# Balanced Clique Computation in Signed Networks: Concepts and Algorithms

Zi Chen, Long Yuan*, Xuemin Lin, Lu Qin, Wenjie Zhang

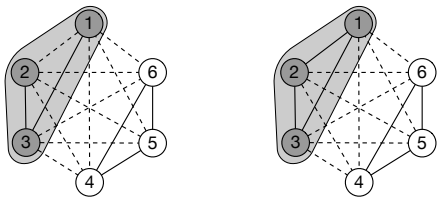
**Abstract**—Clique is one of the most fundamental models for cohesive subgraph mining in network analysis. Existing clique model mainly focuses on unsigned networks. However, in real world, many applications are modeled as signed networks with positive and negative edges. As the signed networks hold their own properties different from the unsigned networks, the existing clique model is inapplicable for the signed networks. Motivated by this, we propose the balanced clique model that considers the most fundamental and dominant theory, structural balance theory, for signed networks. Following the balanced clique model, we study the maximal balanced clique enumeration problem (MBCE) which computes all the maximal balanced cliques in a given signed network. Moreover, in some applications, users prefer a unique and representative balanced clique with maximum size rather than all balanced cliques. Thus, we also study the maximum balanced clique search problem (MBCS) which computes the balanced clique with maximum size. We show that MBCE problem and MBCS problem are both NP-Hard. For the MBCE problem, a straightforward solution is to treat the signed network as two unsigned networks and leverage the off-the-shelf techniques for unsigned networks. However, such a solution is inefficient for large signed networks. To address this problem, in this paper, we first propose a new maximal balanced clique enumeration algorithm by exploiting the unique properties of signed networks. Based on the new proposed algorithm, we devise two optimization strategies to further improve the efficiency of the enumeration. For the MBCS problem, we first propose a baseline solution. To overcome the huge search space problem of the baseline solution, we propose a new search framework based on search space partition. To further improve the efficiency of the new framework, we propose multiple optimization strategies regarding to redundant search branches and invalid candidates. We conduct extensive experiments on large real datasets. The experimental results demonstrate the efficiency, effectiveness and scalability of our proposed algorithms for MBCE problem and MBCS problem.

**Index Terms**—Balanced Clique, Structural Balance Theory, Signed Network, Graph Algorithm



## 1 Introduction

With the proliferation of graph applications, research efforts have been devoted to many fundamental problems in analyzing graph data [1], [2], [3], [4], [5], [6], [7], [8]. Clique is one of the most fundamental cohesive subgraph models in graph analysis, which requires each pair of vertices has an edge. Due to the completeness requirement, clique model owns many interesting cohesiveness properties, such as the distance of any two vertices in a clique is one, every one vertex in a clique forms a dominate set of the clique and the diameter of a clique is one [9]. As a result, clique model has wide application scenarios in social network mining, financial analysis and computational biology and has been extensively investigated for decades. Existing studies on clique mainly focus on the unsigned networks, i.e., all the edges in the graph share the same property [10], [11], [12], [13]. Unfortunately, relationships between two entities in many real-world applications have completely opposite properties, such as friend-foe relationships between users in social networks [14], [15], support-dissent opinions in opinion networks [16], trust-distrust relationships in trust networks [17] and partnership-antagonism in protein-protein interaction networks [18]. Modelling these applications as signed networks with positive and negative edges allows them to capture more sophisticated semantics than unsigned networks [17], [19], [20], [21], [22], [23]. Consequently, existing studies on clique ignoring the sign associated with each edge may be inappropriate to characterize the cohesive subgraphs in a signed network and there is an urgent need to define an exclusive clique model tailored for the signed networks.




(a) $G$ (b) $G'$

Fig. 1: Imbalanced Graph and Balanced Graph

For the signed networks, the most fundamental and dominant theory revealing the dynamics and construction of the signed networks is the *structural balance theory* [14], [17], [19], [20], [21], [22], [23], [24], [25]. The intuition underlying the structural balance theory can be described as the aphorisms: "The friend (resp. enemy) of my friend (resp. enemy) is my friend, the friend (resp. enemy) of my enemy (resp. friend) is my enemy". Specifically, a signed network $G$ is structural balanced if $G$ can be split into two subgraphs such that the edges in the same subgraph are positive and

* *Zi Chen and Long Yuan are the joint first authors. Long Yuan is the corresponding author.*

- *Z. Chen is with the Software Engineering Institute, East China Normal University, Shanghai, China. E-mail: zchen@sei.ecnu.edu.cn.*
- *L. Yuan is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. E-mail: longyuan@njust.edu.cn.*
- *X. Lin and W. Zhang are with University of New South Wales, Sydney, Australia. E-mail: {lxue,zhangw}@cse.unsw.edu.au*
- *L. Qin is with Centre for QCIS, University of Technology, Sydney, Australia. E-mail: lu.qin@uts.edu.au.*

the edges between subgraphs are negative [25]. In a signed network, an imbalanced sub-structure is unstable and tends to evolve into a balanced state. Consider the graph $G$ shown in Figure 1 (a). The negative edge between $v_1$ and $v_2$ makes $G$ imbalanced. $v_1$ and $v_2$ have a mutual "friend" $v_3$ and mutual "enemies" $v_4$, $v_5$ and $v_6$. It means $v_1$ and $v_2$ share more common grounds than differences. According to *structural balance theory*, $v_1$ and $v_2$ tend to be allies as time goes by. $G'$ shown in Figure 1 (b) is the evolved balanced counterpart of $G$. In $G'$, the sign of the edge between $v_1$ and $v_2$ becomes positive. $\{v_1, v_2, v_3\}$ and $\{v_4, v_5, v_6\}$ form two alliances and the edges in the same alliance are positive and the edges connecting different alliances are negative. As illustrated in this example, structural balance reflects the key characteristics of the signed networks.

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

Moreover, in real signed networks, the number of maximal balanced cliques could be extremely large. For instance, in "Douban" network which is a Chinese score service website, there are more than a million balanced cliques in it. However, in some applications, users prefer a unique and representative balanced clique with maximum size rather than all balanced cliques. Maximum clique search problem is a fundamental and hot research topic in graph analysis. In the literature, numerous studies have been conducted, such as maximum clique search [26], [27], maximum quasi-clique search [28], maximum bi-clique search [29], k*-partite clique with maximum edges [30], clique with maximum edge/vertex weight on weighted graph [31], [32]. Motivated by this, we aim to devise a maximum balanced clique search algorithm to find out the balanced clique with maximum vertex size, which can scale to large-scale real signed networks (with more than 100 million edges).

**Applications.** Balanced clique computation can be used in many applications, for example:

*(1) Opinion leaders detection in opinion networks.* Opinion leaders are people who are active in a community capturing the most representative opinions in the social networks [33]. In an opinion network, each vertex represents a user and there is a positive/negative edge between two vertices if one user support/dissent another user. A maximal balanced clique in an opinion network represents a group of users, such that these users actively involve in the opinion networks and have their clear standpoints. Hence, the users in the maximal balanced cliques are good candidates of opinion leaders in the opinion network.

*(2) Finding international alliances-rivalries groups.* The international relationships between nations can be modeled as a signed network, where each vertex represents a nation, positive and negative edges indicate alliances and rivalries, respectively. Computing the maximal balanced cliques in such networks reveals hostile groups of allied forces [14], [34]. We can extend it to find the alliances-rivalries commercial groups among business organizations similarly, such as {Pepsi, KFC} vs {Coke, McDonald} [35].

*(3) Synonym and antonym groups discovery.* In a word network, each vertex represents a word and there is a positive edge between two synonyms and a negative edge between two antonyms [36]. In such signed networks, our model can discover synonym groups that are antonymous with each other, such as, {interior, internal, intimate} and {away, foreign, outer, outside, remote}. These discovered groups may be further used in applications such as automatic question generation [37] and semantic expansion [38].

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

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

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

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

*(4) An efficient maximum balanced clique search algorithm.* To address the maximum balanced clique search problem, we first propose a baseline algorithm. In order to reduce the search space during the search process of baseline, we propose a search space partition-based algorithm MBCSear-SSP by partitioning the whole search space into multiple search regions. In each search region, two size thresholds $\overline{\kappa}$ and $\underline{\kappa}$ are used to search the result matching the size requirement specific to this search region, such that the search space is limited into a small area. To further improve the efficiency of MBCSear-SSP algorithm, we also explore three optimization strategies to prune invalid search branches and candidates during the search process.

*(5) Extensive performance studies on real datasets.* We first evaluate the performance of MBCE algorithms by conducting extensive experimental studies on real datasets. As shown in

our experiments, the baseline approach only works on small datasets while our approach can complete the enumeration efficiently on both small and large datasets. Then, we evaluate the performance of our proposed MBCS algorithm. The baseline algorithm can not get the result within a reasonable time on large datasets, while our optimized algorithm shows high efficiency, effectiveness and scalability.

## 2 Related Work

**Signed network analysis.** Structural balance theory is originally introduced in [24] and generalized in the graph formation in [19], [25]. After that, structural balance theory is developed extensively [17], [20], [21], [22], [23]. In these works, it is interesting to mention that the authors in [22] model the evolving procedure of a signed network and theoretically prove that the network would evolve into a balanced clique when the mean value of the initial friendliness among the vertices $\mu \leq 0$. [39] provides a comprehensive survey on structural balanced theory.

Besides, a large body of literature on mining signed networks has been emerged. Among them, the most closely related work to ours is [40] in which an $(\alpha, k)$-clique model is proposed. Compared with our model, $(\alpha, k)$-clique model only considers the amount of positive and negative edges in the clique and the structural balance of the clique is totally ignored, which makes $(\alpha, k)$-clique model essentially different from our model. In [41], a $k$-balanced trusted clique model is proposed. Although the $k$-balanced trusted clique model has a similar name with our model, it ignores the negative edges in the clique, which means the information of the negative edges are totally missed.

**Clique on unsigned networks.** Clique model is one of the most fundamental cohesive subgraph models. [10] proposes an efficient algorithm for maximal clique enumeration based on backtracking search. [42] first considers the memory consumption during the maximal clique enumeration. Based on [10], more efficient algorithms are investigated [11], [12], [43]. [11] proposes a novel branch pruning strategy, which can efficiently reduce the search space by ignoring the search process from the neighbors of the pivot. [44] reviews recently advances in maximal clique enumeration. Based on clique, other cohesive subgraph models are also studied, such as $k$-core [45], $k$-truss [46], [47], $k$-edge connected component [48], [49], [50], and $(r, s)$-nuclei [51], [52].

## 3 Problem Statement

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

**Definition 3.1: (Balanced Network [25])** Given a signed network $G = (V, E^+, E^-)$, it's balanced iff it can be split into two subgraphs $G_L$ and $G_R$, s.t. $\forall(u, v) \in E^+ \rightarrow u, v \in G_L$ or $u, v \in G_R$, and $\forall(u, v) \in E^- \rightarrow u \in G_L, v \in G_R$ or $u \in G_R, v \in G_L$. □

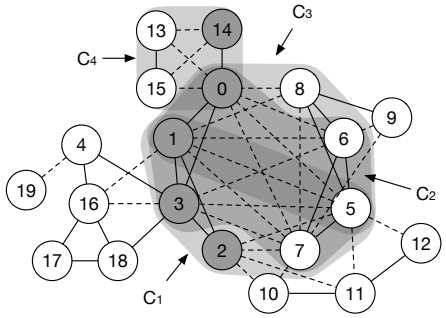



Fig. 2: Maximal Balanced Clique in $G$ ($k = 2$)

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

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

□

**Definition 3.3: (Maximum Balanced Clique)** Given a signed network $G = (V, E^+, E^-)$, a maximum balanced clique $C^*$ in $G$ is a balanced clique with the maximum vertex size. □

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

**Problem Statement.** Given a signed network $G$ and an integer $k$,

- the maximal balanced clique enumeration (MBCE) problem aims to compute all the maximal balanced cliques $C$ in $G$ s.t. $|C_L| \geq k$ and $|C_R| \geq k$ for $C$.
- the maximum balanced clique search (MBCS) problem aims to compute the balanced clique $C^*$ in $G$ s.t. $|C_L^*| \geq k$, $|C_R^*| \geq k$ and $|C_L^*| + |C_R^*|$ is maximum.

**Example 3.1:** Consider the signed network $G$ in Figure 2 in which positive/negative edges are denoted by solid/-dashed lines. Assume $k$ = 2, there are 4 maximal balanced cliques in $G$, namely, $C_1 = \{\{v_1, v_2, v_3\}, \{v_5, v_7\}\}$, $C_2 = \{\{v_0, v_1, v_3\}, \{v_5, v_6, v_7\}\}$, $C_3 = \{\{v_0, v_1\}, \{v_5, v_6, v_8\}\}$, $C_4 = \{\{v_0, v_{14}\}, \{v_{13}, v_{15}\}\}$, where vertices in $C_L$ and $C_R$ are marked with different colors. Among them, $C_2$ is the maximum balanced clique. □

**Problem Hardness.**The MBCE problem is NP-Hard, which can be proved following the NP-Hardness of maximal clique enumeration problem [53], [54]. Given an unsigned network $G = (V, E)$, we can transfer $G$ to a signed network $G'$ as follows: we first keep all the vertices of $G$ in $G'$ and

all the edges of $G$ as positive edges in $G'$; then, we add a new vertex $v$ to $G'$ and connect $v$ to all vertices in $G'$ with negative edges. It's clear that each maximal clique $C$ in $G$ corresponds a maximal balanced clique $\{\{v\}, C\}$ in $G'$ (assume $k = 1$), which means the maximal clique enumeration problem in $G$ can be reduced to the MBCE problem in $G'$. As the maximal clique enumeration problem is NP-Hard [53], [54], our MBCE problem is also NP-Hard.

Similarly, reconsidering $G$ and $G'$, the maximum clique $C^*$ in $G$ corresponds the maximum balanced clique $\{\{v\}, C^*\}$ in $G'$, and vice versa. As the maximum clique search problem is NP-Hard [26], [27], our MBCS problem is also NP-Hard.

## 4 A Baseline Algorithm for MBCE Problem

We first propose a baseline algorithm to address MBCE problem based on existing methods for maximal clique enumeration [12] and maximal biclique enumeration [55] in unsigned networks. For a signed network $G = (V, E^+, E^-)$, we can treat it as the combination of two unsigned networks $G^+ = (V, E^+)$ and $G^- = (V, E^-)$. For any maximal balanced clique $C = \{C_L, C_R\}$ in $G$, it is clear that $C_L$ (resp. $C_R$) is a clique in $G^+$ and the subgraph induced by vertices in $C_L$ and $C_R$ in $G^-$ is a biclique. Therefore, we can enumerate the maximal balanced cliques in $G$ in two steps: 1) compute all the maximal cliques in $G^+$ with [12]; 2) for each pair of the computed maximal cliques $C_i$ and $C_j$ in $G^+$, compute the maximal bicliques in the bipartite subgraph induced by the vertices in $C_i$ and $C_j$ in $G^-$ with [55]. The returned maximal bicliques in $G^-$ are the maximal balanced cliques in $G$.

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

- Memory consumption. Baseline has to store all the maximal cliques in $G^+$ in memory. The number of maximal cliques could be exponential to the number of vertices [11], which makes Baseline unable to handle large networks.
- Efficiency. In baseline, all the maximal cliques in $G^+$ are enumerated and every pair of maximal cliques are explored. The time complexity of Baseline is $O(T_{\mathsf{Cli}}+\eta^2 \cdot T_{\mathsf{BiCli}})$, where $T_{\mathsf{BiCli}}/T_{\mathsf{Cli}}$ represent the time complexity of maximal (bi)clique enumeration, and $\eta$ is the number of enumerated maximal cliques in $G^+$. Considering the maximal (bi)clique enumeration is time-consuming and the number of maximal cliques could be very large, it is inefficient for MBCE problem.

## 5 A New Enumeration Framework

Revisiting baseline, the root leading to its drawbacks discussed above is that it treats the signed network as a specific combination of two unsigned networks and utilizes the existing techniques designed for the unsigned networks. Therefore, we have to explore new techniques by considering the uniqueness of signed networks to overcome the drawbacks of Baseline and improve the efficiency of the enumeration. In this section, we present a new enumeration framework which aims to address the memory consumption problem. In next section, we further optimize the enumeration framework to improve the efficiency.

**Algorithm 1** MBCEnum ($G = (V, E^+, E^-), k$)

```
1: Flag ← true;
2: for each v_i ∈ {v_0, v_1, ··· , v_{n−1}} ∈ V do
3:    C_L ← {v_i}, C_R ← ∅
4:    P_L ← N_G^+(v_i) ∩ {v_{i+1}, ··· , v_{n−1}};
5:    P_R ← N_G^−(v_i) ∩ {v_{i+1}, ··· , v_{n−1}};
6:    Q_L ← N_G^+(v_i) ∩ {v_0, ··· , v_{i−1}};
7:    Q_R ← N_G^−(v_i) ∩ {v_0, ··· , v_{i−1}};
8:    MBCEnumUtil (C_L, C_R, P_L, P_R, Q_L, Q_R);
9: Procedure MBCEnumUtil (C_L, C_R, P_L, P_R, Q_L, Q_R)
10: if P_L = ∅ and P_R = ∅ and Q_L = ∅ and Q_R = ∅ then
11:    if |C_L| ≥ k and |C_R| ≥ k then
12:       output C = {C_L, C_R};
13:    return
14: Flag ←!Flag;
15: if Flag then
16:    for each v ∈ P_L do
17:       MBCEnumUtil(C_L ∪ {v}, C_R, N_G^+(v) ∩ P_L, N_G^−(v) ∩
          P_R, N_G^+(v) ∩ Q_L, N_G^−(v) ∩ Q_R);
18:       P_L ← P_L \ {v}; Q_L ← Q_L ∪ {v};
19:    for each v ∈ P_R do
20:       MBCEnumUtil(C_L, C_R ∪ {v}, N_G^−(v) ∩ P_L, N_G^+(v) ∩
          P_R, N_G^−(v) ∩ Q_L, N_G^+(v) ∩ Q_R);
21:       P_R ← P_R \ {v}; Q_R ← Q_R ∪ {v};
22: else
23:    line 19-21; line 16-18;
```

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

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

**Algorithm of** MBCEnum**.** Following the above idea, our algorithm for MBCE is shown in Algorithm 1. For each vertex $v_i$ in $G$ (line 2), we enumerate all the maximal balanced cliques containing $v_i$ (line 3-8). Note that $v_0, v_1, \ldots, v_n$ are in the degeneracy order [56] of $G$. We use $C_L$ and $C_R$ to maintain the balanced clique, which are initialized with $v_i$ and $\emptyset$, respectively (line 3). Similarly, we also initialize $P_L$ and $P_R$ as discussed above (line 4-5). Moreover, we use $Q_L$ and $Q_R$ to record the vertices that have been processed to avoid outputting duplicate maximal balanced cliques (line 6-7). After initializing these six sets, we invoke procedure MBCEnumUtil to enumerate all the maximal balanced cliques containing $v_i$ (line 8).

Procedure MBCEnumUtil performs the maximal balanced clique enumeration based on the given six sets. If $P_L$, $P_R$, $Q_L$ and $Q_R$ are empty, which means current balanced clique $C = \{C_L, C_R\}$ cannot be enlarged and it is a maximal

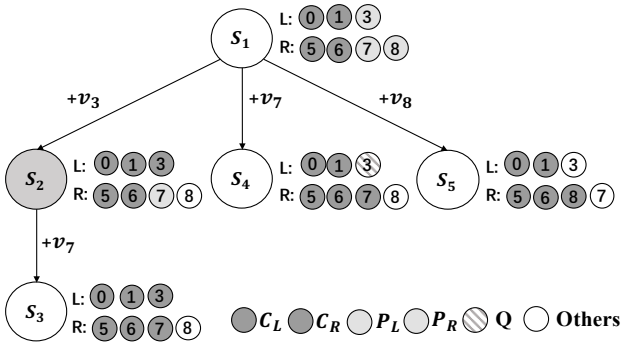



Fig. 3: Search Tree for MBCEnum

balanced clique, MBCEnumUtil checks whether $C_L$ and $C_R$ satisfy the size constraint. If the size constraint is satisfied, it outputs the maximal balanced clique $C$ (line 11-12). Otherwise, MBCEnumUtil adds a vertex from $P_L$ to $C_L$, updates the corresponding $P_L$, $P_R$, $Q_L$ and $Q_R$, and recursively invokes itself to further enlarge the balanced clique (line 17). When $v \in P_L$ is processed, $v$ is removed from $P_L$ and added in $Q_L$ (line 18). Similar processing steps are applied on vertices in $P_R$ (line 19-21). Variable Flag (line 1) is used to control the order of adding new vertex into $C_L$ or $C_R$. With the switch operation in line 14, we can guarantee that we add vertex into $C_L$, then into $C_R$, recursively.

**Example 5.1:** The enumeration procedure of MBCEnum can be illustrated as a search tree. Figure 3 shows part of the search tree when we conduct the MBCE on $G$ in Figure 2 through MBCEnum. $S_1, S_2, \ldots$ represent different search states during the enumeration. At $S_1$, we assume that we have a balanced clique with $C_L$=$\{v_0, v_1\}$, $C_R$=$\{v_5, v_6\}$}, $P_L$=$\{v_3\}$ and $P_R$=$\{v_7, v_8\}$ at this state. We first grow search branch by adding $v_3$ into $C_L$. At $S_2$, $P_L$ is empty, $P_R = \{v_7\}$, hence, we add $v_7$ into $C_R$ at $S_3$ and obtain $C_2 = \{\{v_0, v_1, v_3\}, \{v_5, v_6, v_7\}\}$. Now, the search branch from $v_3$ is finished, we return to $S_1$ state. Since $v_3$ has been explored, it is removed to $Q_L$, $P_L$ is empty now. Then, we add $v_7$ from $P_R$ to $C_R$ at $S_4$. Due to $C_2$ has been found at $S_3$, current result at $S_4$ is not maximal and can not be output. Next, we return to $S_1$ and find $C_3 = \{\{v_0, v_1\}, \{v_5, v_6, v_8\}\}$ by adding $v_8$ into $C_R$. Here, the search procedure at this search tree is finished. Other maximal balanced cliques can be found in a similar way. □

Based on Algorithm 1, it is clear that the memory consumption of our enumeration framework is linear to the size of the input signed network. Therefore, the drawback of large memory consumption in Baseline is avoided.

## 6 Enumeration Optimization Strategies

Although Algorithm 1 addresses the memory consumption problem in MBCE, the efficiency of Algorithm 1 is disappointing. In this section, we present two optimization strategies, namely in-enumeration optimization and pre-enumeration optimization, to further improve the efficiency of the enumeration.

### 6.1 In-Enumeration Optimization

**Branch Pruning.** Branch pruning aims to prune the unfruitful branches in the search tree of Algorithm 1 to improve the performance.

**Pivot Choosing.** Consider the maximal balanced clique search procedure of Algorithm 1, assume that we currently have $C_L$, $C_R$, $P_L$ and $P_R$, and we add a vertex $v$ from $P_L$ to $C_L$ in line 17. After finishing the search starting from $v$, we do not need to further explore the positive neighbors of $v$ in the for loop of line 16 and the negative neighbors of $v$ in the for loop of line 19. The reasons are as follows: w.o.l.g, let $v'$ be a positive neighbor of $v$, although we skip the maximal balanced clique search starting from $v'$, these maximal balanced cliques containing $v'$ must be explored by the searching branches starting $v$ or neighbors of $v'$. Therefore skipping the search starting from $v$'s neighbors does not affect the correctness of Algorithm 1.

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

**Candidate Selection.** In the search procedure of Algorithm 1, heuristically, search starting from a vertex with small local degree will have a short and narrow search branch, which means the search starting from the vertex will be finished very fast. Moreover, due to the search finish of the vertex, the vertex will be added into the excluded set and it can be used to further prune other search branches. Therefore, instead of adding vertices from $P_L$ and $P_R$ into $C_L$ and $C_R$ randomly in line 16 and 19 of Algorithm 1, we add vertices in the increasing order of their local degrees.

**Early Termination.** We consider different conditions that we can terminate the search early in Algorithm 1. For a balanced clique $C = \{C_L, C_R\}$, the maximal possible size of $C_L$ ($C_R$) for the final maximal balanced clique is $|C_L| + |P_L|$ ($|C_R| + |P_R|$). Based on the size constraint of $k$, we have the following rule:

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

In Algorithm 1, we use $Q_L$ and $Q_R$ to store such vertices that the maximal balanced cliques containing them have been enumerated. Therefore, during the enumeration, if there exists a vertex $v \in Q_L(Q_R)$ such that $P_L(P_R) \subseteq N_G^+(v)$ and $P_R(P_L) \subseteq N_G^-(v)$, then we can conclude that the maximal balanced cliques have been enumerated. Following this, we have our second rule:

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

In a certain search of Algorithm 1, if all the vertices in $P_L$ ($P_R$) consist a clique formed by positive edges and every vertex in $P_L$ ($P_R$) has negative edges to all the vertices in $P_R$ ($P_L$), then $P_L$ and $P_R$ consist a balanced clique. Then, based on Definition 3.2, $C_L \cup P_L$ and $C_R \cup P_R$ consist a maximal balanced clique. Therefore, we have our third early termination rule:

- **ET Rule 3:** If $\forall p_l \in P_L$, s.t., $P_L \subseteq \{\{p_l\} \cup N_G^+(p_l)\}$ and $P_R \subseteq N_G^-(p_l)$ and $\forall p_r \in P_R$, s.t., $P_R \subseteq \{\{p_r\} \cup N_G^+(p_r)\}$ and $P_L \subseteq N_G^-(p_r)$, we can output $C = (C_L \cup P_L, C_R \cup P_R)$ and terminate current search directly.

Note that, in order to avoid outputting duplicate maximal balanced cliques, ET Rule 3 must be applied after ET Rule 2.

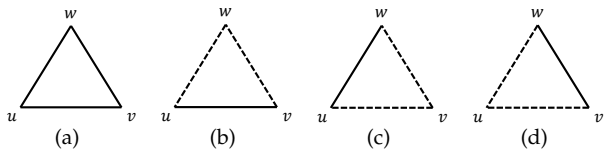



Fig. 4: Different types of common neighbors for $(u, v)$

**Algorithm of** MBCEnum$^*$**.** Utilizing the in-enumeration optimization strategies, we propose the optimized algorithm MBCEnum$^*$. The pseudocode is omitted here due to space constraints.

**Theorem 6.1:** *Given a signed network $G$, the time complexity of* MBCEnum$^*$ *is $O(\sigma n \cdot 3^{\sigma/3})$, where $\sigma$ is the degeneracy number of $G$.*

### 6.2 Pre-Enumeration Optimization

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

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

**Definition 6.1: ($(l, r)$-signed core)** Given a signed network $G = (V, E^+, E^-)$, two integers $l$ and $r$, a $(l, r)$-signed core is a maximal subgraph $\mathcal{C}$ of $G$, s.t., $\min_{v \in \mathcal{C}}\{d_{\mathcal{C}}^+(v)\} = l$, $\min_{v \in \mathcal{C}}\{d_{\mathcal{C}}^-(v)\} = r$. □

**Lemma 6.1:** *Given a signed network $G$ and threshold $k$, a maximal balanced clique satisfying the size constraint with $k$ is contained in a $(k-1, k)$-signed core.*

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

**Algorithm of** VertexReduction**.** Based on Definition 6.1, to compute the $(k-1, k)$-signed core in the signed network $G$, we only need to identify the vertices with $d_G^+(v) < k-1$ or $d_G^-(v) < k$ and remove them from $G$. Due to the removal of such vertices, more vertices will violate the degree constraints, we can further remove these vertices until no such kind of vertices exist in $G$.

**Theorem 6.2:** *Given a signed network $G$ and an integer $k$, the time complexity of* VertexReduction *is $O(n + m)$.*

**Edge Reduction.** In this part, we explore the opportunities to remove unpromising edges with respect to MBCE by considering the common neighbors of an edge formed by different types of edges. Specifically, for a positive/negative edge $(u, v)$, we define the edge common neighbor number:

**Definition 6.2: (Edge Common Neighbor Number)** Given a signed network $G = (V, E^+, E^-)$, for a positive edge $(u, v)$, we define:

- $\delta_G^{++}(u, v) = |\{w | (u, w) \in E^+ \wedge (v, w) \in E^+\}|$
- $\delta_G^{--}(u, v) = |\{w | (u, w) \in E^- \wedge (v, w) \in E^-\}|$

for a negative edge $(u, v)$, we define:

- $\delta_G^{+-}(u, v) = |\{w | (u, w) \in E^+ \wedge (v, w) \in E^-\}|$
- $\delta_G^{-+}(u, v) = |\{w | (u, w) \in E^- \wedge (v, w) \in E^+\}|$

□

Figure 4 shows the different types of common neighbors used in Definition 6.2. For a positive edge $(u, v)$, Figure 4 (a) and (b) show the common neighbor $w$ used in $\delta_G^{++}(u, v)$ and $\delta_G^{--}(u, v)$, respectively. For a negative edge $(u, v)$, Figure 4 (c) and (d) show the common neighbor $w$ used in $\delta_G^{+-}(u, v)$ and $\delta_G^{-+}(u, v)$, respectively. Note that $G$ is undirected and every edge is stored once in $G$. Based on Definition 6.2, we have the following lemma:

**Lemma 6.2:** *Given a signed network $G$ and an integer $k$, let $G'$ be the maximal sub-network of $G$ s.t.,*

*1) $\forall (u, v) \in E_{G'}^+ \rightarrow \delta_{G'}^{++}(u, v) \geq k-2 \wedge \delta_{G'}^{--}(u, v) \geq k$;*
*2) $\forall (u, v) \in E_{G'}^- \rightarrow \delta_{G'}^{+-}(u, v) \geq k-1 \wedge \delta_{G'}^{-+}(u, v) \geq k-1$;*

*then, every maximal balanced clique $C = \{C_L, C_R\}$ in $G$ satisfying the size constraint with $k$ is contained in $G'$.*

**Algorithm of** EdgeReduction**.** With Lemma 6.2, in order to enumerate the maximal balanced cliques in a given signed network $G$ with respect $k$, we only need to keep the edges in $G'$ shown in Lemma 6.2 and the positive/negative edges not in $G'$ can be safely pruned. We first compute $\delta_G^{++}(u, v)$ and $\delta_G^{--}(u, v)$ for each positive edge of $G$ and $\delta_G^{+-}(u, v)$ and $\delta_G^{-+}(u, v)$ for each negative edge of $G$. Following Lemma 6.2, for each positive edge $(u, v)$ such that $\delta_G^{++}(u, v) < k-2$ or $\delta_G^{--}(u, v) < k$, we remove $(u, v)$. After that, we decrease the corresponding edge common neighbor numbers that have been changed due to the removal of $(u, v)$ for the edge incident to $(u, v)$ based on Definition 6.2. It's similar to negative edges. The algorithm terminates when all the edges satisfy conditions in Lemma 6.2.

**Theorem 6.3:** *Given a signed network $G$, an integer $k$, the time complexity of* EdgeReduction *is $O(m^{1.5})$.*

## 7 Maximum Balanced Clique Search

Maximum clique search problem is a fundamental and hot research topic in graph analysis. In this section, we study the maximum balanced clique search problem.

### 7.1 A Baseline Approach

We first propose a baseline approach, namely MBCSear, to compute the maximum balanced clique in the input graph. We continuously enumerate the maximal balanced cliques in the input graph and maintain the maximum balanced clique $C^*$ found so far. For each search branch $(C_L, C_R, P_L, P_R)$, if $|C_L| + |C_R| + |P_L| + |P_R| \leq \epsilon$, where $\epsilon = |C^*|$, we can terminate the branch. When the enumeration finishes, it is easy to verify that $C^*$ is the maximum balanced clique.

**Drawbacks of** MBCSear**.** Although the straightforward approach can find the maximum balanced clique, the complexity of MBCSear is the same as that of MBCEnum$^*$ in the worst case. The search space of MBCSear is huge. In details, the drawbacks of MBCSear are twofold.

- **Lack of rigorous size constraints for $|C_L|$ and $|C_R|$.** Given a signed graph $G$, during the search process, MBCSear only holds the size constraint $|C_L| + |C_R| + |P_L| + |P_R| > \epsilon$ for each search branch. However, when $\epsilon$ is small, most of search branches have $|C_L| + |C_R| + |P_L| + |P_R|$ larger than $\epsilon$ which causes the fail of size constraint for most search branches. Unfortunately, as our algorithm constantly searches larger result than at present, the value of $\epsilon$ is gradually increasing from a small value, which makes MBCSear has to search the result with large search space.
- **Massive invalid search branches.** Although the search branches meet the size constraint with $\epsilon$, the structure between $P_L$ and $P_R$ maybe sparse which will generates invalid search branches. Hence, during the search process, more pruning techniques is needed urgently. Moreover, the optimization strategies based on $k$ in MBCEnum$^*$, like vertex reduction and edge reduction, are limited here, as $C^*$ usually has size much larger than $k$. Therefore, the remaining graph after reduction is still huge on large-scale signed network.

**Main idea.** In the further work, we aim to improve the efficiency of our algorithm.

- To address the first drawback of lacking of rigorous size constraints for $|C_L|$ and $|C_R|$, we can propose $\underline{\kappa}$ and $\overline{\kappa}$ as the lower bounds for $min\{|C_L|, |C_R|\}$ and $max\{|C_L|, |C_R|\}$, respectively. Then, we can get the balanced cliques with $|C_L| \geq \underline{\kappa}$ and $|C_R| \geq \overline{\kappa}$ within narrow search space (assume $|C_L| \leq |C_R|$). Under different value of $\underline{\kappa}$ and $\overline{\kappa}$, the search space is split into multiple partitions. Moreover, with initializing $\underline{\kappa}$ or $\overline{\kappa}$ as large value, we can search balanced cliques with large size as priority. Under large $\epsilon$ and rigorous bounds $\underline{\kappa}$ and $\overline{\kappa}$, the search space can be significantly reduced.
- To address the second drawback of massive invalid search branches, regarding to bounds $\epsilon$, $\underline{\kappa}$ and $\overline{\kappa}$, we can propose optimizations to forecast the size of balanced clique found in the current search branch to avoid invalid search branches and remove redundant vertices from candidates. Moreover, we can extend the vertex reduction and edge reduction of MBCEnum$^*$ with new bounds to prune more useless vertices and edges.

### 7.2 Search Space Partition-based Framework

To improve the efficiency of our approach, regarding to the first drawback of MBCSear, in this subsection, we propose a new maximum balance clique search framework MBCSear-SSP with two lower bounds $\underline{\kappa}$ and $\overline{\kappa}$ for $|C_L|$ and $|C_R|$. Given certain value $\underline{\kappa}_i$ and $\overline{\kappa}_i$, a search region is denoted as $(\underline{\kappa}_i, \overline{\kappa}_i)$, the maximum balanced clique found in it should satisfies $|C_L| \geq \underline{\kappa}_i$ and $|C_R| \geq \overline{\kappa}_i$ if $|C_L| \leq |C_R|$ (otherwise, swap $L$ and $R$). Under different value of $\underline{\kappa}$ and $\overline{\kappa}$, the whole search space can be divided into several search regions. In each search region, we keep searching larger result than at present. When all search regions are explored, the final result $C^*$ can be found.

As our main idea, to search the result with large size as priority, for the first search region $(\underline{\kappa}_0, \overline{\kappa}_0)$, $\overline{\kappa}_0$ is initialized as a large integer value. Obviously, as $\overline{\kappa}_0$ value is large, benefited from the strict size constraint, most of search branches of MBCSear are ineligible now. Hence the result can be found quickly in this search region. Besides, to obey the size threshold $k$, we make $\underline{\kappa}_0 = k$. Then, to cover the whole search space, for the later search regions, we keep increasing $\underline{\kappa}$ and decreasing $\overline{\kappa}$ until $\underline{\kappa} = \overline{\kappa}$. In another word, for $i < j$, $\underline{\kappa}_i \leq \underline{\kappa}_j$ and $\overline{\kappa}_i \geq \overline{\kappa}_j$.

**Algorithm 2** MBCSear-SSP ($G = (V, E^+, E^-), k$)

```
1: compute degeneracy σ of G+ = (V, E+);
2: ε ← 2k; κ ← k; κ̄ ← σ + 1; κ̄' ← −1;
3: while κ̄ ≥ κ and κ̄ < κ̄' do
4:     MBCSear (G = (V, E+, E−), ε) with adding size constraints: min{L, R} < κ or max{L, R} < κ̄;
5:     κ̄' ← κ̄; κ ← max{ε − κ̄, k}; κ̄ ← max{Dec(κ̄), κ};
```

Here, we first assign the possible maximum value to $\overline{\kappa}_0$, we have the following lemma:

**Lemma 7.1:** *Given a signed network $G = (V, E^+, E^-)$, for every balanced clique, we have $max\{|C_L|, |C_R|\} \leq \sigma + 1$, where $\sigma$ is the degeneracy number of $G^+$.*

*Proof.* In unsigned networks, the degeneracy number plus 1 is an upper bound for the maximum size of cliques [27]. Based on Definition 3.2, in a signed network $G = (V, E^+, E^-)$, for every balanced clique $C = \{C_L, C_R\}$, $C_L$ and $C_R$ are traditional cliques in $G^+$. Therefore, $|C_L|$ and $|C_R|$ are must not greater than $\sigma + 1$, respectively. □

Based on Lemma 7.1, we assign $(k, \sigma + 1)$ to the first search region $(\underline{\kappa}_0, \overline{\kappa}_0)$. Then, in the later search region, we continue to seek larger balanced clique than the current one. However, not every search region can find a valid result. To skip invalid search regions, we have the following lemma:

**Lemma 7.2:** *Given a signed network $G$, the maximum balanced clique found in the $i$-th search region $(\underline{\kappa}_i, \overline{\kappa}_i)$ is denoted by $C_i^* = \{C_L^i, C_R^i\}$. Then, for the next search region $(\underline{\kappa}_{i+1}, \overline{\kappa}_{i+1})$ , we have $\underline{\kappa}_{i+1} = |C_i^*| - \overline{\kappa}_i$.*

*Proof.* We prove it by contradiction. Following the $i$-th search region, in the next search region $(\underline{\kappa}_{i+1}, \overline{\kappa}_{i+1})$, if we get a larger balanced clique $C_{i+1}^* = \{C_L^{i+1}, C_R^{i+1}\}$ than $C_i^*$. Based on our search framework, we have $max\{|C_L^{i+1}|, |C_R^{i+1}|\} < \overline{\kappa}_i$, otherwise, $C_{i+1}^*$ will be found in the $i$-th search region rather than the $(i+1)$-th search region. Now, we assume $min\{|C_L^{i+1}|, |C_R^{i+1}|\} < |C_i^*| - \overline{\kappa}_i$. Combining with $max\{|C_L^{i+1}|, |C_R^{i+1}|\} < \overline{\kappa}_i$, we have $|C_{i+1}^*| = |C_L^{i+1}| + |C_R^{i+1}| < |C_i^*|$. Obviously, it is against with our premise that $C_{i+1}^*$ is larger than $C_i^*$. Therefore, the assumption for $min\{|C_L^{i+1}|, |C_R^{i+1}|\} < |C_i^*| - \overline{\kappa}_i$ does not hold. We get $min\{|C_L^{i+1}|, C_R^{i+1}\} \geq |C_i^*| - \overline{\kappa}_i$, i.e., $\underline{\kappa}_{i+1} = |C_i^*| - \overline{\kappa}_i$. □

Based on Lemma 7.2, after the $i$-th search region, the search regions with $\underline{\kappa} < |C_i^*| - \overline{\kappa}_i$ can be skipped directly.

**Algorithm of** MBCSear-SSP**.** Following the above idea, the new maximum balanced clique search algorithm MBCSear-SSP is shown at Algorithm 2. Given a signed network $G = (V, E^+, E^-)$ and size threshold $k$, we first compute the degeneracy number $\sigma$ of $G^+ = (V, E^+)$ (line 1). We initialize $\epsilon = 2k$, $\underline{\kappa} = k$, $\overline{\kappa} = \sigma + 1$ (line 2). Then, in each search region, Algorithm 2 invokes MBCSear to find the maximum balanced clique in current search region with adding size constraints: $min\{\overline{L}, \overline{R}\} < \underline{\kappa}$ or $max\{\overline{L}, \overline{R}\} < \overline{\kappa}$

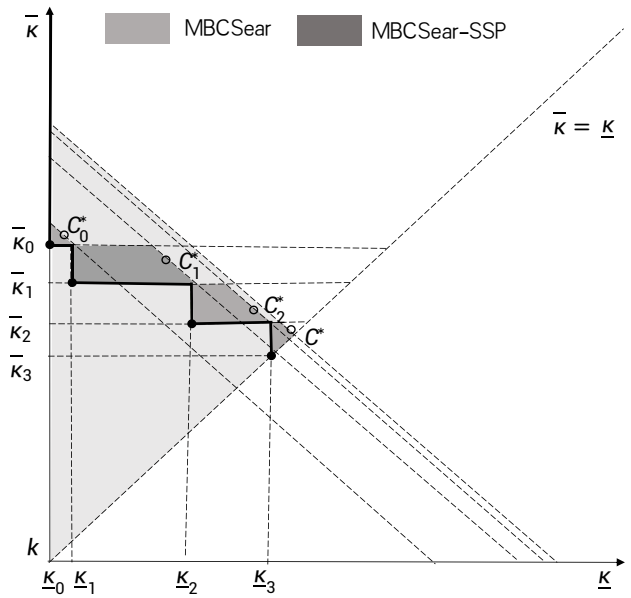



Fig. 5: The search space of MBCSear and MBCSear-SSP

on each search branch. For the next search region, we let $\underline{\kappa} = max\{\epsilon - \overline{\kappa}, k\}$ (line 5), since the size threshold $k$ is held for $C_L$ and $C_R$ as well. For the next value of $\overline{\kappa}$, $\overline{\kappa} = max\{Dec(\overline{\kappa}), \underline{\kappa}\}$ , where $Dec(\overline{\kappa})$ is used to decrease $\overline{\kappa}$. When $\overline{\kappa} < \underline{\kappa}$, let $\overline{\kappa}$=$\underline{\kappa}$. The search process finishes when $\overline{\kappa} = \underline{\kappa}$ and $\overline{\kappa}$ can not be reduced anymore, since if $\overline{\kappa}$ is unchanged, this search region is covered by the previous region already(line 3). When the search process of the final search region is finished, Algorithm 2 terminates and returns the maximum balanced clique $C^*$ in $G$.

Then, we discuss how $Dec(\overline{\kappa})$ decreases the value of $\overline{\kappa}$. The total running time of MBCSear-SSP can be formulated as $T = \Sigma_{i=0}^{\omega} t_i$, where $\omega$ is the number of search regions and $t_i$ is the partial running time of the $i$-th search region. In this paper, to keep the efficiency of MBCSear-SSP, we give a heuristic way to decrease $\overline{\kappa}$. In detail, if the last search region is time-consuming (set time threshold like 20 seconds), to make the total running time $T$ as small as possible, we reduce the amount of search regions $\omega$ by decreasing $\overline{\kappa}$ by a large value, i.e., $Dec(\overline{\kappa}) = \lceil \frac{\overline{\kappa}}{2} \rceil$, otherwise, $Dec(\overline{\kappa}) = \overline{\kappa} - 2$.

Now, we compare the search space between MBCSear and MBCSear-SSP. As shown at Figure 5, since $C^*$ is the maximum balanced clique, MBCSear searches all balanced cliques with size less than $C^*$ until $C^*$ is found. The search space of MBCSear is shown at Figure 5. For the search space of MBCSear-SSP, benefited from the tight bounds for $|C_L|$ and $|C_R|$ at each search region, MBCSear-SSP searches local maximum balanced clique within small search space. For instance, as shown at Figure 5, it finds the current maximum balanced clique $C_0^*$ in the first search region. Then, in the second search region, it searches balanced cliques with $\overline{\kappa}_1 \leq |C_R| < \overline{\kappa}_0$ and $|C_L| \geq \overline{\kappa}_1$ (assume $|C_L| \leq |C_R|$) until $C_1^*$ is found. As shown at Figure 5, the search space of MBCSear-SSP is much smaller than MBCSear.

**Example 7.1:** Reconsidering the signed network $G$ in Figure 2, $k = 2$, Algorithm 2 first computes degeneracy number $\sigma$ of $G^+$, get $\sigma = 2$. So, the first search region is $(2, 3)$. Algorithm 2 find result $C_0^* = \{\{v_0, v_1, v_3\}, \{v_5, v_6, v_7\}\}$ at first, $\epsilon = 6$. Then, based on Lemma 7.2, $\underline{\kappa}_1$ is 3. As we should keep $\overline{\kappa} \geq \underline{\kappa}$, $\overline{\kappa}_1$ is 3 as well. However, as the value of $\overline{\kappa}$ is unchanged, this search region is covered by the first search region. Hence, Algorithm 2 is terminated and returns $C_0^*$ as $C^*$. Comparing with MBCSear, the search space of MBCSear-SSP is reduced from $(2, 2)$ to $(2, 3)$. □

### 7.3 Optimization Strategies

Regarding to the second drawback of MBCSear on massive invalid search branches in each search region, in this subsection, we explore the chance to further improve the efficiency of our approach. Under our rigorous lower bounds $\underline{\kappa}$ and $\overline{\kappa}$, we first propose two optimization strategies, coloring-based branch pruning and vertex domination-based candidate pruning, to prune invalid search branches and remove meaningless vertices from candidates. Then, we extend the vertex&edge reduction technologies of MBCE to prune more unnecessary vertices and edges in advance.

#### *7.3.1 Coloring-based Branch Pruning*

Given a search region $(\underline{\kappa}, \overline{\kappa})$ and a search branch, if the upper bound of the balanced clique size in current search branch is less than the lower bounds $\underline{\kappa}$, $\overline{\kappa}$ and $\epsilon$, current search branch can be pruned directly. Looking back to MBCSear, it uses the candidates size to form the upper bound. However, this upper bound is too loose, because although the number of candidates is large, the connectivity between candidates maybe sparse, which will lead to many invalid search branches. Hence, now, we aim to propose a tighter upper bound based on vertex coloring.

**Definition 7.1: (Vertex Coloring [57])** Given a graph $G$, vertex coloring in $G$ aims to assign colors to each vertex such that vertices are different in color from their neighbors. The amount of colors needed in $G$ is named chromatic number, denoted by $\gamma(G)$. □

**Lemma 7.3:** *Given a search branch $(C_L, C_R, P_L, P_R)$, the maximum balanced clique from this branch is denoted as $C' = \{C'_L, C'_R\}$, we have that $|C'_L| \leq \gamma(\mathbb{G}_L) + |C_L|$, $|C'_R| \leq \gamma(\mathbb{G}_R) + |C_R|$, where $\mathbb{G}_{L(R)}$ is the positive subgraph produced by $P_{L(R)}$.*

*Proof.* Given a graph $G$, the chromatic number $\gamma(G)$ is an upper bound of the maximum size of cliques in $G$ [27]. Based on it, the lemma can be proved. □

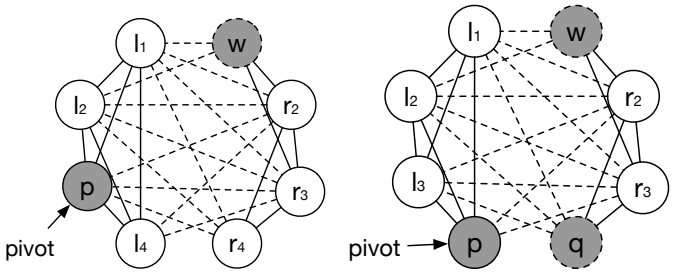
Based on Lemma 7.3, if the upper bound does not meet the size requirements of current search region, i.e., $min\{\gamma(\mathbb{G}_L) + |C_L|, \gamma(\mathbb{G}_R) + |C_R|\} < \underline{\kappa}$ or $max\{\gamma(\mathbb{G}_L) + |C_L|, \gamma(\mathbb{G}_R) + |C_R|\} < \overline{\kappa}$ or $\gamma(\mathbb{G}_L) + |C_L| + \gamma(\mathbb{G}_R) + |C_R| \leq \epsilon$, the search branch can be early terminated directly.

**Algorithm of** ColoringPrune. We propose ColoringPrune algorithm to prune search branches. The pseudocode is shown at Algorithm 3. It first computes $\gamma(\mathbb{G}_L)$ and $\gamma(\mathbb{G}_R)$ (line 1-8). Then it returns true if the upper bound does not meet the size requirements, which means this search branch can be pruned directly, otherwise, returns false (line 9-11).

**Theorem 7.1:** *The space complexity of Algorithm 3 is $O(|P_L| + |P_R| + |E_{\mathbb{G}_L}| + |E_{\mathbb{G}_R}|)$, the time complexity is $O(|P_L| + |P_R| + |E_{\mathbb{G}_L}| + |E_{\mathbb{G}_R}|)$.*

**Algorithm 3** ColoringPrune($C_L, C_R, P_L, P_R, \epsilon, \underline{\kappa}, \overline{\kappa}$)

```
1: 𝔾_L ← G⁺(P_L); 𝔾_R ← G⁺(P_R);
2: γ(𝔾_{L(R)}) ← 0; col(v) ← 0 for each v ∈ P_{L(R)};
3: for each v ∈ P_{L(R)} do
4:     col(v) ← 1;
5:     while ∃u ∈ N_{𝔾_{L(R)}}(v), s.t., col(u) = col(v) do
6:         col(v) ← col(v) + 1;
7:     if col(v) > γ(𝔾_L) then
8:         γ(𝔾_{L(R)}) ← γ(𝔾_{L(R)}) + 1;
9: if min{γ(𝔾_L) + |C_L|, γ(𝔾_R) + |C_R|} < κ or max{γ(𝔾_L) +
   |C_L|, γ(𝔾_R)+|C_R|} < κ̄ or γ(𝔾_L)+|C_L|+γ(𝔾_R)+|C_R| ≤ ϵ
   then
10:    return true;
11: return false;
```




(a) Case 1&3 (b) Case 2&4

Fig. 6: Vertex Domination Cases

Note that Algorithm 3 can be directly applied to MBCE problem with $\underline{\kappa} = k$, $\overline{\kappa} = k$, $\epsilon = 2k$. However, since $k$ is usually small, the effectiveness of Algorithm 3 is limited in MBCE problem.

#### 7.3.2 Vertex Domination-based Candidate Pruning

To further improve the efficiency, we reduce the number of candidates in $P_L$ and $P_R$ at each search branch by pruning invalid vertices from candidates. Our key thought is that if we have the prior knowledge to know that vertex $v$ forms balanced clique with size larger than that of vertex $u$, then, $u$ is dominated by $v$, denoted by $u \in \Phi_v$, the search relevant to $u$ can be skipped. Simplely, for each search branch, we can use the local neighborhood between candidates to figure out the domination relationship, i.e., if $N_l(u) \subseteq N_l(v)$, then $u \in \Phi_v$. Then, we have:

**Lemma 7.4:** *Given a signed network $G$ and a search branch with candidate sets $P_L$ and $P_R$, for each vertex $v \in P_L \cup P_R$, if $v$ is dominated, the sub-search branch from $v$ can be skipped.*

*Proof.* Given a search branch, for vertices $u, v$ in candidates such that $u \in \Phi_v$, we use $C_v$ and $C_u$ to indicate the balanced cliques maintaining $v$ and $u$, respectively. Due to $N_l(u) \subseteq N_l(v)$, then $C_u \setminus \{u\} \subseteq C_v \setminus \{v\}$, i.e., $|C_u| \leq |C_v|$. Hence $u$ won't belong to any balanced clique larger than $C_v$. The search from $u$ can be skipped. □

The vertex domination can be computed with adjacency list join operation. The time complexity of computing vertex domination at single search branch is $O(\mathrm{C}^2_{\ltimes} \cdot (\ltimes - 1)) = O(\ltimes^3)$, where $\ltimes = |P_L| + |P_R|$, which is time-consuming. Moreover, the amount of search branches is huge. To obtain the vertex domination effectively, in this paper, we only consider four special cases based on pivot technique, which can be computed within const time. The four special cases are introduced as follows, $p$ is the selected pivot.

- **Case 1:** If $p \in Q_L \cup Q_R$ and $newP_L \cup newP_R = \{w\}$, then $\Phi_p = \{w\}$.
- **Case 2:** If $p \in Q_L \cup Q_R$, $newP_L \cup newP_R = \{w, q\}$ and $(w, q) \notin E$, then, $\Phi_p = \{w, q\}$.
- **Case 3:** If $p \in P_L \cup P_R$, $newP_L \cup newP_R = \{p, w\}$, then, $\Phi_p = \{w\}$.
- **Case 4:** If $p \in P_L \cup P_R$, $newP_L \cup newP_R = \{p, w, q\}$ and $(w, q) \notin E$, then, $\Phi_p = \{w, q\}$.

Figure 6 shows the four special cases of vertex domination, respectively. For case 1&3(Figure 6(a)), $p$ is selected as pivot, then, the surviving candidate is $w$ as other vertices are $p$'s neighbors. Since $N_l(w) \subseteq N_l(p)$, $w$ is dominated by $p$. For case 2&4(Figure 6(b)), $w$ and $q$ are two surviving candidates with pivot $p$, as $(w, q) \notin E$, they will not appear at a common balanced clique, hence, $N_l(w) \subseteq N_l(p)$, $N_l(q) \subseteq N_l(p)$, $\Phi_p = \{w, q\}$.

Based on Lemma 7.4, when we meet the above four special cases of vertex domination, we skip the searches from vertices in $\Phi_p$ by deleting them from candidate sets, this process only consumes const time. In this way, the invalid candidates can be further pruned effectively.

#### 7.3.3 Vertex&Edge Reduction Variants

Although the optimization strategies proposed in MBCE algorithm, like vertex reduction and edge reduction, are still applicable for MBCSear, they only ensure that $C_L$ and $C_R$ are not less than $k$ but lack of binding force of our new size bounds $\underline{\kappa}$, $\overline{\kappa}$ and $\epsilon$. Moreover, the value of $\overline{\kappa}$, $\underline{\kappa}$ and $\epsilon$ are much larger than $k$, especially for $\overline{\kappa}$ and $\epsilon$, which makes the effectiveness of the reduction optimizations is limited in MBCS problem. Hence, we extend the vertex reduction and edge reduction such that they can support tighter bounds to prune more vertices and edges.

**Vertex reduction variant.** We first propose the vertex reduction variant with considering the degree of vertices. We have the following lemma:

**Lemma 7.5:** *Given a signed network $G = (V, E^+, E^-)$, a search region $(\underline{\kappa}_i, \overline{\kappa}_i)$ and $\epsilon$, $\mathbb{G} = (\mathbb{V}, \mathbb{E}^+, \mathbb{E}^-)$ is a subgraph of $G$, s.t., $(a)\forall v \in \mathbb{V}, min\{d^+_{\mathbb{G}}(v) + 1, d^-_{\mathbb{G}}(v)\} \geq \underline{\kappa}_i, max\{d^+_{\mathbb{G}}(v) + 1, d^-_{\mathbb{G}}(v)\} \geq \overline{\kappa}_i$; $(b)\forall v \in \mathbb{V}, d^+_{\mathbb{G}}(v) + d^-_{\mathbb{G}}(v) + 1 > \epsilon$. We have $C^*_i \subseteq \mathbb{G}$.*

*Proof.* Given a search region $(\underline{\kappa}_i, \overline{\kappa}_i)$, let $C^*_i = \{C_L, C_R\}$ is the maximum balanced clique in current search region. Based on Algorithm 2, $C^*_i = \{C_L, C_R\}$ should satisfy $min\{|C_L|, |C_R|\} \geq \underline{\kappa}_i$ and $max\{|C_L|, |C_R|\} \geq \overline{\kappa}_i$. Meanwhile, based on Definition 3.2, $\forall v \in C_L$, $d^+_{C^*_i}(v) = |C_L| - 1$, $d^-_{C^*_i}(v) = |C_R|$. $\forall v \in C_R$, $d^+_{C^*_i}(v) = |C_R| - 1$, $d^-_{C^*_i}(v) = |C_L|$. Combining them, we get $min\{d^+_{C^*_i}(v) + 1, d^-_{C^*_i}(v)\} \geq \underline{\kappa}$, $max\{d^+_{C^*_i}(v) + 1, d^-_{C^*_i}(v)\} \geq \overline{\kappa}$. Moreover, as $C^*_i$ is the current maximum balanced clique, the degree of vertices in $C^*_i$ should not less than $\epsilon$. □

**Algorithm of** VertexReduction$^+$**.** Based on Lemma 7.5, we can reduce the size of the candidate sets by continuously deleting the vertices that do not meet the degree constraints in Lemma 7.5. We propose VertexReduction$^+$ algorithm, the pseudocode is shown at Algorithm 4. It continuously

**Algorithm 4** VertexReduction$^+$ $(C_L, C_R, P_L, P_R, \epsilon, \underline{\kappa}, \overline{\kappa})$

1: $V' \leftarrow P_L \cup P_R$; $\mathbb{G} \leftarrow G(V')$ ;
2: **while** $\exists v \in P_{L(R)}$, s.t. $min\{d^+_{\mathbb{G}}(v)+1+|C_{L(R)}|, d^-_{\mathbb{G}}(v)+|C_{R(L)}|\} < \underline{\kappa}$ or $max\{d^+_{\mathbb{G}}(v)+1+|C_{L(R)}|, d^-_{\mathbb{G}}(v)+|C_{R(L)}|\} < \overline{\kappa}$ or $d^+_{\mathbb{G}}(v)+1+d^-_{\mathbb{G}}(v)+|C_L|+|C_R| \leq \epsilon$ **do**
3: **for each** $u \in N^+_{\mathbb{G}}(v)$ **do**
4: $d^+_{\mathbb{G}}(u) \leftarrow d^+_{\mathbb{G}}(u) - 1$;
5: **for each** $u \in N^-_{\mathbb{G}}(v)$ **do**
6: $d^-_{\mathbb{G}}(u) \leftarrow d^-_{\mathbb{G}}(u) - 1$;
7: $\mathbb{G} \leftarrow \mathbb{G} \setminus v$;
8: $P_{L(R)} \leftarrow P_{L(R)} \setminus v$;

removes vertices until all vertices left in candidate sets meet the degree constraints.

**Theorem 7.2:** *The space complexity of Algorithm 4 is* $O(|P_L| + |P_R| + |E^+_{\mathbb{G}}| + |E^-_{\mathbb{G}}|)$. *The time complexity of Algorithm 4 is* $O(|P_L| + |P_R| + |E^+_{\mathbb{G}}| + |E^-_{\mathbb{G}}|)$, *where* $\mathbb{G} = G(P_L \cup P_R)$.

**Edge reduction variant.** After the vertex reduction variant, we extend the edge reduction now. Inspired by the edge reduction technology utilized in MBCE, we continue to explore the edge reduction under a certain search region. Reconsidering the edge common neighbor number introduced at Definition 6.2, we have the following lemma:

**Lemma 7.6:** *Given a signed network* $G$, *a search region* $(\underline{\kappa}_i, \overline{\kappa}_i)$ *and* $\epsilon$, $\mathbb{G} = (\mathbb{V}, \mathbb{E}^+, \mathbb{E}^-)$ *is a subgraph of* $G$, *s.t.,*

- $\forall (u,v) \in \mathbb{E}^+ \rightarrow min\{\delta^{++}_{\mathbb{G}}(u,v)+2, \delta^{--}_{\mathbb{G}}(u,v)\} \geq \underline{\kappa}_i \wedge max\{\delta^{++}_{\mathbb{G}}(u,v)+2, \delta^{--}_{\mathbb{G}}(u,v)\} \geq \overline{\kappa}_i$;
- $\forall (u,v) \in \mathbb{E}^- \rightarrow min\{\delta^{+-}_{\mathbb{G}}(u,v)+1, \delta^{-+}_{\mathbb{G}}(u,v)+1\} \geq \underline{\kappa}_i \wedge max\{\delta^{+-}_{\mathbb{G}}(u,v)+1, \delta^{-+}_{\mathbb{G}}(u,v)+1\} \geq \overline{\kappa}_i$.

*Then, we have* $C^*_i \subseteq \mathbb{G}$

**Algorithm of** EdgeReduction$^+$**.** Based on Lemma 7.6, we propose EdgeReduction$^+$ algorithm. Given a signed network $G = (V, E^+, E^-)$, a search region $(\underline{\kappa}_i, \overline{\kappa}_i)$ and $\epsilon$, before the search starts, it removes the invalid edges that do not meet the requirements for the edge common neighbor number in Lemma 7.6 until no more edges can be pruned. The pseudocode is omitted here. The time complexity of EdgeReduction$^+$ is $O(m^{1.5})$.

### 7.3.4 The Optimized Algorithm

Utilizing the above optimization strategies, i.e., coloring-based branch pruning, vertex domination-based candidate pruning and vertex&edge reduction variants, we propose our optimized algorithm MBCSear$^*$ to search maximum balanced clique in a given search region. The pseudocode is shown at Algorithm 5. Given a search region $(\underline{\kappa}, \overline{\kappa})$, for each search branch in this region, it first prunes candidates in $P_L$ and $P_R$ by invoking VertexReduction$^+$ algorithm (line 4). Then, the surviving candidate sets are judged to see whether it meets the size requirements (line 5-7). If the candidate sets and explored sets are both empty, we get a larger result and return it (line 8-11). Otherwise, Algorithm 5 tries to prune invalid branch by invoking ColoringPrune (line 12-13). After that, it chooses the pivot and distinguishes the four special cases for vertex domination to further prune candidates (line 18-27). Then, it continuously search larger balanced clique in the remaining candidates by recursively calling itself. When all search branches are finished, Algorithm 5 can get the maximum balanced clique in the given search region.

**Algorithm 5** MBCSear$^*$ $(G = (V, E^+, E^-), \epsilon)$

1: **for each** $v \in V$ **do**
2: initialize $C_L, C_R, P_L, P_R, Q_L, Q_R$;
3: MBCSearUtil$^*$ $(C_L, C_R, P_L, P_R, Q_L, Q_R, \epsilon)$;

**Procedure** MBCSearUtil$^*$ $(C_L, C_R, P_L, P_R, Q_L, Q_R, \epsilon)$
// **Vertex Reduction Varint**
4: VertexReduction$^+$$(C_L, C_R, P_L, P_R, \epsilon, \underline{\kappa}, \overline{\kappa})$;
5: $\overline{L} \leftarrow |C_L| + |P_L|$; $\overline{R} \leftarrow |C_R| + |P_R|$;
6: **if** $\overline{L} + \overline{R} \leq \epsilon$ or $min\{\overline{L}, \overline{R}\} < \underline{\kappa}$ or $max\{\overline{L}, \overline{R}\} < \overline{\kappa}$ **then**
7: **return**;
8: **if** $P_L = \emptyset$ and $P_R = \emptyset$ and $Q_L = \emptyset$ and $Q_R = \emptyset$ **then**
9: $C^* \leftarrow \{C_L, C_R\}$;
10: $\epsilon \leftarrow |C_L| + |C_R|$;
11: **return**;
// **Coloring-based Branch Pruning**
12: **if** ColoringPrune$(C_L, C_R, P_L, P_R, \epsilon, \underline{\kappa}, \overline{\kappa})$ **then**
13: **return**;
14: $p \leftarrow$ argmax$_{v \in P_L \cup P_R \cup Q_L \cup Q_R}\{d_l(v)\}$;
15: // assume $p$ from $P_L \cup Q_L$ ($P_R \cup Q_R$ )
16: $newP_L \leftarrow P_L \setminus N^{+(-)}_G(p)$;
17: $newP_R \leftarrow P_R \setminus N^{-(+)}_G(p)$;
// **Vertex Domination-based Candidate Pruning**
// **Case 1**
18: **if** $p \in Q_{L(R)}$ and $|newP_L| + |newP_R| = 1$ **then**
19: **return**;
// **Case 2**
20: **if** $p \in Q_{L(R)}$ and $|newP_L| + |newP_R| = 2$ **then**
21: **if** $w, q \in newP_L \cup newP_R$ and $(w,q) \notin E$ **then**
22: **return**;
// **Case 3**
23: **if** $p \in P_{L(R)}$ and $|newP_L| + |newP_R| = 2$ **then**
24: $newP_{L(R)} \leftarrow \{p\}$; $newP_{R(L)} \leftarrow \emptyset$;
// **Case 4**
25: **if** $p \in P_{L(R)}$ and $|newP_L| + |newP_R| = 3$ **then**
26: **if** $w, q \in newP_L \cup newP_R$ and $(w,q) \notin E$ **then**
27: $newP_{L(R)} \leftarrow \{p\}$; $newP_{R(L)} \leftarrow \emptyset$;
28: search result within $newP_L$ and $newP_R$ as MBCEnum;

**Algorithm 6** MBCSear-SSP$^*$ $(G = (V, E^+, E^-), k)$

1: compute degeneracy $\sigma$ of $G^+ = (V, E^+)$;
2: $\epsilon \leftarrow 2k$; $\underline{\kappa} \leftarrow k$; $\overline{\kappa} \leftarrow \sigma + 1$; $\overline{\kappa}' \leftarrow -1$;
3: **while** $\overline{\kappa} \geq \underline{\kappa}$ and $\overline{\kappa} < \overline{\kappa}'$ **do**
4: $G' \leftarrow$ EdgeReduction$^+(G, \epsilon, \underline{\kappa}, \overline{\kappa})$;
5: MBCSear$^*$ $(G', \epsilon)$;
6: $\overline{\kappa}' \leftarrow \overline{\kappa}$; $\underline{\kappa} \leftarrow max\{\epsilon - \overline{\kappa}, k\}$; $\overline{\kappa} \leftarrow max\{\lceil \frac{\overline{\kappa}}{2} \rceil, \underline{\kappa}\}$;

Based on MBCSear$^*$ algorithm, we are ready to propose the formal optimized algorithm MBCSear-SSP$^*$, the pseudocode is shown at Algorithm 6. Given a signed network $G = (V, E^+, E^-)$ and size threshold $k$, for each search region $(\overline{\kappa}, \underline{\kappa})$, the algorithm first invokes EdgeReduction$^+$ to reduce the graph size by removing invalid edges before search start (line 4). Then, it invokes Algorithm 5 to search the maximum balanced clique in current search region (line 5). In the end, when finish the search in all search regions, it gets the maximum balanced clique $C^*$ in $G$ and terminates.

## 8 Performance Studies

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

**Algorithms.** We evaluate MBCE algorithms and MBCS algorithms.

TABLE 1: Statistic for real datasets

| Dataset | $n$ | $m$ | $\|E^+\|$ | $\|E^-\|$ |
|---|---|---|---|---|
| AdjWordNet | 21,247 | 426,896 | 378,993 | 47,903 |
| Slashdot | 77,357 | 516,575 | 396,378 | 120,197 |
| Epinions | 131,828 | 841,372 | 717,667 | 123,705 |
| DBLP | 1,314,050 | 5,179,945 | 1,471,903 | 3,708,042 |
| Douban | 1,588,565 | 13,918,375 | 9,034,537 | 4,883,838 |
| Pokec | 1,632,803 | 30,622,564 | 15,179,203 | 7,122,761 |
| Livejournal | 4,847,571 | 42,851,237 | 29,105,031 | 13,746,206 |
| Orkut | 3,072,441 | 117,184,899 | 79,664,169 | 37,520,730 |
| Dbpedia | 18,268,992 | 126,890,209 | 86,002,736 | 40,887,473 |

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

For MBCS algorithms, they are MBCSear, MBCSear-SSP and MBCSear-SSP*. MBCSear is the baseline approach introduced at Section 7.1. MBCSear-SSP is proposed at Section 7.2. MBCSear-SSP* is the improved algorithm shown at Section 7.3. Note that, for fair, we apply the EdgeReduction$^+$ proposed at Section 7.3.3 to both MBCSear-SSP and MBCSear-SSP*.

All algorithms are implemented in C++, using g++ complier with -O3. The time cost is measured as the amount of wall-clock time elapsed during the program's execution. If an algorithm cannot finish in 12 hours, we denote the processing time as INF.

**Real datasets.** We evaluate our algorithms on nine real datasets. Slashdot and Epinions are signed networks in real world. AdjWordNet, DBLP and Douban are signed networks used in [58], [40] and [59], respectively. For other datasets, we transfer them from unsigned network to signed network. In order to simulate the balanced clique as much as possible, the vertices in the graph are divided into two groups with a ratio of 4:1, the edges connecting vertices from the same group are positive edges, otherwise, they are negative edges. In this way, all the cliques in the original graph correspond to balanced cliques in the signed graph. For data source, AdjWordNet is downloaded from WordNet (https://wordnet.princeton.edu/). DBLP and Dbpedia are downloaded from KONECT (http://konect.cc/). Douban is from authors in [59]. Other datasets are downloaded from SNAP (http://snap.stanford.edu). The details of each dataset are shown in Table 1.

### 8.1 The Performance of MBCE Algorithms

**Exp-1: Efficiency of MBCE algorithms when varying $k$.** In this experiment, we evaluate the efficiency of three algorithms when varying $k$ from 4 to 10 and the results are shown in Figure 7.

As shown in Figure 7, Baseline consumes the most time among three algorithms on all datasets when we vary $k$ and it can only handle the small datasets. MBCEnum is faster than Baseline on most of the test cases as MBCEnum takes the uniqueness of the signed networks into consideration and enumerates the maximal balanced cliques based on the signed network directly. MBCEnum* is the most efficient algorithm on all datasets when varying $k$ due to the utilization of in-enumeration optimization strategies, which reveals the effectiveness of in-enumeration optimization strategies. Another phenomena shown in Figure 7 is that the running time of all algorithms decreases as $k$ increases. This is because as $k$ increases, the pruning power of the optimization strategies proposed in Section 6 strengthens.

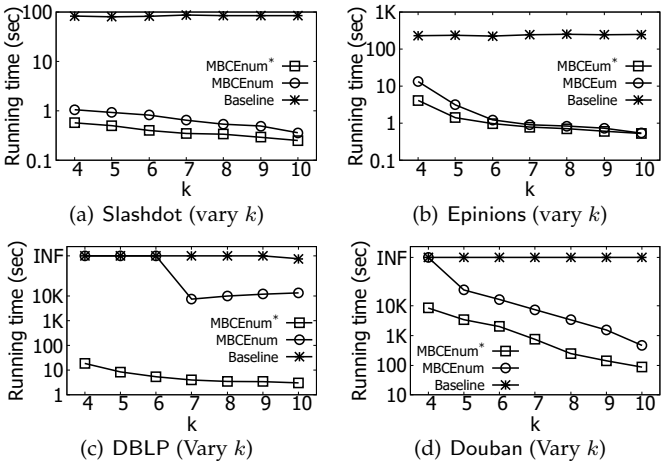



Fig. 7: Running time of MBCE algorithms varying $k$

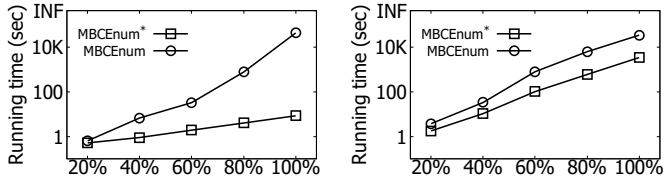



Fig. 8: Scalability of MBCEnum and MBCEnum*, $k$=4

**Exp-2: Scalability of MBCE algorithms.** In this experiment, we test the scalability of MBCEnum and MBCEnum* on two large datasets DBLP and Douban by varying their vertices from 20% to 100%. Figure 8 shows the results.

As shown in Figure 8, when $n$ increases, the running time of both algorithms increases as well, but MBCEnum* outperforms MBCEnum for all cases on both datasets. For example, on DBLP, when we sample $20\%$ vertices, the running time of MBCEnum and MBCEnum* is 0.6 seconds and 0.5 seconds, respectively, while when sampling $80\%$ vertices, their running times are 770.6 seconds and 4.0 seconds, respectively. It shows that MBCEnum* has a good scalability in practice.

**Exp-3: Case study on AdjWordNet.** In this experiment, we perform a case study on the real dateset AdjWordNet. In this dataset, two synonyms have a positive edge and two antonyms have a negative edge, and Table 2 shows some results obtained by our algorithm. As shown in Table 2, words in $C_L$ or $C_R$ have similar meaning while each word from $C_L$ is an antonym to all words in $C_R$. This case study verifies that maximal balanced clique enumeration can be applied in the applications to find synonym and antonym groups on dictionary data.

TABLE 2: Case study on AdjWordNet

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

### 8.2 The Performance of MBCS Algorithms

**Exp-4: Efficiency of** MBCS **algorithms when varying** $k$**.** To evaluate the efficiency of MBCS algorithms, we record the running time of them on eight datasets, $k = [2 - 10]$, the results are shown at Figure 9.

As shown at Figure 9, with the value of $k$ increasing, the running time of three algorithms decreases on most datasets. For each value of $k$, MBCSear-SSP$^*$ is the fastest algorithm of the three algorithms, while MBCSear is the most time-consuming. Moreover, on large graphs as Livejournal, Orkut and Dbpedia, when $k$=2, MBCSear and MBCSear-SSP can not get the result within a reasonable time, only MBCSear-SSP$^*$ can get the result. The running time of MBCSear-SSP$^*$ on the three graphs are 413.4s, 3626.0s and 1833.6s, respectively. It's because that MBCSear has to search on the whole graph, while MBCSear-SSP uses search space partition to search the maximum balanced clique within a partial subgraph. Based on MBCSear-SSP, benefited from multiple optimization strategies for further reducing the search space, MBCSear-SSP$^*$ is the most efficient algorithm of them, it can efficiently search result on all datasets.

**Exp-5: Effectiveness of** MBCS **algorithms when varying** $k$**.** To intuitively compare the effectiveness of three MBCS algorithms, in this experiment, we record the amount of calculation of three algorithms on eight datasets, $k = [2 - 5]$. The calculation quantity is the time of invoking MBCSearUtil and MBCSearUtil$^*$, which can intuitively represent the search space of different algorithms. The experimental results are shown at Figure 10.

As shown at Figure 10(a), when $k = 2$, on all datasets, the calculation quantity of MBCSear-SSP$^*$ is much less than that of other algorithms. Meanwhile, MBCSear-SSP's calculation quantity is less than MBCSear's. For instance, on DBLP(DB), the calculation quantity of MBCSear, MBCSear-SSP and MBCSear-SSP$^*$ are 155,621, 328 and 183, respectively. When $k > 2$, the trend is similar. It's because MBCSear-SSP$^*$ and MBCSear-SSP are based on search space partitions, which can reduce the total search space effectively. The experimental results also confirm the reason for the efficiency of MBCSear-SSP$^*$ at Exp-4.

**Exp-6: Search process of** MBCSear-SSP$^*$ **on real datasets.** In this experiment, we show the search process of MBCSear-SSP$^*$ algorithm on Douban and Pokec datasets, $k$=2. Table 3 shows every search region $(\overline{\kappa}, \underline{\kappa})$, the size of its input

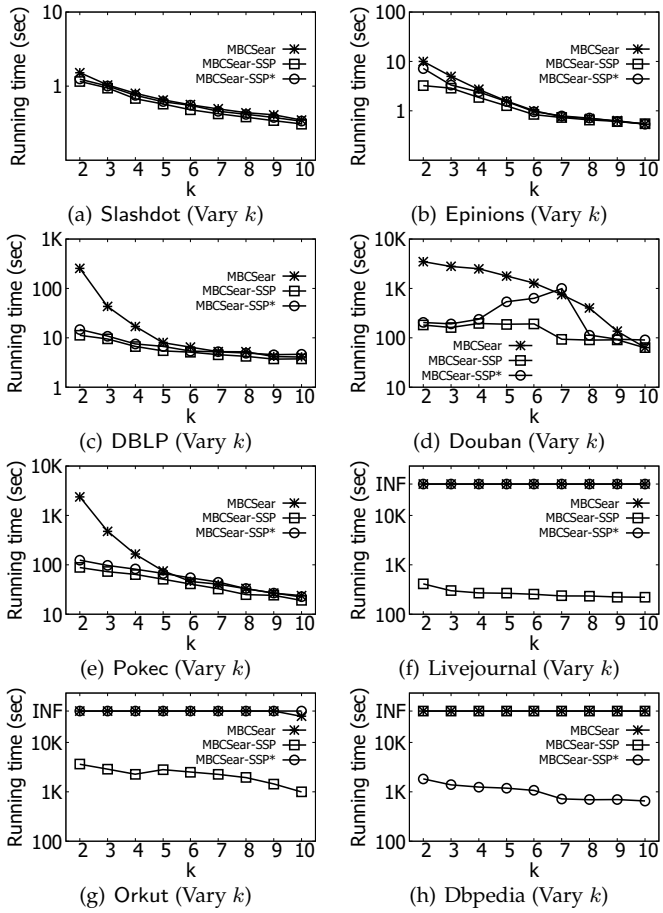



Fig. 9: Running time of MBCS algorithms varying $k$

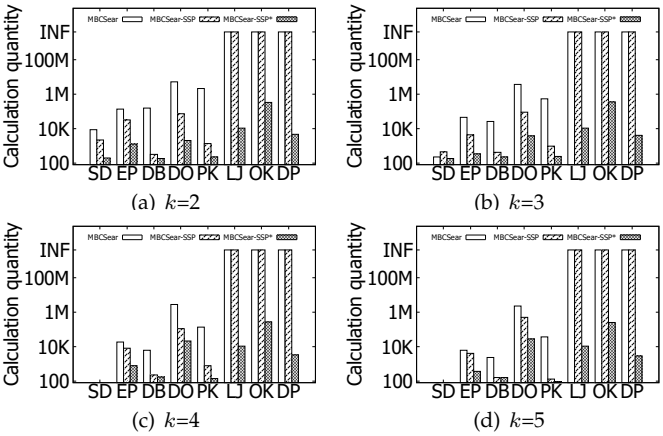



Fig. 10: Calculation quantity of different algorithms

graph $G'$ including positive edges number $|E^+_{G'}|$ and negative edges number $|E^-_{G'}|$, the ratio of $G'$ in the original graph $G$, and the maximum balanced clique size $\epsilon$ found so far. Since Douban and Pokec are big datasets, the search process is time consuming, hence, MBCSear-SSP$^*$ adopts $Dec(\overline{k}) = \lceil \frac{\overline{\kappa}}{2} \rceil$.

As Table 3 shows, on Douban, the first search region $(\underline{\kappa}_0, \overline{\kappa}_0)$ is $(2, 78)$, and $\epsilon$ is initialized as 4. MBCSear-SSP$^*$ first invokes EdgeReduction$^+$ to pre-reduce useless edges in $G$. Due to the large value of $\overline{\kappa}_0$, all edges are pruned from $G$. Hence, $G'$ is empty here. For the second search region $(2, 39)$, $G'$ is still empty. For the third search region $(2, 20)$,

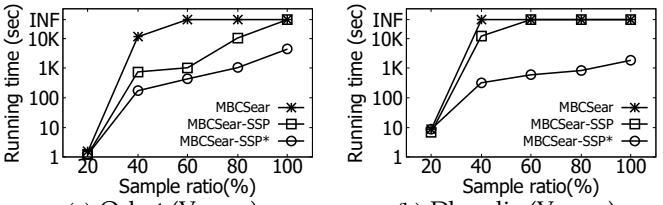
TABLE 3: The search process of MBCSear-SSP*

(a) MBCSear-SSP* on Douban, $k$=2

| Index | Search Region | $\lvert E^+_{G'}\rvert$ | $\lvert E^-_{G'}\rvert$ | $G'/G(\%)$ | $\epsilon$ |
|---|---|---|---|---|---|
| 0 | (2,78) | 0 | 0 | 0 | 4 |
| 1 | (2,39) | 0 | 0 | 0 | 4 |
| 2 | (2,20) | 21,695 | 12,190 | 0.24 | 33 |
| 3 | (13,13) | 0 | 0 | 0 | 33 |

(b) MBCSear-SSP* on Pokec, $k$=2

| Index | Search Region | $\lvert E^+_{G'}\rvert$ | $\lvert E^-_{G'}\rvert$ | $G'/G(\%)$ | $\epsilon$ |
|---|---|---|---|---|---|
| 0 | (2,37) | 0 | 0 | 0 | 4 |
| 1 | (2,19) | 6,605 | 2,521 | 0.30 | 29 |
| 2 | (10,10) | 0 | 0 | 0 | 29 |




(a) Orkut (Vary $n$) (b) Dbpedia (Vary $n$)

Fig. 11: Scalability of MBCS algorithms, $k$=2

$G'$ has 21,695 positive edges and 12,190 negative edges, it only holds 0.24% edges of the original graph which is much less than $G$. Then, MBCSear-SSP* finds the maximum balanced clique $C^*$ on $G'$. The size of $C^*$ is 33. For the last search region $(13, 13)$, $G'$ is empty, the search process is finished. The search process on Pokec is similar.

By observing the search process of MBCSear-SSP* on the two datasets, We find two significant phenomena. First, benefited from the search space partition paradigm, the number of search regions is limited. Second, the input graph $G'$ for each search region is much smaller than the original graph, because the edge reduction strategy can remove most of the invalid edges before the search starting.

**Exp-7: Scalability of** MBCS **algorithms.** In this experiment, we evaluate the scalability of MBCS algorithms on two biggest datasets Orkut and Dbpedia as Exp-2. The results are shown at Figure 11.

As Figure 11 shows, with the number of vertices increases, the running time of three algorithms increases as well. Among them, the growth rate of MBCSear-SSP* is the most stable. For instance, on Dbpedia with 40% vertices, MBCSear cannot get result within a reasonable time, the running time of MBCSear-SSP and MBCSear-SSP* are 11953.0s and 319.9s, respectively. On Dbpedia with more than 40% vertices, only MBCSear-SSP* can get result within a reasonable time. The trend of running time on Orkut is similar. Therefore, MBCSear-SSP* can scale to large-scale graphs.

## 9 Conclusions

In this paper, we study the maximal balanced clique enumeration problem in signed networks. We propose a new enumeration algorithm tailored for signed networks. Based on the new enumeration algorithm, we explore two optimization strategies to further improve the efficiency of the enumeration algorithm. Besides, we study the maximum balanced clique search problem, and propose a novel search space partition-based search framework. Moreover, we explore multiple optimization strategies to further reduce the search space during search process. The experimental results on real datasets demonstrate the efficiency, effectiveness and scalability of our solutions.

## References


[1] D. Ouyang, L. Yuan, F. Zhang, L. Qin, and X. Lin, "Towards efficient path skyline computation in bicriteria networks," in *Proceedings of DASFAA*, 2018, pp. 239–254.

[2] L. Yuan, L. Qin, W. Zhang, L. Chang, and J. Yang, "Index-based densest clique percolation community search in networks," *IEEE TKDE*, vol. 30, no. 5, pp. 922–935, 2018.

[3] L. Yuan, L. Qin, X. Lin, L. Chang, and W. Zhang, "Effective and efficient dynamic graph coloring," *PVLDB*, vol. 11, no. 3, pp. 338–351, 2017.

[4] X. Feng, L. Chang, X. Lin, L. Qin, W. Zhang, and L. Yuan, "Distributed computing connected components with linear communication cost," *Distributed and Parallel Databases*, vol. 36, no. 3, pp. 555–592, 2018.

[5] B. Liu, L. Yuan, X. Lin, L. Qin, W. Zhang, and J. Zhou, "Efficient ($\alpha$, $\beta$)-core computation: an index-based approach," in *Proceedings of WWW*, 2019, pp. 1130–1141.

[6] X. Wu, L. Yuan, X. Lin, S. Yang, and W. Zhang, "Towards efficient k-tripeak decomposition on large graphs," in *Proceedings of DASFAA*, 2019, pp. 604–621.

[7] Z. Qing, L. Yuan, F. Zhang, L. Qin, X. Lin, and W. Zhang, "External topological sorting in large graphs," in *Proceedings of DASFAA*, 2018, pp. 203–220.

[8] D. Ouyang, L. Yuan, L. Qin, L. Chang, and Y. Zhang, "Efficient shortest path index maintenance on dynamic road networks with theoretical guarantees," *PVLDB*, vol. 13, no. 5, pp. 602–615, 2020.

[9] J. Pattillo, N. Youssef, and S. Butenko, "On clique relaxation models in network analysis," *European Journal of Operational Research*, vol. 226, no. 1, pp. 9–18, 2013.

[10] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph (algorithm 457)," *Commun. ACM*, vol. 16, no. 9, pp. 575–576, 1973.

[11] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," in *International Symposium on Algorithms and Computation*, 2010, pp. 403–414.

[12] D. Eppstein and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," in *International Symposium on Experimental Algorithms*, 2011, pp. 364–375.

[13] L. Yuan, L. Qin, X. Lin, L. Chang, and W. Zhang, "Diversified top-k clique search," *VLDB J.*, vol. 25, no. 2, pp. 171–196, 2016.

[14] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[15] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in *IEEE 16th International Conference on Data Mining*, 2016, pp. 221–230.

[16] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: mining a social network with negative edges," in *Proceedings of WWW*, 2009, pp. 741–750.

[17] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proceedings of SIGCHI*, 2010, pp. 1361–1370.

[18] L. Ou-Yang, D.-Q. Dai, and X.-F. Zhang, "Detecting protein complexes from signed protein-protein interaction networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 6, pp. 1333–1344, 2015.

[19] D. Cartwright and F. Harary, "Structural balance: a generalization of heider's theory." *Psychological review*, vol. 63, no. 5, p. 277, 1956.

[20] S. A. Marvel, S. H. Strogatz, and J. M. Kleinberg, "Energy landscape of social balance," *Physical review letters*, vol. 103, no. 19, p. 198701, 2009.

[21] P. Abell and M. Ludwig, "Structural balance: a dynamic perspective," *Journal of Mathematical Sociology*, vol. 33, no. 2, pp. 129–155, 2009.

[22] S. A. Marvel, J. Kleinberg, R. D. Kleinberg, and S. H. Strogatz, "Continuous-time model of structural balance," *Proceedings of the National Academy of Sciences*, vol. 108, no. 5, pp. 1771–1776, 2011.

[23] T. Derr, C. Aggarwal, and J. Tang, "Signed network modeling based on structural balance theory," in *Proceedings of CIKM*, 2018, pp. 557–566.

[24] F. Heider, "Attitudes and cognitive organization," *The Journal of psychology*, vol. 21, no. 1, pp. 107–112, 1946.

[25] F. Harary *et al.*, "On the notion of balance of a signed graph." *The Michigan Mathematical Journal*, vol. 2, no. 2, pp. 143–146, 1953.

[26] L. Chang, "Efficient maximum clique computation and enumeration over large sparse graphs," *VLDB J.*, vol. 29, no. 5, pp. 999–1022, 2020.

[27] C. Lu, J. X. Yu, H. Wei, and Y. Zhang, "Finding the maximum clique in massive graphs," *PVLDB*, vol. 10, no. 11, pp. 1538–1549, 2017.

[28] J. Chen, S. Cai, S. Pan, Y. Wang, Q. Lin, M. Zhao, and M. Yin, "Nuqclq: An effective local search algorithm for maximum quasi-clique problem," in *AAAI 2021*, pp. 12 258–12 266.

[29] B. Lyu, L. Qin, X. Lin, Y. Zhang, Z. Qian, and J. Zhou, "Maximum biclique search at billion scale," *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1359–1372, 2020.

[30] A. Zhou, Y. Wang, and L. Chen, "Finding large diverse communities on networks: The edge maximum k*-partite clique," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2576–2589, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p2576-zhou.pdf

[31] E. Sevinç and T. Dökeroglu, "A novel parallel local search algorithm for the maximum vertex weight clique problem in large graphs," *Soft Comput.*, vol. 24, no. 5, pp. 3551–3567, 2020.

[32] R. Li, X. Wu, H. Liu, J. Wu, and M. Yin, "An efficient local search for the maximum edge weighted clique problem," *IEEE Access*, vol. 6, pp. 10 743–10 753, 2018.

[33] X. Song, Y. Chi, K. Hino, and B. Tseng, "Identifying opinion leaders in the blogosphere," in *Proceedings of CIKM*, 2007, pp. 971–974.

[34] R. Axelrod and D. S. Bennett, "A landscape theory of aggregation," *BJPS*, vol. 23, no. 02, p. 211–233, 1993.

[35] M. Išoraitė, "Importance of strategic alliances in company's activity," *BJPS*, vol. 1, no. 5, p. 39–46, 2009.

[36] G. A. Miller., "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, p. 39–41, 1995.

[37] V. Kumar, N. Joshi, A. Mukherjee, G. Ramakrishnan, and P. Jyothi, "Cross-lingual training for automatic question generation," in *ACL*, 2019, pp. 4863–4872.

[38] A. Krishnan, D. P, S. Ranu, and S. Mehta, "Leveraging semantic resources in diversified query expansion," *World Wide Web*, vol. 21, no. 4, pp. 1041–1067, 2018.

[39] X. Zheng and D. Zeng, "Social balance in signed networks," *Information Systems Frontiers*, vol. 17, no. 5, pp. 1077–1095, 2015.

[40] R.-H. Li, Q. Dai, L. Qin, G. Wang, X. Xiao, J. X. Yu, and S. Qiao, "Efficient signed clique search in signed networks," in *Proceedings of ICDE*, 2018, pp. 245–256.

[41] F. Hao, S. S. Yau, G. Min, and L. T. Yang, "Detecting k-balanced trusted cliques in signed social networks," *IEEE Internet Computing*, vol. 18, no. 2, pp. 24–31, 2014.

[42] Akkoyunlu and E. A, "The enumeration of maximal cliques of large graphs[j]," *SIAM Journal on Computing*, vol. 2, no. 1, pp. 1–6, 1973.

[43] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, no. 1, pp. 28–42, 2006.

[44] F. Fakhfakh, M. Tounsi, M. Mosbah, and A. H. Kacem, "Algorithms for finding maximal and maximum cliques: A survey," in *International Conference on Intelligent Systems Design and Applications*, 2017, pp. 745–754.

[45] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[46] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," *IEEE Transactions on Knowledge and Data Engineering*, 2008.

[47] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," in *SIGMOD*, 2014, pp. 1311–1322.

[48] R. Zhou, C. Liu, J. X. Yu, W. Liang, B. Chen, and J. Li, "Finding maximal k-edge-connected subgraphs from a large graph," *in EDBT*, 2012.

[49] L. Yuan, L. Qin, X. Lin, L. Chang, and W. Zhang, "I/O efficient ECC graph decomposition via graph reduction," *PVLDB*, vol. 9, no. 7, pp. 516–527, 2016.

[50] ——, "I/O efficient ECC graph decomposition via graph reduction," *VLDB J.*, vol. 26, no. 2, pp. 275–300, 2017.

[51] A. E. Sariyuce, C. Seshadhri, A. Pinar, and U. V. Catalyurek, "Finding the hierarchy of dense subgraphs using nucleus decompositions," in *Proceedings of WWW*, 2015, pp. 927–937.

[52] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek, "Nucleus decompositions for identifying hierarchy of dense subgraphs," *TWEB*, vol. 11, no. 3, pp. 16:1–16:27, 2017.

[53] J. Cheng, L. Zhu, Y. Ke, and S. Chu, "Fast algorithms for maximal clique enumeration with limited memory," in *Proceedings of SIGKDD*, 2012, pp. 1240–1248.

[54] M. C. Schmidt, N. F. Samatova, K. Thomas, and B.-H. Park, "A scalable, parallel algorithm for maximal clique enumeration," *J. Parallel Distrib. Comput.*, vol. 69, no. 4, pp. 417–428, 2009.

[55] Y. Zhang, C. A. Phillips, G. L. Rogers, E. J. Baker, E. J. Chesler, and M. A. Langston, "On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types," *BMC bioinformatics*, vol. 15, no. 1, p. 110, 2014.

[56] D. Wen, L. Qin, Y. Zhang, X. Lin, and J. X. Yu, "I/O efficient core graph decomposition: Application to degeneracy ordering," *IEEE TKDE*, vol. 31, no. 1, pp. 75–90, 2019.

[57] L. Lovász, M. E. Saks, and W. T. Trotter, "An on-line graph coloring algorithm with sublinear performance ratio," *Discret. Math.*, vol. 75, no. 1-3, pp. 319–325, 1989.

[58] L. Chu, Z. Wang, J. Pei, J. Wang, Z. Zhao, and E. Chen, "Finding gangs in war from signed networks," in *Proceedings of SIGKDD*, 2016, pp. 1505–1514.

[59] T. Xu, D. Liu, E. Chen, H. Cao, and J. Tian, "Towards annotating media contents through social diffusion analysis," in *Proceedings of ICDM*, 2012, pp. 1158–1163.