *negative neighbors*. 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., $G[S] = (S, \{(u, v) \in E \mid u \in S, v \in S\})$. Frequently used notations are summarized in Table I.

Definition 1 (Structural Balanced Group [15]): 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.

TABLE I
Frequently Used Notations

Notation	Definition
$G = (V, E)$	an undirected and signed graph
E^+, E^-	positive/negative edges
n, m	number of vertices and edges in G
$N(v)$	set of neighbors of v
$N^+(v)$, $N^-(v)$	set of only positive/negative neighbors of v
$d(v)$	degree of v
$d^+(v)$, $d^-(v)$	positive/negative degree of v
τ	threshold used to restrict the two sides of a balanced clique
ω_τ	size of the maximum balanced clique under threshold τ
α	threshold used to restrict the maximal balanced clique size
C^*	the maximum balanced clique
G_v	ego-network of v
g_v	dichromatic-network of v

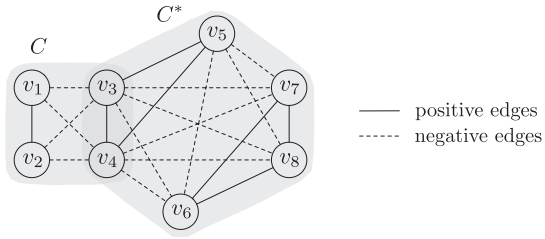


Fig. 2. A toy signed graph.

Definition 2 (Structural Balanced Clique [16]): 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 Fig. 2. $C = \{v_1, v_2, v_3, v_4\}$ 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 the following two problems.

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

We refer to a balanced clique C satisfying $|C_L| \geq \tau$ and $|C_R| \geq \tau$ as satisfying the constraint τ. We denote the size of the maximum balanced clique C^* as $\omega_\tau(G)$, i.e., $\omega_\tau(G) = |C^*|$. Note that, there may exist multiple balanced cliques of size $\omega_\tau(G)$, and the maximum balanced clique computation problem only aims to find one of them.

Problem 2 (Large Balanced Clique Enumeration): Given a signed graph G and two non-negative thresholds τ and α, the

large balanced clique enumeration problem aims to enumerate all maximal balanced cliques $C \subseteq V$ satisfying the constraint τ such that $|C| \geq \omega_\tau(G) - \alpha$. A balanced clique is maximal if none of its proper supersets is balanced.

In the above problem definitions, we require each side (i.e., C_L and C_R) to be of size at least τ, in the same way as [16]. This is to ensure that the two sides are not extremely small, and to avoid reporting too skewed results. A guideline for specifying τ can be found in [31]. For the large balanced clique enumeration problem, we require the end-user to specify a relative size threshold α for C (i.e., $|C| \geq \omega_\tau(G) - \alpha$), instead of an absolute threshold λ (i.e., $|C| \geq \lambda$). This is because a relative size threshold α is much easier to specify than an absolute threshold λ: a too large λ may lead to no results, while a too small λ may lead to an enormous number of results. Consider the signed graph G in Fig. 2 and suppose $\tau = 2$. $C = \{v_1, v_2, v_3, v_4\}$ and $C^* = \{v_3, v_4, v_5, v_6, v_7, v_8\}$ are two balanced cliques. C^* is the maximum balanced clique since it is the largest possible and $\omega_2(G) = |C^*| = 6$. Suppose $\alpha = 2$, both C and C^* will be reported as large balanced cliques, since their sizes are no less than $\omega_2(G) - \alpha = 4$.

Hardness Analysis: We prove the hardness of the two problems in the following two theorems.

Theorem 1: The maximum balanced clique computation problem is NP-hard.

Proof: We prove the NP-hardness of the problem by reducing from the classic maximum clique problem over unsigned graphs which is NP-hard [32]. 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 τ vertices$C_L = \{u_1, \ldots, u_\tau\}$ where all edges are positive edges. Finally, we add into G^s a negative edge between each vertex of C_L and each vertex of V. Then, G has a clique of size τ if and only if G^s has a balanced clique satisfying the constraint τ. This is because (1) for each clique C_R of size τ in G, (C_L, C_R) forms a balanced clique satisfying the constraint τ in G^s, and (2) for each balanced clique (C_L, C'_R) satisfying the constraint τ in G^s, C'_R must be a clique of size at least τ in G. Thus, the maximum balanced clique problem is NP-hard. □

Theorem 2: The large balanced clique enumeration problem is #P-complete.

Proof: This theorem directly follows from the facts that (1) the problem of enumerating all maximal balanced cliques is #P-complete [16], and (2) it is a special case of our problem, i.e., by setting α to be a very large value (e.g., $\alpha = |V|$). □

III. Maximum Balanced Clique Computation

In this section, we study the maximum balanced clique computation 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*.

Algorithm 1: MBC.

Input: A signed graph $G = (V, E^+, E^-)$ and a threshold τ
Output: The maximum balanced clique C^*
1 Reduce G by VertexReduction and EdgeReduction of [16];
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 $\quad P_L \leftarrow P_L \setminus \{v\}$;

A. An Enumeration-Based Baseline Approach MBC

The problem of enumerating all maximal balanced cliques has been studied by Chen et al. [16], and their enumeration algorithm MBCEnum 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.

MBCEnum proposed by Chen et al. [16] is an adaptation of the classic BK algorithm, proposed by Bron and Kerbosch [26] 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 [16] also proposed a vertex reduction method VertexReduction and an edge reduction method EdgeReduction to reduce the input graph based on the threshold τ. The general idea of VertexReduction is that every vertex in a balanced clique satisfying the constraint τ must have a positive degree at least $\tau - 1$ and a negative degree at least τ; 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 constraint τ must participate in a certain number of triangles of each type; we omit the details, which can be found in [16]. 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 computation problem. The

pseudocode of MBC is shown in Algorithm 1, which is self-explanatory. It can be easily verified that all calls to $\mathsf{Enum}(C_L, C_R, P_L, P_R)$, 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.

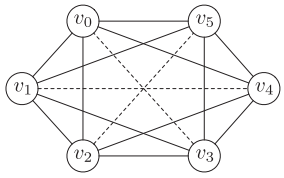


Fig. 3. Ineffectiveness of simply discarding edge signs.

B. A Dichromatic Clique-Based Approach MBC*

Our empirical studies in Section V 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 in MBC (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 on unsigned graphs [27], [28], [29], [30]. Specifically, we aim to utilize the *degree-based pruning* and *graph coloring-based upper bound* for computing maximum balanced clique. To illustrate, let's consider an *unsigned* graph, and a lower bound lb 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 [27]): 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 (1) we have already found a clique of size lb and we are now searching for cliques of size larger than lb and (2) the size of the largest clique that a vertex can participate in is its degree plus one. 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 bound [29]): The maximum clique size of a graph is at most its chromatic number.

Graph coloring-based upper bound is correct because a clique can include at most one vertex of each color, by noting that there is no edge between vertices of the same color. However, computing the chromatic number is an NP-hard problem [32]. Thus, heuristic techniques (e.g., see [29]) 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 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 discarding 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 Fig. 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 $\tau = 1$.

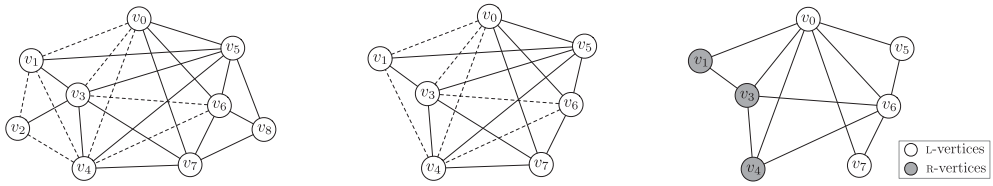
A Novel Graph Reduction Technique: To resolve the above drawbacks, we propose a novel graph reduction technique. The general idea is based on the following observation that removes edges from a subgraph. Suppose we know that vertex u is in the maximum balanced clique C^*. Then, according to the structural 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 above 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 τ 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 computation problem, which is formally defined as follows.

Problem 3 (Maximum Dichromatic Clique Computation): The input of the maximum dichromatic clique computation 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 τ. 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 τ.

The above discussion is based on the assumption that we know a vertex u in the maximum balanced clique. In practice,



(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.

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 such 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 computation over a large signed graph G to a series of maximum dichromatic clique computation 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 Fig. 4(a) and assume v_0 has the lowest rank according to the total ordering $\prec$. Fig. 4(b) illustrates the ego-network G_{v_0} of v_0 which does not include vertices $\{v_2, v_8\}$, and Fig. 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 3: The size of the maximum balanced clique in $G = (V, E^+, E^-)$ that satisfies the constraint τ 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 τ.

Proof: Recall that $\omega_\tau(G)$ denotes the size of the maximum balanced clique in G that satisfies the constraint τ. 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 constraint τ. 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 $\omega_\tau(G) \geq \max_{u \in V} \delta(g_u, \tau)$. Second, let C^* be a maximum balanced clique in G satisfying the constraint τ, 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, $\omega_\tau(G) \leq \max_{u \in V} \delta(g_u, \tau)$, and the theorem holds. □

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.

- First, each dichromatic network is small as discussed above. Although we need to solve n instances of the maximum dichromatic clique computation problem in the worst case, our empirical studies in Section V 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.
- Second, 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.

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 [16] to reduce the input signed graph G (Line 1). Note that, we do not apply EdgeReduction of [16]; 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 V. 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 [33], 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 [33]. Based on the degeneracy ordering DOrder, we process vertices in the reverse order according to DOrder (Line 5); we say a vertex

Algorithm 2: MBC*.

Input: A signed graph $G = (V, E^+, E^-)$, and a threshold τ
Output: The maximum balanced clique C^*
1 Reduce G by VertexReduction of [16];
2 $C^* \leftarrow$ MBC-Heu(G, τ);
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 $g_u \leftarrow$ the dichromatic-network of u;
7 $g \leftarrow$ the $|C^*|$-core of g_u;
8 **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 **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 $v \leftarrow$ the vertex of $\mathcal{B}$ with the minimum degree in g;
19 **if** $v \in V_L(g)$ **then** $\tau'_L \leftarrow \tau_L - 1$; $\tau'_R \leftarrow \tau_R$;
20 **else** $\tau'_L \leftarrow \tau_L$; $\tau'_R \leftarrow \tau_R - 1$;
21 MDC$(C \cup \{v\}, g[N_g(v)], \tau'_L, \tau'_R)$;
22 Remove v from $\mathcal{B}$ and g;

ranks higher if it appears later in DOrder. Our intuition is that later vertices in the degeneracy ordering are more likely to be in dense subgraphs, e.g., there is a suffix of the degeneracy ordering that is a 2-approximation to the densest subgraph problem [34]. Thus, processing vertices in the reverse order of the degeneracy ordering is more likely to find large balanced cliques early on, and consequently we can use its size to prune a lot of the search space, e.g., at Lines 11–12 of Algorithm 2. 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 τ_L and τ_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 τ_L and τ_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 τ_L and τ_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 4: The time complexity of MBC* (Algorithm 2) is $\mathcal{O}(n \cdot m \cdot 2^{\delta})$ and the space complexity is $\mathcal{O}(m)$, where δ is the degeneracy of G.

Proof: First, 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. Second, 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 δ [29]. Thus, for a fixed dichromatic network g_u, the total number of invocations to MDC is at most 2^{δ}, 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 [35]. 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. □

C. A Heuristic Algorithm MBC-Heu

In this subsection, we present a heuristic algorithm MBC-Heu for a large balanced clique, 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

Algorithm 3: MBC-Heu.

Input: A signed graph $G = (V, E^+, E^-)$, and a threshold τ
Output: A balanced clique C satisfying the constraint τ or $\emptyset$
1 $u \leftarrow$ maximum-degree vertex in G;
2 $g \leftarrow$ the dichromatic network g_u of u;
3 $C \leftarrow \{u\}$;
4 **while** $V_L(g) \neq \emptyset$ **or** $V_R(g) \neq \emptyset$ **do**
5 **if** $V_L(g) = \emptyset$ **or** ($V_R(g) \neq \emptyset$ **and** $|C_L| \geq |C_R|$) **then**
6 $v \leftarrow$ maximum-degree vertex of $V_R(g)$ in g;
7 **else** $v \leftarrow$ maximum-degree vertex of $V_L(g)$ in g;
8 $C \leftarrow C \cup \{v\}$; $g \leftarrow g[N_g(v)]$;
9 **if** $|C_L| \geq \tau$ **and** $|C_R| \geq \tau$ **then return** C;
10 **else return** $\emptyset$;

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.

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 constraint τ (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 5: The time complexity and space complexity of MBC-Heu (Algorithm 3) are both $\mathcal{O}(m)$.

Proof: 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 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, i.e., there is only one copy of g during the execution. □

IV. Large Balanced Clique Enumeration

In this section, we study the large balanced clique enumeration problem, which aims to enumerate all maximal balanced cliques C satisfying the constraint τ such that $|C| \geq \omega_\tau(G) - \alpha$ for user-given thresholds τ and α. We first present a baseline algorithm BCE in Section IV-A, and then propose a dichromatic clique-based efficient algorithm BCE* in Section IV-B.

A. A Baseline Approach BCE

Recall that the MBCEnum algorithm proposed by Chen et al. [16] enumerates all maximal balanced cliques satisfying the constraint τ. A baseline algorithm can be designed by adapting MBCEnum. A straightforward adaptation would be first invoking MBCEnum to enumerate all maximal balanced cliques and then reporting those of size at least $|C^*| - \alpha$, where C^* is the largest one among all identified maximal balanced cliques. It is easy to observe that much computation is wasted on detecting small maximal balanced cliques in this straightforward adaptation.

To improve the efficiency, we incorporate pruning techniques into the enumeration process such that small maximal balanced cliques are not enumerated. That is, let C^* be the largest balanced clique found so far, then we use $|C^*| - \alpha$ as a lower bound to prune search branches that will not generate large balanced cliques; note that C^* keeps updating during the enumeration process and is a maximum balanced clique when the algorithm terminates. Recall from Section III-A that MBCEnum grows a balanced clique $C_L \cup C_R$ by iteratively adding candidate vertices from P_L to C_L and adding candidate vertices from P_R to C_R. Thus, in a search branch with inputs (C_L, C_R, P_L, P_R), the size of the largest balanced clique that can be found through recursions of this search branch is at most $|C_L| + |C_R| + |P_L| + |P_R|$. Consequently, we can safely terminate this search branch if its upper bound $|C_L| + |C_R| + |P_L| + |P_R|$ is smaller than $|C^*| - \alpha$. We denote this baseline algorithm as BCE.

B. A Dichromatic Clique-Based Approach BCE*

Due to the same drawbacks as MBC for computing a maximum balanced clique, BCE is still inefficient for enumerating all large balanced cliques. In this subsection, we propose a dichromatic clique-based approach BCE* for efficiently enumerating all large balanced cliques, by following the general idea of MBC* that is introduced in Section III-B.

Following the arguments in Section III-B, we know that for any balanced clique C containing u, 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 $u \in C_L$. Therefore, all large balanced cliques containing u can be found in the subgraph G_u of G induced by vertices $\{u\} \cup N_G^+(u) \cup N_G^-(u) = \{u\} \cup N_G(u)$. In addition, all conflicting edges can be removed from G_u such that we can ignore edge signs and transform G_u into a dichromatic-network g_u, in the same way as in Section III-B. Let $V_L = \{u\} \cup N_G^+(u)$ be the L-vertices and $V_R = N_G^-(u)$ be the R-vertices. It is easy to verify that *each balanced clique C containing u in G is a dichromatic clique containing u in g_u where $C_L = C \cap V_L$ and $C_R = C \cap V_R$, and vice versa*. Consequently, our problem becomes enumerating large maximal dichromatic cliques in g_u; note that every maximal clique in g_u must contain u, due to the way of constructing g_u. We term this problem as large dichromatic clique enumeration problem, formally defined as follows.

Problem 4 (Large Dichromatic Clique Enumeration): Given a *dichromatic graph* $g = (V(g), E(g))$ where $V(g) = V_L \cup V_R$ with $V_L \cap V_R = \emptyset$, and non-negative thresholds τ and λ, the large dichromatic clique enumeration problem aims to enumerate all maximal cliques C in g such that $|C| \geq \lambda$, $|C \cap V_L| \geq \tau$, and $|C \cap V_R| \geq \tau$.

Algorithm 4: BCE*.

Input: A signed graph G, and thresholds τ and α
Output: All large maximal balanced cliques in G

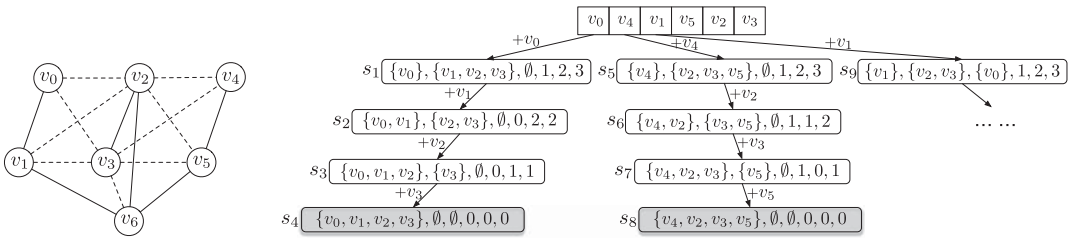
1 $C^* \leftarrow \mathsf{MBC}^*(G, \tau)$;
2 $\lambda \leftarrow \max\{|C^*| - \alpha, 2\tau\}$;
3 Reduce G based on λ and τ;
4 Compute a degeneracy ordering for vertices of G; w.l.o.g., assume the ordering is $\{v_1, v_2, \ldots, v_n\}$;
5 **for each** *vertex* $v_i \in \{v_1, v_2, \ldots, v_n\}$ **do**
6 $\quad g_{v_i} \leftarrow$ the dichromatic-network of v_i induced by $\{v_i\} \cup (N_G(v_i) \cap \{v_{i+1}, \ldots, v_n\})$;
7 $\quad$ Reduce g_{v_i} based on λ and τ;
8 $\quad$ **if** $v_i \notin V(g_{v_i})$ **or** $\mathsf{colorUB}(g_{v_i}) < \lambda$ **then continue**;
9 $\quad \mathsf{DCE}(\{v_i\}, g_{v_i} \backslash \{v_i\}, N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}, \tau - 1, \tau, \lambda - 1)$;

Procedure $\mathsf{DCE}(C, g = (V_L, V_R, E), X, \tau_L, \tau_R, \lambda)$
10 **if** $g = \emptyset$ **and** $X = \emptyset$ **and** $\tau_L \leq 0$ **and** $\tau_R \leq 0$ **and** $\lambda \leq 0$ **then**
11 $\quad$ **report** C as a large maximal balanced clique;
12 Reduce g based on λ, τ_L and τ_R;
13 **if** $\mathsf{colorUB}(g) < \lambda$ **then return**;
14 $v^* \leftarrow \arg\max_{v \in X \cup V(g)} |N_g(v)|$;
15 **for each** *vertex* $v \in V(g) \setminus N_g(v^*)$ **do**
16 $\quad$ **if** $v \in V_L(g)$ **then** $\tau'_L \leftarrow \tau_L - 1$; $\tau'_R \leftarrow \tau_R$;
17 $\quad$ **else** $\tau'_L \leftarrow \tau_L$; $\tau'_R \leftarrow \tau_R - 1$;
18 $\quad \mathsf{DCE}(C \cup \{v\}, g[N_g(v)], X \cap N(v), \tau'_L, \tau'_R, \lambda - 1)$;
19 $\quad g \leftarrow g \backslash \{v\}$; $X \leftarrow X \cup \{v\}$;

To obtain all large balanced cliques in G, we can enumerate all large dichromatic cliques in the dichromatic-network g_u for each $u \in V(G)$. However, an obvious issue is that a maximal dichromatic clique C will be reported multiple times. That is, C will be reported when processing the dichromatic-network g_u for each $u \in C$. This also means that a lot of computations are wasted. To avoid duplication and to speed up the computation, we define a total ordering $\prec$ for all vertices of G — without loss of generality, assume the ordering is $\{v_1, v_2, \ldots, v_n\}$ — and revise the definition of dichromatic-network g_{v_i} to be the subgraph of G induced by v_i and v_i's higher-ranked neighbors, i.e., induced by $\{v_i\} \cup (N_G(v_i) \cap \{v_{i+1}, \ldots, v_n\})$. Still, conflicting edges are removed from g_{v_i} and its vertices are labeled as L-vertices and R-vertices. Then, when processing g_{v_i}, we only aim to enumerate all maximal balanced cliques C of G such that $v_i \in C$ and v_i ranks the lowest among all vertices of C according to $\prec$. Note however that, due to the revised definition of g_{v_i}, a maximal clique C in g_{v_i} may not be maximal in G; specifically, some of v_i's lower-ranked neighbors, i.e., $N_G(v_i) \cap \{v_1, v_2, \ldots, v_{i-1}\}$, may be able to be added to enlarge C. To check whether a maximal clique C in g_{v_i} is also maximal in G, we maintain the set of common neighbors of C in G, denoted Λ_C. To be precise, Λ_C is the set of vertices of G such that each vertex $x \in \Lambda_C$ (i) either has a positive edge to every vertex of $C \cap V_L$ and a negative edge to every vertex of $C \cap V_R$, (ii) or has a positive edge to every vertex of $C \cap V_R$ and a negative edge to every vertex of $C \cap V_L$. As a result, C is maximal if and only if $\Lambda_C = \emptyset$.

Based on the above discussions, we transform the large balanced clique enumeration problem over a signed graph to a series of large dichromatic clique enumeration problems over unsigned dichromatic-networks. The pseudocode of our algorithm for enumerating all large balanced cliques is given in Algorithm 4, denoted BCE*. We first compute the maximum balanced clique C^* in G by invoking MBC* (Line 1), and derive the size threshold of large balanced cliques as the larger one between $|C^*| - \alpha$ and 2τ, denoted as λ (Line 2); note that, a balanced clique satisfying the constraint τ must be of size at least 2τ. Then, we prune unqualified vertices from G based on λ and τ (Line 3). That is, any vertex in a large balanced clique must have at least $\tau - 1$ positive neighbors, at least τ negative neighbors and at least $\lambda - 1$ neighbors in total; thus, any vertex violating either of these three conditions can be safely removed from G. Next, we compute a degeneracy ordering for vertices of G (Line 4) — without loss of generality, we assume the degeneracy ordering is $\{v_1, v_2, \ldots, v_n\}$ — and process vertices in this order (Line 5). Note that, the vertex processing order actually does not matter here since the thresholds τ and λ used for pruning are fixed; that is, processing vertices in different orders at Line 5 of Algorithm 4 would result in the same running time. When processing vertex v_i, we construct the dichromatic-network g_{v_i} (Line 6), which is the subgraph of G induced by $\{v_i\} \cup (N_G(v_i) \cap \{v_{i+1}, \ldots, v_n\})$ after removing conflicting edges and discarding edge signs, and enumerate all maximal balanced cliques of G that have v_i as the lowest-ranked vertex, by invoking the procedure DCE (Line 9). Before invoking DCE, we first reduce the size of g_{v_i} based on λ and τ (Line 7), and compute a coloring-based upper bound for the maximum clique size in g_{v_i} (Line 8); if the upper bound is smaller than λ, then g_{v_i} will contain no large balanced cliques. The reduction at Line 7 is similar to that at Line 3, with subtle differences. Specifically, a vertex in V_L (resp. V_R) can be removed from g_{v_i} if it has either less than $\tau - 1$ neighbors in V_L (resp. V_R), or less than τ neighbors in V_R (resp. V_L), or less than $\lambda - 1$ total neighbors in $V_L \cup V_R$.

The pseudocode of DCE is also given in Algorithm 4. The input of DCE consists of vertex subsets $C \subseteq V(g_{v_i})$ and an exclusive set $X \subseteq V$, a subgraph g, and three thresholds τ_L, τ_R, λ. The algorithm maintains the invariants that (1) $C, V(g)$, and X are disjoint, (2) $V(g) \cup X$ is the set of common neighbors of C, i.e., $V(g) \cup X = \Lambda_C$ as defined above, such that C is maximal if and only if $V(g) \cup X = \emptyset$, and (3) $C \cup C'$ for $C' \subseteq V(g)$ is a large maximal dichromatic clique in g_{v_i} if and only if C' is a maximal dichromatic clique in g such that $|C' \cap V_L(g)| \geq \tau_L$, $|C' \cap V_R(g)| \geq \tau_R$, $|C'| \geq \lambda$. DCE works as follows. First, C is a large maximal balanced clique in G if $V(g) \cup X = \emptyset$, and $\tau_L \leq 0, \tau_R \leq 0$ and $\lambda \leq 0$ (Lines 10–11). If C is not maximal, then we reduce g based on τ_L, τ_R, λ and conduct upper bound-based pruning by using the coloring-based upper bound (Lines 12–13), in a similar way to Lines 7–8. After that, we pick a pivot vertex from $V(g) \cup X$, which is the vertex that has the largest number of neighbors in g (Line 14). Note that g is a subgraph of g_{v_i}. For a vertex that is in g_{v_i}, its set of neighbors in g is trivially defined based on g_{v_i}. For a vertex v that is not in g_{v_i}, i.e., $v \in N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ that is passed to DCE at Line 9, its neighbors in g_{v_i} and thus in g are defined as follows where conflicting edges between $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ and $V(g_{v_i})$ are removed: if v is a positive neighbor v_i in G, then v's neighbors in g_{v_i} are defined as the positive neighbors in $V_L(g_{v_i})$ and negative neighbors in $V_R(g_{v_i})$; otherwise, v's neighbors in g_{v_i} are defined as the positive neighbors in $V_R(g_{v_i})$ and negative neighbors in $V_L(g_{v_i})$. Let $v^* \in V(g) \cup X$ be the pivot



(a) An example signed graph (b) A snippet of the search tree of BCE* ($\tau = 2$ and $\alpha = 1$)

Fig. 5. Illustration of BCE*.

vertex. Then, it is easy to see that any maximal clique in g must contain a non-neighbor of v^*. Thus, we only need to generate new branches for each vertex of $V(g) \setminus N_g(v^*)$ (Line 15). It is worth mentioning that although DCE shares similar spirit to the pivoted version of the BK algorithm [26] that enumerates all maximal cliques in an unsigned graph, DCE is different by considering dichromatic cliques and incorporating pruning techniques at Lines 12–13. Moreover, our idea of transforming the large balanced clique enumeration problem to large dichromatic clique enumeration problems that do not need to consider edge signs is new.

Example 2: A snippet of the search tree of running BCE* on the signed graph in Fig. 5(a) with $\tau = 2$ and $\alpha = 0$ is shown in Fig. 5(b), where each rectangle is a search state s_i represented by $(C, V(g), X, \tau_L, \tau_R, \lambda)$. First, we invoke MBC* to compute the maximum balanced clique $C^* = \{v_0, v_1, v_2, v_3\}$, and derive the size threshold λ as $\max\{|C^*| - \alpha, 2\tau\} = 4$. Then we reduce the graph based on λ and τ; in this step, v_6 will be pruned since its negative degree is less than 2. Next, the degeneracy ordering for the remaining vertices is computed as $\{v_0, v_4, v_1, v_5, v_2, v_3\}$, and we process them in this order.

Now, let us see the enumeration process for v_0. We construct the dichromatic-network g_{v_0}, which is the subgraph of G induced by $\{v_0, v_1, v_2, v_3\}$ by discarding the conflicting edges and edge signs. Then, we start to find large dichromatic cliques in g_{v_0} by invoking DCE. As shown in state s_1, C is initialized as $\{v_0\}$ and $V(g)$ is initialized as $\{v_1, v_2, v_3\}$. Note that v_1 is an L-vertex since it is a positive neighbor of v_0, v_2 and v_3 are R-vertices since they are negative neighbors of v_0. X is empty since v_0 is the lowest-ranked vertex. τ_L, τ_R and λ are 1, 2 and 3, respectively. For s_1, vertex v_1 is chosen as the pivot. Since both v_2 and v_3 are adjacent to the pivot v_1, only one new branching is generated for s_1, i.e., v_1 is added into C and we reach state s_2. After adding v_1 to C, both τ_L and λ are decreased by 1 since v_1 is an L-vertex. For s_2, vertex v_2 is selected as the pivot and we only generate one new branching by adding v_2 into C (i.e., state s_3). Similarly, we continue the expansion by adding v_3 into C and reach the maximal balanced clique $C = \{v_0, v_1, v_2, v_3\}$ at state s_4. In a similar way, we enumerate large balanced cliques for other vertices. Finally, we find two large balanced cliques in this graph: $C_1 = \{v_0, v_1, v_2, v_3\}$ (at state s_4) and $C_2 = \{v_2, v_3, v_4, v_5\}$ (at state s_8).

Theorem 6: The time complexity of BCE* (Algorithm 4) excluding Line 1 is $\mathcal{O}(\delta \cdot n \cdot 3^{\delta/3})$ and the space complexity is $\mathcal{O}(m)$, where δ is the degeneracy of G.

Proof: Line 2 runs in constant time, and each of Line 3 and Line 4 runs in $\mathcal{O}(m)$ time. Then, we enumerate all large balanced cliques for each vertex v_i by invoking DCE (Lines 5–9). As each dichromatic-network g_{v_i} contains at most δ vertices (i.e., $|N_G(v_i) \cap \{v_{i+1}, \ldots, v_n\}| \leq \delta$), Lines 6–8 run in $\mathcal{O}(\delta^2)$ time. Also note that Lines 11–13 run in $\mathcal{O}(|V(g)|^2)$ time. Thus, the total time complexity of Algorithm 4 excluding Line 1 is $\mathcal{O}(\delta \cdot n \cdot 3^{\delta/3})$ by following the arguments of [35]. In addition, the space complexity of Algorithm 4 is $\mathcal{O}(m)$ which can be shown in the same way as the proof of Theorem 4. □

V. Experiments

In this section, we empirically evaluate the efficiency and effectiveness of our algorithms. For the maximum balanced clique computation problem, we implemented MBC (Algorithm 1), MBC* (Algorithm 2), as well as three variants of these two algorithms: (1) MBC-noER which is MBC without EdgeReduction, (2) MBC-Adv which is the improved version of MBC that applies the degree-based pruning and coloring-based upper bound by ignoring edge signs, and (3) MBC*-withER which is the variant of MBC* that also applies EdgeReduction at Line 1 of Algorithm 2.

For the large balanced clique enumeration problem, we implemented the baseline algorithm BCE and our advanced algorithm BCE* (Algorithm 4). In addition, we also implemented BCE-Adv, which is an improved version of BCE by (1) invoking MBC* to compute $\omega_\tau(G)$ and (2) conducting degree-based pruning and coloring-based upper bounding via ignoring edge signs.

All the algorithms are implemented in C++, compiled with g++ 7.5.0 with the -O3 flag.[1] All the experiments are conducted on a machine with an Intel Core-i7 3.20 GHz CPU and 64 GB RAM running Ubuntu 18.04. 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 threshold $\tau = 3$ by default.

Datasets: We evaluate the algorithms on 13 datasets. Bitcoin, Reddit and Epinions are signed networks downloaded from SNAP.[2] AdjWordNet[3] captures the synonym-and-antonym relation among adjectives in the English language. Referendum and WikiConflict are signed networks used in [19] and are obtained from the authors of [19]. Amazon, BookCross, TripAdvisor and YahooSong are rating networks downloaded from KONECT .[4]

[1] Our source codes are available at https://github.com/kyaocs/MSBC and https://github.com/kyaocs/BCE, respectively, for the two studied problems.
[2] https://snap.stanford.edu/
[3] https://wordnet.princeton.edu/
[4] http://konect.cc/networks/

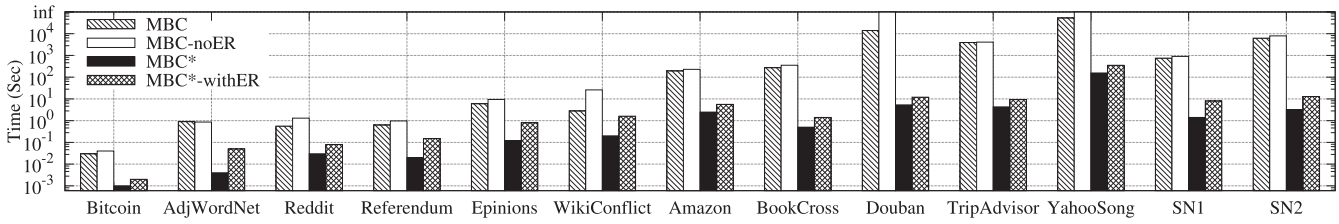


Fig. 6. Running time on all graphs for maximum balanced clique computation ($\tau = 3$).

TABLE II
STATISTICS OF DATASETS

Dataset	$\|V\|$	$\|E\|$	$\frac{\|E^-\|}{\|E\|}$	$\omega_3(G)$	Category
Bitcoin	5,881	21,492	15%	11	Trade
AdjWordNet	16,259	76,845	32%	60	Language
Reddit	54,075	220,151	8%	8	Social
Referendum	10,884	251,406	5%	19	Political
Epinions	131,828	711,210	17%	15	Social
WikiConflict	116,717	2,026,646	63%	6	Editing
Amazon	176,816	2,685,570	11%	29	Rating
BookCross	63,535	3,890,104	7%	550	Rating
Douban	1,588,455	18,709,948	25%	116	Social
TripAdvisor	145,315	20,569,277	14%	1,916	Rating
YahooSong	1,000,990	30,139,524	18%	127	Rating
SN1	2,000,000	50,154,048	41%	13	Synthetic
SN2	2,000,000	111,573,268	39%	19	Synthetic

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. Douban is a signed network used in [9]. 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 [36]. Statistics of the graphs are shown in Table II, in which $\frac{|E^-|}{|E|}$ is the ratio of negative edges and $\omega_3(G)$ is the maximum balanced clique size under threshold $\tau = 3$.

A. *Efficiency for Maximum Balanced Clique Computation*

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 Fig. 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 Fig. 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.

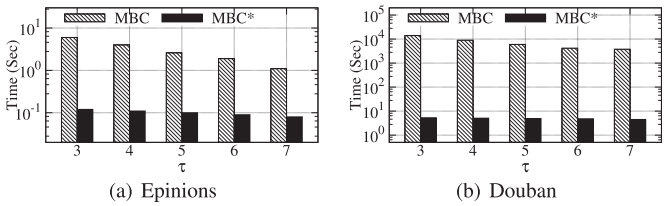


Fig. 7. Varying the threshold τ.

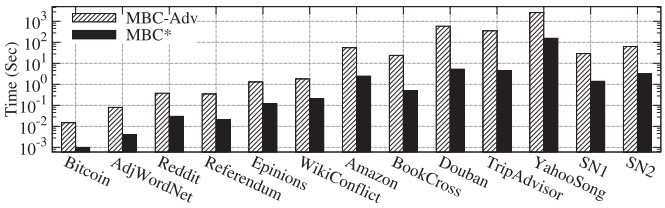


Fig. 8. Influence of our MDC transformation.

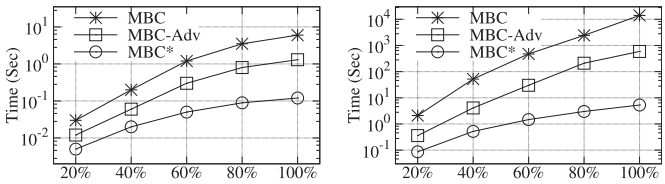
The results of evaluating MBC* against MBC by varying τ from 3 to 7 are shown in Fig. 7. MBC* consistently outperforms MBC for different τ values. We also observe that the running time of MBC decreases as τ increases, while that of MBC* is almost insensitive to τ. This is because for larger τ values, the pruning power of EdgeReduction 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 Fig. 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

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

Graphs	Heu	#MDC	SR1	SR2
Bitcoin	11	0	-	-
AdjWordNet	60	0	-	-
Reddit	6	96	41%	86%
Referendum	19	20	6%	31%
Epinions	11	82	30%	70%
WikiConflict	0	43	48%	94%
Amazon	15	952	64%	81%
BookCross	150	3	61%	85%
Douban	83	10	73%	98%
TripAdvisor	1916	0	-	-
YahooSong	36	575	23%	31%
SN1	10	19	45%	90%
SN2	12	28	42%	85%



(a) Epinions (b) Douban

Fig. 9. Scalability testing ($\tau = 3$, vary graph size).

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 III, 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 III. 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.

Scalability Testing: We now test the scalability of MBC, MBC-Adv and MBC* on two large datasets Epinions 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 Fig. 9, 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.

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 III. On Bitcoin, AdjWordNet, Referendum and TripAdvisor, MBC-Heu finds the optimal solution. However, there are also graphs for which the

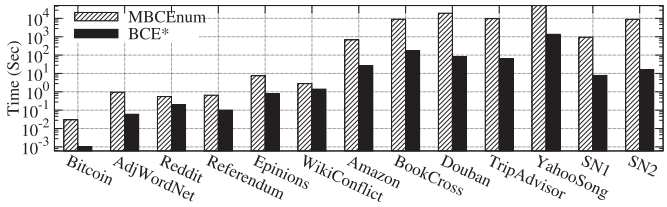


Fig. 10. Compare with MBCEnum [16] for $\alpha = \infty$ ($\tau = 3$).

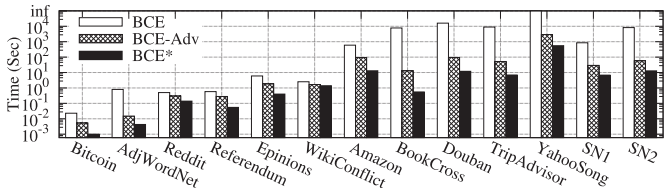


Fig. 11. Running time on all graphs for large balanced clique enumeration ($\tau = 3$ and $\alpha = 1$).

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 τ.

B. Efficiency for Large Balanced Clique Enumeration

Enumerating All Maximal Balanced Cliques: We first compare our algorithm BCE* with the maximal balanced clique enumeration algorithm MBCEnum proposed by Chen et al. [16], i.e., we set $\alpha = \infty$ for our problem. The results on all graphs are shown in Fig. 10. Our algorithm BCE* consistently runs faster than MBCEnum, and the improvement can be up to two orders of magnitude, e.g., on graphs BookCross, Douban, TripAdvisor, YahooSong, SN1 and SN2. This demonstrates the superiority of our novel graph reduction technique that transforms maximal balanced clique enumeration over G to maximal dichromatic clique enumeration over small dichromatic networks of G.

Against Baseline Algorithm BCE*:* In this evaluation, we compare our algorithm BCE* against the baseline algorithm BCE for enumerating large balanced cliques. The results on all datasets for $\tau = 3$ and $\alpha = 1$ are reported in Fig. 11. We can clearly see that BCE* outperforms BCE on all datasets, and the improvement is up-to four orders of magnitude (e.g., on BookCross). The efficiency of BCE* over BCE is mainly due to two reasons. First, BCE* invokes our algorithm MBC* to efficiently obtain the maximum balanced clique C^*, based on which a large portion of the input graph can be pruned by Line 3 of Algorithm 4. Second, our novel graph reduction technique transforms large balanced clique enumeration problem over G to a series of large dichromatic clique enumeration problems over small dichromatic networks of G. The dichromatic networks are generally small after removing conflicting edges and pruning unpromising vertices based on λ and τ. We also observe that the improvement of BCE* over BCE on WikiConflict is not as significant as on other datasets. The main reason is that all

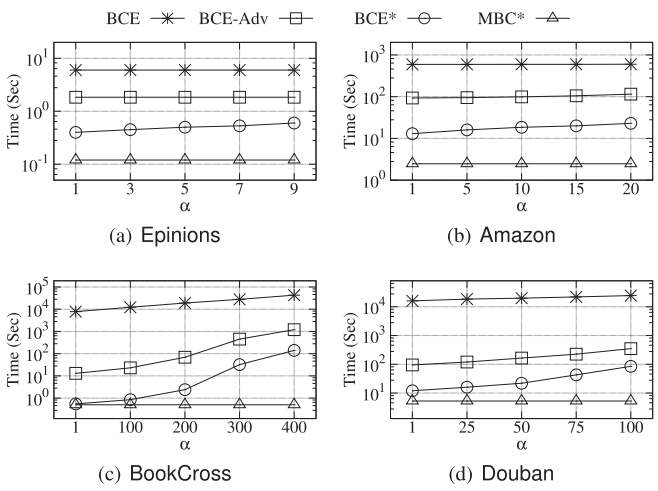


Fig. 12. Running time by varying α ($\tau = 3$).

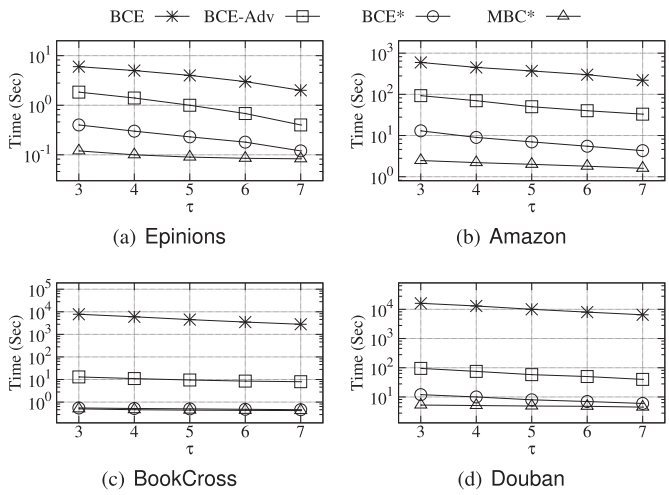


Fig. 13. Running time by varying τ ($\alpha = 1$).

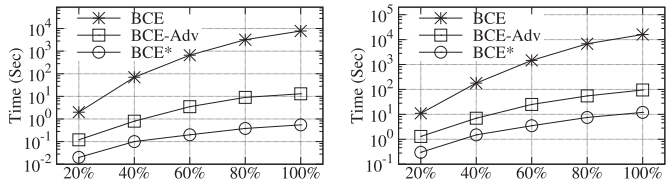


Fig. 14. Scalability testing (vary graph size).

maximal balanced cliques in WikiConflict are of the same size; this can be observed from Table V. Consequently, the size-based pruning techniques used in BCE* have no effect on this dataset.

To separately evaluate the advantage of our graph reduction technique, we compare BCE* against BCE − Adv that also invokes MBC* for computing $\omega_\tau(G)$ but does not apply our graph reduction technique. In contrast, BCE − Adv conducts the degree-based pruning and coloring-based upper bounding by simply discarding edge signs. The results are also shown in Fig. 11. We can see that BCE* consistently outperforms BCE-Adv, and the improvement is up to one order of magnitude, e.g., on graphs Amazon, BookCross and Douban. This demonstrates the advantage of our graph reduction technique.

Evaluate the Algorithms by Varying α: In this experiment, we evaluate the algorithms BCE*, BCE and BCE − Adv by varying α. The results on graphs Epinions, Amazon, BookCross, and Douban are shown in Fig. 12. Note that the values of α are selected according to $\omega_\tau(G)$ and thus are different for different graphs; please refer to Table II for the values of $\omega_\tau(G)$ for these graphs. We can see that as expected, the running time of all the three algorithms increase, since more results will be reported. For example, on BookCross, the number of large maximal balanced cliques increases from 384 to 19,841,800 when α increases from 1 to 400. Nevertheless, BCE* consistently outperforms both BCE and BCE-Adv. In Fig. 12 we also report the running time of MBC* which is invoked by BCE* and BCE − Adv to compute $\omega_\tau(G)$ and is irrelevant to α. We can see that the running time of MBC* is not significant compared with that of BCE*; this is more evident for large α. Thus, it makes sense to invoke MBC* for computing $\omega_\tau(G)$ in BCE*.

Evaluate the Algorithms by Varying τ: In this experiment, we evaluate the algorithms BCE*, BCE and BCE-Adv by varying τ from 3 to 7. Here, α is set as 1. The results on graphs Epinions, Amazon, BookCross, and Douban are shown in Fig. 13. We can see that the running time of all three algorithms decreases when τ increases. There are two main reasons. First, more vertices can be pruned in the vertex reduction phase when τ becomes large. For example, on Epinions, the resulting graph after vertex reduction contains 6,461 vertices and 246,054 edges for $\tau = 3$, and the numbers become 2,155 and 110,033 for $\tau = 7$. Second, when τ increases, the number of large balanced cliques decreases, which means a smaller search space. For example, on Epinions, the number of large balanced cliques is 86 for $\tau = 3$ and the number decreases to 1 for $\tau = 7$.

Scalability Testing: In this experiment, we test the scalability of BCE, BCE-Adv and BCE* on BookCross and Douban by varying the graph size to be $\{20\%, \ldots, 100\%\}$ of the input graph. The results are shown in Fig. 14. We can see that when the graph size increases, the processing time of all algorithms increases which is as expected. Nevertheless, BCE* outperforms BCE and BCE-Adv in all cases and scales better than BCE.

C. Efficiency on Synthetic Graphs

In this subsection, we evaluate the influence of degeneracy and negative edge ratio on the efficiency of our algorithms MBC* and BCE*. We generate synthetic graphs by varying the parameters.

Influence of Degeneracy: To evaluate the influence of degeneracy on our algorithms, we generate a set of synthetic graphs with different degeneracy values based on SN1. Specifically, we fix the number of vertices as 2,000,000, and generate five synthetic graphs with degeneracy values ranging from 10 to 200 by increasing the exponent of degree distribution of the graph generator. Correspondingly, the number of edges increases from 16,812,827 to 435,256,749; that is, as graph degeneracy increases, so does graph density. The performances of MBC* and BCE* on these five graphs for $\tau = 3$ and $\alpha = 1$ are shown

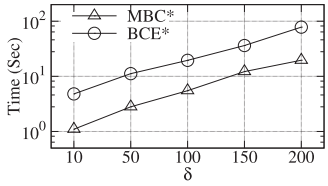


Fig. 15. Evaluate degeneracy.

TABLE IV
INFLUENCE OF DEGENERACY AND NEGATIVE EDGE RATIO ON MBC* AND BCE*

Degeneracy	$\omega_3(G)$	# of LBC	Negative edge ratio	$\omega_3(G)$	# of LBC
10	9	5	10%	6	7
50	14	8	30%	12	7
100	19	11	50%	15	9
150	22	12	70%	11	6
200	25	15	90%	9	5

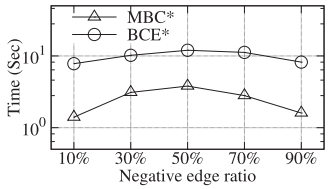


Fig. 16. Evaluate negative edge ratio.

in Fig. 15. We can see that the running time of both MBC* and BCE* increases with the increasing of degeneracy. There are two main reasons. First, fewer vertices are removed in the vertex reduction phase when the degeneracy increases, as the vertices are more densely connected to each other. Second, there will be more MDC instances generated by MBC* and BCE* for larger degeneracy.

We also report the maximum balanced clique size and the number of large balanced cliques under different degeneracy values in the first three columns of Table IV; here, $\tau = 3$ and $\alpha = 1$. As expected, both the size of the maximum balanced clique and the number of large balanced cliques increase with the increase of graph degeneracy.

Influence of Negative Edge Ratio: To evaluate the influence of negative edge ratio on our algorithms, we generate a set of synthetic graphs with different negative edge ratios based on SN1. Specifically, we fix the number of vertices as 2,000,000, and generate five graphs with increasing negative edge ratios from 10% to 90%. The results of running MBC* and BCE* on these graphs for $\tau = 3$ and $\alpha = 1$ are shown in Fig. 16. We can see that the running time of MBC* and BCE* peaks at the negative edge ratio of 50%. This is because when the negative edge ratio is low (resp. high), a significant number of vertices are removed in the vertex reduction phase due to the insufficient negative (resp. positive) degree. Besides, the lack of negative (resp. positive) edges impedes the formation of valid balanced cliques.

We also report the maximum balanced clique size and the number of large balanced cliques under different negative edge ratios in the last three columns of Table IV. We can see that both the maximum balanced clique and the number of large balanced cliques peak at the negative edge ratio of 50%.

TABLE V
NUMBER OF LARGE BALANCED CLIQUES BY VARYING α ($\tau = 3$)

Graphs	α						
	0	1	3	5	7	9	$\lvert V\rvert$
Bitcoin	6	31	87	133	133	133	133
AdjWordNet	2	4	4	4	4	4	858
Reddit	2	14	197	197	197	197	197
Referendum	2	27	585	2,661	4,773	5,958	6,773
Epinions	9	86	767	3,182	9,717	17,574	17,574
WikiConflict	37	37	37	37	37	37	37
Amazon	43	572	5,471	22,268	66,611	155,155	1,556,500
BookCross	192	384	384	384	384	384	20,643,998
Douban	83	210	773	1,298	3,768	6,293	93,275
TripAdvisor	2	2	2	2	2	2	8,570
YahooSong	3	17	83	312	767	1,563	195,964
SN1	1	5	14	29	56	56	56
SN2	1	7	18	35	65	133	397

D. Effectiveness of Enumerating Large Balanced Cliques

In this subsection, we evaluate the effectiveness of our model of enumerating only large maximal balanced cliques instead of all maximal balanced cliques that is studied by Chen et al. [16]. Note that, in our conference version [31], we also compared the effectiveness of the balanced clique model with the PolarSeeds model that is formulated by Xiao et al. [18].

The Number of Large Maximal Balanced Cliques by Varying α: In this experiment, we vary α from 0 to 9, and report the number of large maximal balanced cliques that are of size at least $\omega_\tau(G) - \alpha$. We also report the number of large maximal balanced cliques when $\alpha = |V|$. In this case, all maximal balanced cliques will be enumerated. The results on all graphs are shown in Table V. As expected, the number of reported balanced cliques increases when α becomes larger. We can also see that the number of maximal balanced cliques of size $\omega_\tau(G) - \alpha$ is already very large for $\alpha = 9$, e.g., on Referendum, Epinions, Amazon, Douban, and YahooSong; the number of all maximal balanced cliques will be even larger when $\alpha = |V|$. This demonstrates the usefulness of our model which only enumerates large maximal balanced cliques: by varying α, end-users can control the number of reported balanced cliques; also, we guarantee that all non-reported maximal balanced cliques will be of smaller size than the reported ones.

Case Study on AdjWordNet: In this experiment, we conduct case study on AdjWordNet, in which each vertex represents an adjective, and a positive (resp. negative) edge indicates synonymous (resp. antonymous) relationship. Here, we enumerate all large maximal balanced cliques with $\tau = 3$ and $\alpha = 40$; note that, $\omega_3(G) = 60$ for AdjWordNet. There are totally 119 cliques reported, among which 3 are illustrated in Table VI. The first one is a maximum balanced clique with $|C_L| = 28$ and $|C_R| = 32$, the second one is of size 36 (i.e., $|C_L| = 18$ and $|C_R| = 18$), and the third one is of size 28 (i.e., $|C_L| = 18$ and $|C_R| = 10$). 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

TABLE VI
CASE STUDY ON AdjWordNet

C_L	C_R
good, better, best, wonderful, excellent, great, superior, awesome, correct, well, sound, respectable, right, superb, honorable, optimum, terrific....	bad, worse, worst, terrible, poor, awful, inferior, unwell, weak, contemptible, dreadful, unsound, wrong, dishonorable, incorrect, unrespectable, horrific....
affixed, closed, tied, tight, cloudy, compact, covert, determinate, determined, fastened, ill-defined, incapable, opaque, shut, solved, unclear, unreceptive, buttoned	assailable, loose, open, opened, candid, capable, clear, exposed, heart-to-heart, overt, receptive, subject, undecided, undefendable, undefended, undetermined, unfastened, unresolved
easy, easygoing, gentle, mild, balmy, cushy, delicate, diffuse, diffused, flabby, flaccid, indulgent, lenient, piano, soft, sonant, subdued, voiced	uneasy, hard, hardened, difficult, intense, loud, nonindulgent, rugged, unvoiced, forte
...	...

antonymous with each other. This case study shows that large balanced cliques reveal interesting patterns in AdjWordNet.

VI. Related Works

Signed graph was first studied by Harary et al. [15], 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 by Figueiredo et al. [37] and Ordozgoiti et al. [10]. The problem is NP-hard. A branch-and-cut exact algorithm, which only works for graphs with up-to a few thousand vertices, is proposed by Figueiredo et al. [37]. Heuristic algorithms without any guarantee on the solution optimality are investigated in [10], [37]. 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*. 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 by Chen et al. [16], where the problem of enumerating all maximal balanced cliques is studied. We adapted the enumeration algorithm MBCEnum proposed by Chen et al. [16] to solve our problems. Our empirical studies show that our newly proposed algorithms MBC* and BCE* outperforms the adaptation of MBCEnum to solve our problems by several orders of magnitude. Moreover, BCE* also outperforms MBCEnum for enumerating all maximal balanced cliques.

Hao et al. [38] 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 by Hao et al. [38] cannot be applied to solve our problems. The notion of (α, k)-clique is defined by Li et al. [39] for signed graphs, which is a clique such that each vertex has at most k negative neighbors and at least αk positive neighbors in the clique. As the structural balanced constraint is ignored, the techniques proposed by Li et al. [39] cannot be applied to our problem.

On the other hand, there is a huge literature on the classic problems of *maximum clique computation* and *maximal clique enumeration* over unsigned graphs, e.g., [27], [28], [29], [30], [35], [40], [41]. However, these techniques cannot be directly applied to our problem due to the existence of edge signs and the enforcement of structural balanced constraint.

VII. Conclusion

In this paper, we first studied the maximum balanced clique problem in signed graphs. We proposed techniques to transform the maximum balanced clique problem over a signed graph 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 extended our techniques to the large balanced clique enumeration problem. Experimental results on real datasets demonstrated the efficiency, effectiveness and scalability of our algorithms.

References

[1] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 1–37, 2016.

[2] A. D. Easley and J. M. Kleinberg, *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. New York, NY, USA: Cambridge Univ. Press, 2010.

[3] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: Mining a social network with negative edges," in *Proc. 18th Int. Conf. World Wide*, 2009, pp. 741–750.

[4] C. Giatsidis, B. Cautis, S. Maniu, D. M. Thilikos, and M. Vazirgiannis, "Quantifying trust dynamics in signed graphs, the S-cores approach," in *Proc. Int. Conf. Des. Mater.*, 2014, pp. 668–676.

[5] L. Ou-Yang, D.-Q. Dai, and X.-F. Zhang, "Detecting protein complexes from signed protein-protein interaction networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 12, no. 6, pp. 1333–1344, Nov./Dec. 2015.

[6] T. Derr, C. Wang, S. Wang, and J. Tang, "Relevance measurements in online signed social networks," in *Proc. 14th Int. Workshop Mining Learn. Graphs*, 2018, pp. 1–8.

[7] I. Rahaman and P. Hosein, "Extending DeGroot opinion formation for signed graphs and minimizing polarization," in *Proc. Int. Conf. Complex Netw. Their Appl.*, 2021, pp. 298–309.

[8] F. S. Rizi and M. Granitzer, "Signed heterogeneous network embedding in social media," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, 2020, pp. 1877–1880.

[9] L. Chu, Z. Wang, J. Pei, J. Wang, Z. Zhao, and E. Chen, "Finding gangs in war from signed networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1505–1514.

[10] B. Ordozgoiti, "Antonis matakos, and aristides gionis. finding large balanced subgraphs in signed networks," in *Proc. Web Conf.*, 2020, pp. 1378–1388.

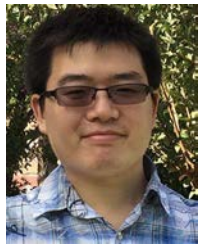
[11] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 641–650.

[12] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning," in *Proc. 19th Int. Conf. World Wide Web*, 2013, pp. 1477–1488.

[13] C.C. Chen, Y.-H. Meng-Chieh, W. Chung, and Y.-C. Sun, "An effective recommendation method for cold start new users using trust and distrust networks," *Inf. Sci.*, vol. 224, pp. 19–36, 2013.
[14] J. Tang, C. Aggarwal, and H. Liu, "Recommendations in signed social networks," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 31–40.
[15] F. Harary et al., "On the notion of balance of a signed graph," *Michigan Math. J.*, vol. 2, no. 2, pp. 143–146, 1953.
[16] Z. Chen, L. Yuan, X. Lin, L. Qin, and J. Yang, "Efficient maximal balanced clique enumeration in signed networks," in *Proc. Web Conf.*, 2020, pp. 339–349.
[17] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky, "Community interaction and conflict on the web," in *Proc. World Wide Web Conf.*, 2018, pp. 933–943.
[18] H. Xiao, B. Ordozgoiti, and A. Gionis, "Searching for polarization in signed graphs: A local spectral approach," in *Proc. Web Conf.*, 2020, pp. 362–372.
[19] F. Bonchi, E. Galimberti, A. Gionis, B. Ordozgoiti, and G. Ruffo, "Discovering polarized communities in signed networks," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 961–970.
[20] A. Suratanee et al., "Characterizing protein interactions employing a genome-wide siRNA cellular phenotyping screen," *PLoS Comput. Biol.*, vol. 10, no. 9, 2014, Art. no. e1003814.
[21] S. Yim, H. Yu, D. Jang, and D. Lee, "Annotating activation/inhibition relationships to protein-protein interactions using gene ontology relations," *BMC Syst. Biol.*, vol. 12, no. 1, 2018, Art. no. 9.
[22] A. Vinayagam et al., "Integrating protein-protein interaction networks with phenotypes reveals signs of interactions," *Nature Methods*, vol. 11, no. 1, pp. 94–99, 2014.
[23] G. A. Miller, "WordNet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[24] A. Krishnan, P. Deepak, S. Ranu, and S. Mehta, "Leveraging semantic resources in diversified query expansion," *World Wide Web*, vol. 21, no. 4, pp. 1041–1067, 2018.
[25] V. Kumar, N. Joshi, A. Mukherjee, G. Ramakrishnan, and P. Jyothi, "Cross-lingual training for automatic question generation," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4863–4872.
[26] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph (algorithm 457)," *Commun. ACM*, vol. 16, no. 9, pp. 575–576, 1973.
[27] E. Maslov, M. Batsyn, and P. M. Pardalos, "Speeding up branch and bound algorithms for solving the maximum clique problem," *J. Glob. Optim.*, vol. 59, no. 1, pp. 1–21, 2014.
[28] E. Tomita, "Efficient algorithms for finding maximum and maximal cliques and their applications," in *Proc. Int. Workshop Algorithms Comput.*, 2017, pp. 3–15.
[29] L. Chang, "Efficient maximum clique computation over large sparse graphs," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 529–538.
[30] L. Chang, "Efficient maximum clique computation and enumeration over large sparse graphs," *VLDB J.*, vol. 29, no. 5, pp. 999–1022, 2020.
[31] K. Yao, L. Chang, and L. Qin, "Computing maximum structural balanced cliques in signed graphs," in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 1004–1016.
[32] R. M. Karp, "Reducibility among combinatorial problems," in *Proc. IEEE 50th Annu. Symp. Found. Comput. Sci.*, pp. 85–103, 1972, pp. 283–292.
[33] W. D. Matula and L. L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *J. ACM*, vol. 30, no. 3, pp. 417–427, 1983.
[34] L. Chang and L. Qin, *Cohesive Subgraph Computation over Large Sparse Graphs*. Springer Series in the Data Sciences, Berlin, Germany: Springer, 2018.
[35] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," *ACM J. Exp. Algorithmics*, vol. 18, pp. 3.1–3.21, 2013.
[36] Y. Su, B. Wang, F. Cheng, L. Zhang, X. Zhang, and L. Pan, "An algorithm based on positive and negative links for community detection in signed networks," *Sci. Rep.*, vol. 7, no. 1, pp. 1–12, 2017.
[37] R. M. V. de Figueiredo and Y. Frota, "The maximum balanced subgraph of a signed graph: Applications and solution approaches," *Eur. J. Oper. Res.*, vol. 236, no. 2, pp. 473–487, 2014.
[38] F. Hao, S. S. Yau, G. Min, and L. T. Yang, "Detecting K-balanced trusted cliques in signed social networks," *IEEE Internet Comput.*, vol. 18, no. 2, pp. 24–31, Mar./Apr. 2014.
[39] R.-H. Li et al., "Efficient signed clique search in signed networks," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 245–256.
[40] C.-M. Li, Z. Fang, and K. Xu, "Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem," in *Proc. IEEE 25th Int. Conf. Tools Artif. Intell.*, 2013, pp. 939–946.
[41] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theor. Comput. Sci.*, vol. 363, no. 1, pp. 28–42, 2006.

Kai Yao received the bachelor degree from the China University of Mining and Technology, in 2014 and the MPhil degree from the Hong Kong University of Science and Technology, in 2019, and the PhD degree from the University of Sydney, in 2023. He is currently a research associate with the School of Computer Science with the University of Sydney. His research interests include big graph analytics and graph query processing.

Lijun Chang received the bachelor degree from the Renmin University of China, in 2007, and the PhD degree from the Chinese University of Hong Kong, in 2011. He is an associate professor in the School of Computer Science with the University of Sydney. He worked as a postdoc and then DECRA research fellow with the University of New South Wales from 2012 to 2017. His research interests are in the fields of big graph (network) analytics, with a focus on designing practical algorithms and developing theoretical foundations for massive graph analysis.



Lu Qin received the BE degree from the department of Computer Science and Technology in the Renmin University of China, in 2006, and the PhD degree from the Department of Systems Engineering and Engineering Management in the Chinese University of Hong Kong, in 2010. He is now an associate professor in the Centre of Quantum Computation and Intelligent Systems (QCIS) in the University of Technology Sydney (UTS). His research interests include parallel big graph processing, I/O efficient algorithms on massive graphs, and keyword search in relational databases.