Computing Significant Cliques in Large Labeled Networks

Yu-Xuan Qiu, Dong Wen, Rong-Hua Li, Lu Qin, Michael Yu, and Xuemin Lin, *Fellow, IEEE*

Abstract—Mining cohesive subgraphs and communities is a fundamental problem in network analysis and has drawn much attention in the last decade. Most existing cohesive subgraph models mainly consider the structural cohesion but ignore the subgraph significance. In this article, we formulate a new model, called statistically significant clique, to mine significant cohesive subgraphs in large vertex-labeled graphs. A statistically significant clique is a complete subgraph with a significance value exceeding a given threshold. The subgraph significance is evaluated by a widely used metric called chi-square statistic. We study the problem of enumerating all maximal statistically significant cliques. The problem is proved to be NP-hard. We propose an efficient branch-and-bound algorithm with several elegant pruning strategies to solve our problem. We conduct extensive experiments on seven large real-world datasets to show the practical efficiency of our algorithms. We also conduct a case study to evaluate the effectiveness of our proposed model.

Index Terms—Clique, statistical significance, labeled graph, cohesive subgraph, big graph processing

1 Introduction

Graph models have been used to capture the relationships among entities in a wide spectrum of applications, such as social networks [1], [2], biological networks [3], [4], and collaboration networks [5], [6]. A proliferation in graph-based applications has steered research efforts toward the challenges in managing and analyzing graphs. Discovering cohesive subgraphs is a fundamental problem with numerous applications like detecting social communities [7], [8] and mining protein complexes [9]. This paper aims to mine statistically significant cohesive subgraphs, which have never been considered in previous studies. Generally, the task identifies a set of densely connected subgraphs with certain properties beyond the standard distribution.

Many efforts have been made on extracting significant substructures [10], [11], [12], [13] among the studies for mining subgraph patterns. Several works study the problem of frequent subgraph mining [12], [13], where a subgraph is considered to be significant if its frequency exceeds a predefined threshold. However, the subgraph frequency only considers the structural property, while vertices in a tremendous amount of real-world graphs are always associated with a set of labels or attributes. For example, in a protein-protein-interaction network, each vertex represents a protein, and the labels may represent protein functionalities. To capture the label statistics in significant subgraph mining, Arora et al. studied the problem of computing *statistically significant connected subgraphs* [14].

In statistics, the significance provides the evidence concerning the plausibility of the *null hypothesis*, which hypothesizes that the data distribution is only affected by random chance. If we have evidence to reject the null hypothesis, the corresponding result is considered statistically significant. In the statistically significant connected subgraph model, the null hypothesis is that the labels on each vertex are assumed to be assigned independently and randomly from a fixed probability distribution. The deviation between the actual labels and the expected labels measures the statistical significance and is computed via a function called *chi-square statistic* [15], which has also been used in many other works [14], [16], [17], [18]. The higher the chi-square, the higher the statistical significance [12], [14]. In this paper, we also evaluate the significance by the chi-square statistic. Given a set of vertices U and their labels, the chi-square statistic is formally defined as follows.

$$f(U) = \sum_{i=1}^{l} \frac{(y_i - yp_i)^2}{yp_i},$$

where l is the number of distinct labels, y is the total number of all labels in U, y_i is the number of the ith labels, and p_i is the expected frequency of the ith label. A higher chi-square statistic means a higher deviation between the real and

- *Yu-Xuan Qiu and Lu Qin are with the AAII, University of Technology Sydney, Ultimo, NSW 2007, Australia. E-mail: qiuyx.cs@gmail.com, lu.qin@uts.edu.au.*
- *Dong Wen and Michael Yu are with the School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia. E-mail: dong.wen@unsw.edu.au, mryu@cse.unsw.edu.au.*
- *Rong-Hua Li is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. E-mail: lironghuabit@126.com.*
- *Xuemin Lin is with the Antai College of Economics & Management, Shanghai Jiaotong University, Shanghai 200240, China. E-mail: xuemin.lin@sjtu.edu.cn.*

Manuscript received 4 December 2021; revised 25 September 2022; accepted 8 November 2022. Date of publication 21 November 2022; date of current version 13 May 2023.
The work of Rong-Hua Li was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3301301, in part by NSFC under Grants U2241211, 62072034, and U1809206, and in part by CCF-Huawei Populus Grove Fund. The work of Lu Qin was supported in part by ARC under Grants FT200100787 and DP210101347.
(Corresponding author: Xuemin Lin.)
Recommended for acceptance by P. D?Urso.
Digital Object Identifier no. 10.1109/TBDATA.2022.3223644

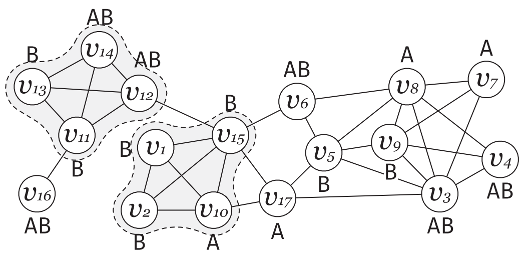


Fig. 1. A labeled graph G and maximal (k, θ)-significant cliques in G given $k = 4, \theta = 6.0, p_A = 0.8$, and $p_B = 0.2$.

expected label distributions which may indicate some intrinsic properties. For example, the numbers of male and female staff are expected to be similar in a company. Given the real numbers of them, a high chi-square statistic may indicate some gender inequality in the company.

Clique is a fundamental and commonly used model for cohesive subgraph detection [19]. An induced subgraph S is a clique if there exists an edge between every pair of vertices in S. The clique model has drawn much research attention, such as enumerating maximal cliques [20], [21], [22], computing the maximum clique [23], [24], and mining clique-based subgraphs, e.g., signed cliques [25], [26], [27].

Our Model. Based on the concepts of chi-square statistic and clique, we propose a novel significant cohesive subgraph model, which is called (k, θ)-significant cliques, in vertex-labeled graphs. Given a size constraint k, a significance threshold θ, and a probability distribution P as the input, a (k, θ)-clique S satisfies the following three properties: (i) S is a clique in which every pair of vertices is connected; (ii) the chi-square statistic of S is at least equal to θ; and (iii) the number of vertices in S is no smaller than k. The first two properties support us to find significant cohesive subgraphs in the graph, and the third property helps us avoid some small graphlets like edges and triangles. We study the problem of enumerating all maximal (k, θ)-significant cliques in a graph. Given an integer $k = 4$, a real value $\theta = 6$ and a label probability distribution $\{p_{\mathsf{A}} = 0.8, p_{\mathsf{B}} = 0.2\}$, Fig. 1 shows an example of all maximal (k, θ)-significant cliques.

Applications. The problem of enumerating all maximal (k, θ)-significant cliques has many applications, including but not limited to discovering influential research groups in collaboration networks [28], detecting topic-centric communities in social networks [29], and revealing important functional organizations in PPI networks [30].

Research Group Discovery in Co-Author Networks. In a co-author network (e.g., DBLP), two researchers are connected by an edge if they have a common publication. A researcher may have several labels or attributes, and each label represents a conference or a journal where the researcher has a paper published. Setting a relatively low expected frequency for some high-ranking conferences or journals in a research domain leads to a higher chi-square statistic for groups with more such publications. Our model can be applied to identify outstanding research groups with many high-quality publications. We have conducted a case study on DBLP to discover such research groups in Section 6.

Topic-Centric Community Detection in Social Networks. In social networks, each user may have several labels representing the followed topics, like soccer and basketball. The model can help mine topic-centric communities that have a strong association with some specific topics (or features) far beyond others. For example, in sports marketing, it is crucial to locate the avid sports fans. Assume that we would like to mine a set of pure "soccer" communities for recommendations and advertisements. A straightforward method is to collect all the vertices following "soccer" and compute cliques in the induced subgraph. However, this method does not consider other labels, and the resulting communities may also be highly interested in other sports. By setting a suitable parameter, our model can find a set of communities whose members care about "soccer" far beyond other sports.

Organization Mining in Biological Networks. In PPI networks, each protein is assigned several labels by its functionalities. By setting specific parameters, our model can be used to identify a set of biological organizations (closely connected proteins) with some particular statistics far beyond normal. The derived structures may play crucial roles in certain biological processes.

Note that there have been several keyword-based community models in labeled networks. However, they cannot easily cover our research problem and techniques. First, existing works either cannot guarantee strong structural cohesion [31], [32] or focus on other cohesion models like k-core and k-truss [33]. To the best of our knowledge, we are the first to study significant clique mining in labeled graphs given the prevalence of the fundamental clique model. Our model guarantees both the strongest structural cohesion and flexible keyword significance. Second, several works only accept an input graph and cannot support personalized keyword (distribution) queries [31], [33], [34], [35], [36]. For example, Chu et al. [33] find cohesive subgraphs where the common pattern is frequent in all vertices. The common pattern is generated by the algorithm. Third, several keyword-based community detection models accept a set of keywords as the input and only consider the existence of keywords [37], [38], [39]. Such models compute cohesive subgraphs where each vertex covers as many given keywords as possible. In these models, the importance of all input keywords is always the same. By contrast, the statistical significance model provides an input of keyword distribution. Thus, we can flexibly assign different strengths to input keywords according to specific scenarios.

Challenges. It is challenging to compute all maximal (k, θ)-significant cliques. First, the problem is NP-hard, which is proved in Section 2.2 by showing that the maximal clique enumeration is a special case of our problem. Second, the significance constraint in the model is not anti-monotonic. In other words, even though we find a clique S with a chi-square statistic less than θ, a sub-clique of S may still have a chi-square statistic larger than θ. As a result, we still need to check every possible sub-clique of S further. Therefore, the technical challenges include how to correctly output maximal (k, θ)-significant cliques without any duplication and how to prune the search space effectively.

Our Solution. Based on the concept of k-core [40] and graph coloring [41] in existing studies, we propose a basic branch-and-bound algorithm. However, the integer k is normally small to only filter out some small motifs, which diminishes the pruning effectiveness of basic structural rules. To improve the practical efficiency, we observe an upper bound for the chi-square statistic of a given vertex set (Theorem 2). Based on the upper bound, we combine the concepts of k-core and graph coloring and derive a vertex reduction strategy with stronger structural pruning effectiveness and a statistical pruning rule. We further extend the ideas from vertices to edges and propose an edge pruning strategy.

Contributions. We summarize the main contributions in this paper as follows.

- *An elegant significant cohesive subgraph model.* We propose a novel subgraph model, called (k, θ)-significant clique in labeled graphs. We prove that the problem of computing all maximal (k, θ)-significant cliques is NP-hard.
- *An algorithm for significant clique enumeration.* We propose a novel branch-and-bound algorithm to enumerate maximal (k, θ)-significant cliques without any duplication.
- *Several strategies to prune the search space.* We propose two effective pruning strategies from the perspectives of both vertex reduction and edge reduction, which take $O(m \cdot \log deg_{max})$ and $O(m^{1.5})$ times, respectively. m is the number of edges, and deg_{max} is the maximum degree.
- *Extensive performance studies.* We conduct extensive experiments on seven real-world datasets to evaluate the efficiency of our proposed algorithms. We also conduct a case study to show the effectiveness of our model.

Organization. The rest of this paper is organized as follows. Section 2 introduces background knowledges and defines the problem. Section 3 proposes a non-trivial baseline algorithm. Section 4 gives several pruning strategies. Section 5 presents the final algorithm. Section 6 reports the performance studies. Section 7 introduces related works, and Section 8 concludes the paper. We omit the proofs for several straightforward lemmas and theorems, which can be obtained in the full version [42].

2 Preliminary

2.1 Problem Definition

We consider an undirected labeled graph $G(V, E, \mathcal{L})$. V is the set of vertices. $E \subseteq (V \times V)$ is the set of edges. $\mathcal{L}$ assigns one or more labels to each vertex from a label set L, i.e., $\mathcal{L} : V \to \bigcup_{v \in V, L_v \subseteq L} L_v$. We use n and m to represent $|V|$ and $|E|$, respectively. Given a vertex u, the neighbor set of u is denoted by $N(u)$, i.e., $N(u) = \{v \in V | (u, v) \in E\}$. The degree of u is denoted by $deg(u)$, i.e., $deg(u) = |N(u)|$. A subgraph $S(V_S, E_S)$ is called an induced subgraph of G if $V_S \subseteq V$ and $E_S = \{(u, v) \in E | u \in V_S, v \in V_S\}$. A subgraph S of G is a clique if every two vertices in S are connected, i.e., $\forall u, v \in V_S, (u, v) \in E_S$. The clique S is called a k-clique if there are k vertices in S, i.e., $|V_S| = k$.

Given a set of vertices $U \in V$, assume that l is the number of distinct labels in U, i.e., $l = |\bigcup_{u \in U} \mathcal{L}(u)|$. We have an observed frequency vector $Y = \{y_1, y_2, \ldots, y_l\}$, where $y = \sum_{i=1}^{l} y_i = \sum_{u \in U} |\mathcal{L}(u)|$. Given a fixed label probability distribution $P = \{p_1, p_2, \ldots, p_l\}$, the *chi-square* statistic [15] (also called statistical significance [14]) of U is defined as follows.

$$f(U) = \sum_{i=1}^{l} \frac{(y_i - yp_i)^2}{yp_i} = \sum_{i=1}^{l} \frac{y_i^2}{yp_i} - y. \tag{1}$$

Example 1. Given the graph G in Fig. 1, we consider the induced subgraph of $\{v_5, v_6, v_8\}$. We have two A labels and two B labels. We have $l = 2$. The observed frequency vector is $Y = \{y_\mathsf{A} = 2, y_\mathsf{B} = 2\}$, and $y = 4$. Assume that the probability distribution of the labels is $P = \{p_\mathsf{A} = 0.8, p_\mathsf{B} = 0.2\}$. The chi-square of $\{v_5, v_6, v_8\}$ is $\frac{4}{4 \times 0.8} + \frac{4}{4 \times 0.2} - 4 = 2.25$.

The chi-square statistic of a subgraph represents the deviation of the observed label frequency from the expected frequency distribution, which is a widely used metric to quantify the statistical significance [15], [16], [17], [18]. Arora et al. [14] show that the subgraph with a large chi-square statistic is considered to be highly significant. Based on Eq. (1), we define a new subgraph model, called (k, θ)*-significant clique*, as follows.

Definition 1. *(Significant Clique) Given a graph G, a probability distribution $P = \{p_1, p_2, \ldots, p_l\}$, an integer k and a real value θ, a (k, θ)-significant clique is an induced subgraph C that satisfies the following constraints:*

- *Clique constraint: C is a clique;*
- *Chi-square constraint: $f(V_C) \geq \theta$;*
- *Size constraint: $|V_C| \geq k$.*

In Definition 1, the clique constraint ensures the subgraph is densely connected and can be a cohesive pattern or a social community. The chi-square constraint ensures the subgraph is highly significant in the given graph. The size constraint filters out small resulting motifs in the (k, θ)-significant cliques. The probability distribution P enables flexibility to adjust the importance of the labels.

Definition 2. *(Maximal Significant Clique) A subgraph C of G is a maximal (k, θ)-significant clique if (i) C is a (k, θ)-significant clique, and (ii) there is no (k, θ)-significant clique C' in G who contains clique C.*

A (k, θ)-significant clique may contain several subgraphs which are still (k, θ)-significant cliques. The maximality of the model reduces the redundancy in resulting subgraphs.

Example 2. Fig. 1 shows an example of the maximal significant cliques. Given $k = 4, \theta = 6.0, p_\mathsf{A} = 0.8$ and $p_\mathsf{B} = 0.2$, all maximal (4,6)-significant cliques in G are marked by gray. Note that if $k = 3$, we have $f(v_{11}, v_{12}, v_{13}) = 7.563$. As a result, the induced subgraph of $\{v_{11}, v_{12}, v_{13}\}$ is a (3,6)-significant clique but not maximal.

We use (k, θ)-clique to represent the maximal (k, θ)-significant clique for short when context is clear. Based on Definition 2, we define the research problem as follows.

Problem Statement. Given a labeled graph G, a probability distribution P, an integer k and a real value θ, we aim to enumerate all maximal (k, θ)-significant cliques in G.

2.2 Hardness and Challenges

2.2.1 NP-Hard Time Complexity

We prove the hardness of our problem by considering a closely related problem — maximal clique enumeration, which has been widely studied in the literature [43], [44], [45], [46]. All maximal cliques are the results of a special case of our problem. Specifically, given $k = 0$ and $\theta = 0$, the chi-square statistic of an arbitrary vertex set is always no less than θ, and the problem of enumerating (0,0)-significant cliques is equivalent to the problem of maximal clique enumeration. Given that the maximal clique enumeration problem is NP-hard, our problem is also NP-hard.

2.2.2 Non-Monotonicity

Given a set S, an anti-monotonic constraint means that if S satisfies (or does not satisfy) the constraint, any subset of S also satisfies (or does not satisfy) the constraint. For example, the clique constraint in Definition 1 is anti-monotonic since any subgraph of a clique is also a clique. The size constraint in Definition 1 is anti-monotonic since if a graph S has fewer than k vertices, any subgraph of S also has fewer than k vertices. However, the chi-square constraint in Definition 1 is not anti-monotonic. In other words, given a graph S with $f(V_S) \geq \theta$ and an arbitrary subgraph S' of S, we cannot derive $f(V_{S'}) \geq \theta$ and vise versa.

Example 3. Given the graph G in Fig. 1, assume that $k = 4, \theta = 6.0, p_{\mathsf{A}} = 0.8$ and $p_{\mathsf{B}} = 0.2$. Considering the induced triangle of $\{v_{12}, v_{13}, v_{14}\}$, we have the chi-square value $f(v_{12}, v_{13}, v_{14}) = 5$, which is less than the expected threshold θ. However, we cannot remove the vertices since an induced supergraph of $\{v_{11}, v_{12}, v_{13}, v_{14}\}$ has a chi-square value 8.167. On the other hand, we consider the vertex set $\{v_1, v_2, v_{10}, v_{15}, v_{17}\}$, whose chi-square is 5 and less than θ. However, we still cannot remove the vertices since a subset $\{v_1, v_2, v_{10}, v_{15}\}$ has a chi-square value 7.563, which is larger than θ.

Without the anti-monotonicity, we cannot immediately borrow the idea of existing algorithms for maximal clique enumeration. Specifically, once finding a maximal clique C, even $f(V_C) < \theta$, it is possible that a sub-clique C' of C satisfies $f(V_{C'}) \geq \theta$. Consequently, we cannot filter out C and need to further check every possible sub-clique of C with no fewer than k vertices. The method works but produces numerous intermediate results since a (k, θ)-significant clique may be involved in several maximal cliques. In addition, the number of cliques can be extremely large (up to $3^{n/3}$ in the worst case [45]), which makes the naive solution costly in big graphs. Therefore, the main challenges are how to avoid outputting the duplicated results and how to prune the search space effectively.

3 A Branch-and-Bound Algorithm

3.1 Basic Structural Graph Reduction

To handle the challenges discussed in Section 2.2, we give a non-trivial baseline algorithm in this section. We start by introducing several basic pruning rules, which can be easily derived from existing clique studies, like k-clique enumeration [47] and the maximum clique computation [24].

Core Based Pruning. The first structural pruning rule is based on k-core, which is formally defined as follows.

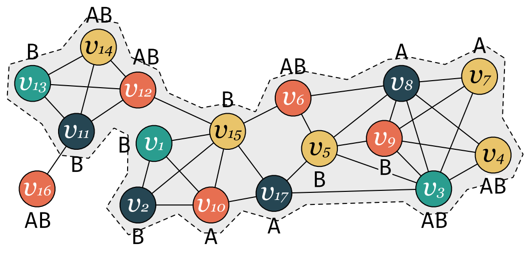


Fig. 2. A coloring and the 3-core of the graph G.

Definition 3. *(k-Core) Given a graph G and an integer k, a k-core in G is a maximal connected subgraph in which the degree of every vertex is at least k [40].*

Lemma 1. *Given a graph G and a vertex u, u is contained in a k-clique only if it is contained in a $(k-1)$-core [48].*

Based on Lemma 1, all vertices not belonging to the $(k-1)$-core can be safely removed before (k, θ)-clique computation. Given an integer k, we can compute the $(k-1)$-core by iteratively removing all vertices with degree less than $k - 1$. The running time is bounded by $O(m)$ [49]. An example of the 3-core in the graph G of Fig. 1 is marked by gray in Fig. 2.

Graph Coloring-Based Pruning. The second structural pruning rule is based on the concept of graph coloring. Given a graph $G(V, E)$, a coloring of G is an assignment of a *color number*, denoted by $\mathsf{color}(u)$, for each vertex u such that two adjacent vertices share different colors, i.e., $\forall (u, v) \in E, \mathsf{color}(u) \neq \mathsf{color}(v)$. Given a colored graph G and a subgraph S of G, we use $\mathsf{colors}(S)$ or $\mathsf{colors}(V_S)$ to denote all distinct color numbers in S, i.e., $\mathsf{colors}(S) = \{\mathcal{C} | \exists u \in V_S, \mathsf{color}(u) = \mathcal{C}\}$.

Lemma 2. *A graph G contains a k-clique only if there are at least k distinct colors in G, i.e., $|\mathsf{colors}(G)| \geq k$ [41].*

Based on Lemma 2, we avoid enumerating (k, θ)-significant cliques of a subgraph S if the number of distinct colors in V_S is less than k, i.e., $|\mathsf{colors}(S)| < k$. The pruning effectiveness closely depends on the coloring result. The fewer distinct color numbers, the more subgraphs can be pruned. However, it is NP-hard to color a graph with the minimum distinct color numbers [50]. Several heuristic methods have been proposed for coloring graphs in practice, and a widely used one is the greedy method following the graph degeneracy order [51]. Specifically, a vertex permutation $\{v_1, v_2, \ldots, v_n\}$ is a *degeneracy order* if every vertex v_i has the smallest degree in the induced subgraph of $\{v_i, v_{i+1}, \ldots, v_n\}$. Computing the degeneracy order takes $O(m)$ time. The coloring algorithm processes vertices in the reverse degeneracy order and greedily assigns each vertex the smallest color number that is not the same as that of any colored neighbor. The degeneracy-order-based coloring can be conducted in $O(m)$ time. A graph coloring for the graph G in Fig. 1 is provided in Fig. 2.

3.2 Computing Maximal Significant Cliques

The Key Idea. We propose a branch-and-bound algorithm, called $\mathsf{SigClique}$, to enumerate all (k, θ)-significant cliques. Without loss of generality, we assume that the input graph is connected. Given a set of vertices R initialized as V, we aim to compute all (k, θ)-significant cliques in the induced subgraph of R. We first identify whether $G[R]$ is a

(k,θ)-significant clique. If $G[R]$ is not a valid (k,θ)-significant clique, SigClique randomly picks a vertex u and divides the search space into two subspaces: (i) the subspace of all (k,θ)-cliques containing u, and (ii) the subspace of all (k,θ)-cliques excluding u. Then SigClique recursively performs the same strategy for each subspace. In each recursion, we use I to denote the set of vertices that must be included in the (k,θ)-cliques in R. I is initialized as $\emptyset$. Consequently, R consists of I and a set of candidate vertices, which can be potentially included in the (k,θ)-clique. In each recursion, we pick a vertex from the set $R\setminus I$. We immediately terminate the search if $R=I$ since R is not a (k,θ)-clique and no subspace can be further explored.

If R is a valid (k,θ)-significant clique, we check the maximality of R by adding all possible common neighbors of R. No matter whether R is maximal or not, we terminate the current search space since all following (k,θ)-significant cliques are subsets of R and can never be maximal.

Algorithm 1. SigClique$(G(V,E),\theta,k)$

```
1: color $G$ based on the degeneracy order;
2: Enum$(V,\emptyset,\theta,k)$;
1: **Procedure** Enum$(R,I,\theta,k)$ :
2:    $R \leftarrow$ all vertices in $(k-1)$-core of $G[R]$;
3:    **if** $R\cap I \neq I$ **then return**;
4:    **if** $|\text{colors}(R)| < k$ **then return**;
5:    **if** $R$ *is a* $(k,\theta)$*-significant clique* **then**
6:       **if** IsMax$(R,\bigcap_{v\in R} N(v),\theta)$ **then** output $R$;
         // early termination
         **return**;
7:    **if** $R=I$ **then return**;
8:    pick a vertex $u$ from $R\setminus I$;
9:    Enum$(I\cup N_R(u)\cup\{u\}, I\cup\{u\},\theta,k)$;
10:   Enum$(R\setminus\{u\}, I,\theta,k)$;
1: **Procedure** IsMax$(R,C,\theta)$ :
2:    **if** $C=\emptyset$ **then return** $true$;
3:    pick a vertex $u$ from $C$;
4:    **if** $f(R\cup\{u\})\geq\theta$ **then return** $false$;
5:    **if** !IsMax$(R\cup\{u\}, C\cap N(u),\theta)$ **then**
6:       **return** $false$;
7:    **if** !IsMax$(R, C\setminus\{u\},\theta)$ **then return** $false$;
8:    **return** $true$;
```

The Algorithm. The pseudocode of SigClique is shown in Algorithm 1. In addition to R and I, we input k and θ to the procedure Enum. In Line 2 of Enum, we reduce the graph by computing $(k-1)$-core based on Lemma 1. Since all vertices in I must be contained in the (k,θ)-clique, we terminate the current search space if a vertex in I is removed in the $(k-1)$-core computation in Line 3. Based on Lemma 2, we count the number of distinct colors in R and terminate the search if R cannot be a k-clique. Line 5 identifies whether R is a (k,θ)-significant clique by checking the degree of each vertex and the chi-square statistic of R. Line 6 checks the maximality of R by invoking IsMax. Line 9 searches the subspace including u, where $N_R(u)$ represents u's neighbors in R. Line 10 searches the subspace excluding u.

In the procedure IsMax, R is the set of vertices to be checked, and C is all common neighbors of vertices in R. In Line 2, $C=\emptyset$ means no candidate vertex can be added to R, and R must be maximal. The maximality search is also divided into two subspaces. Line 5 identifies whether $R\cup\{u\}$ is maximal. Line 7 checks whether R is maximal when excluding u from the candidate set.

Theorem 1. *Algorithm 1 correctly computes all maximal (k,θ)-significant cliques in the graph G.*

4 Statistical Graph Reduction

Even though Algorithm 1 successfully computes (k,θ)-cliques without any redundancy, the pruning effectiveness is still limited, especially in large graphs and given a small size constraint. In this section, we study several pruning strategies regarding the chi-square statistic. Section 4.1 formulates a new cohesive subgraph model called (k,θ)-significant core. Section 4.2 embeds the concept of graph coloring to the (k,θ)-significant core, which improves the effectiveness of both structural pruning and statistical pruning. Section 4.3 extends the idea of (k,θ)-significant core to prune edges.

4.1 Pruning via Significant Core

To support the statistical pruning over the graph, we first give a key theorem as follows.

Theorem 2. *Given a set of labeled vertices V, a probability distribution P, two arbitrary subsets V_1 and V_2 with $V_1\cup V_2=V$ and $V_1\cap V_2=\emptyset$, we have $f(V_1)+f(V_2)\geq f(V)$.*

Proof. We prove the theorem by showing $f=f(V_1)+f(V_2)-f(V)\geq 0$. We expand f as follows based on Eq. (1).

$$f=\sum_{i=1}^{l}\frac{y_{1i}^2}{y_1p_i}-y_1+\sum_{i=1}^{l}\frac{y_{2i}^2}{y_2p_i}-y_2-\sum_{i=1}^{l}\frac{y_i^2}{yp_i}+y.$$

Given that $y_1+y_2=y$ and $y_{1i}+y_{2i}=y_i$, we transform the formula as follows.

$$\begin{aligned}f&=\sum_{i=1}^{l}\frac{1}{y_1y_2yp_i}\cdot\left(y_{1i}^2y_2^2+y_{2i}^2y_1^2-2y_{1i}y_{2i}y_1y_2\right)\\&=\sum_{i=1}^{l}\frac{(y_{1i}y_2-y_{2i}y_1)^2}{y_1y_2yp_i}\geq 0.\end{aligned}$$

We have $f\geq 0$, and the proof is completed. □

Based on Theorem 2, we formulate a new cohesive subgraph model, called (k,θ)-significant core, to prune unpromising vertices in the problem of (k,θ)-clique enumeration. Related definitions are given as follows.

Definition 4. (Neighborhood Significance) *Given a vertex u, the neighborhood significance of u, denoted by $f_n(u)$, is the sum of the significance of u and all its neighbors, i.e., $f_n(u)=f(u)+\sum_{v\in N(u)}f(v)$.*[1]

Definition 5. (Significant Core) *Given a graph G, an integer k and a positive real value θ, (k,θ)-Significant Core (SC for short) is a maximal connected subgraph of G in which every vertex u satisfies (i) $deg(u)\geq k$, and (ii) $f_n(u)\geq\theta$.*

1. Here, we use $f(u)+\sum_{v\in N(u)}f(v)$ as $f(u\cup N(u))$ requires much more effort to compute, although the later is tighter.

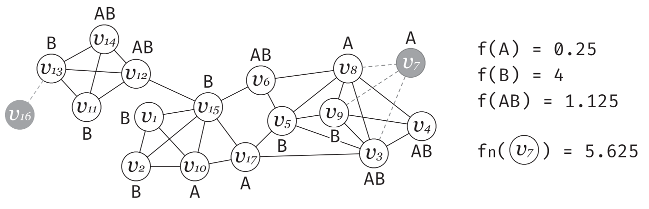


Fig. 3. Pruning G via (3,6)-significant core.

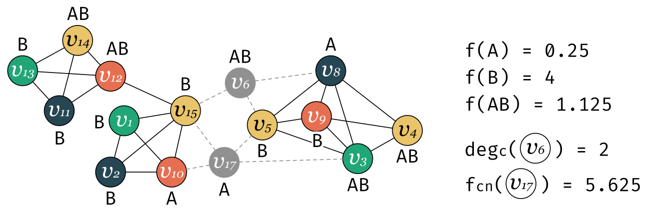


Fig. 4. Pruning G via colorful (3,6)-significant core.

Based on Definition 5, we can prune vertices supported by the following lemma.

Lemma 3. *A maximal (k,θ)-significant clique must be contained in a $(k-1,\theta)$-significant core.*

Example 4. Fig. 3gives an example of the pruning result via the (3,6)-significant core. The chi-square statistics of several required label sets are given on the right of Fig. 3. After computing the (3,6)-significant core, the vertices v_7 and v_{16} are removed. Specifically, v_{16} is removed since $deg(v_{16}) < 3$. For the vertex v_7, we have $f_n(v_7) = f(v_7) + f(v_3) + f(v_8) + f(v_9) = 5.625 < 6$. All the remaining vertices have degrees no less than 3 and neighborhood significance no less than 6. For example, the neighborhood significance of the vertex v_4 is 6.5.

Given a graph G, we can compute all (k,θ)-significant cores by a method similar to k-core computation. We recursively remove a vertex with degree less than k or neighborhood significance less than θ. If a neighbor v of u is removed, we update $f_n(u)$ to $f_n(u) - f(v)$. The time complexity is analyzed as follows.

Theorem 3. *Computing all (k,θ)-cores takes $O(m)$ time.*

4.2 Pruning via Colorful Significant Core

In this subsection, we further improve the pruning effectiveness by embedding the concept of graph coloring in (k,θ)-significant core. Compared with the significant core model, we strictly prune more vertices in terms of both graph structure and label statistics. From the structural perspective, we can combine the degree-based bound (Lemma 1) and the coloring-based bound (Lemma 2) as follows.

Definition 6. *(Colorful Degree) The colorful degree of a vertex u, denoted by $deg_c(u)$, is the number of distinct colors in $N(u)$, i.e., $deg_c(u) = |\mathsf{colors}(N(u))|$.*

Lemma 4. *For any (k,θ)-significant clique C, we have $deg_c(u) \geq k-1$ for every $u \in V_C$.*

Proof. We prove Lemma 4 by contradiction. We assume that there exists a vertex $u \in V_C$ whose degree is less than $k-1$. Then, we have $|\mathsf{colors}(N(u))| < k-1$. According to Lemma 2, $N(u)$ cannot contain a $(k-1)$-clique, thus, $\{u\} \cup N(u)$ cannot contain a k-clique. However, as C is a (k,θ)-significant clique, for every vertex $u \in V_C$, we have $\{u\} \cup N(u)$ contains a k-clique. That is a contradiction. □

Example 5. We give an example in the colored graph G of Fig. 2. Considering the vertex v_6, the degree of v_6 is 3. However, the colorful degree of v_6 is only 2, since there are only two distinct colors in the neighborhood of v_6. According to Lemma 4, v_6 cannot be in any $(4,\theta)$-significant clique.

From the statistical perspective, we combine the concepts of neighborhood significance (Definition 4) and the coloring-based bound (Lemma 2) as follows.

Definition 7. *(Colorful Neighborhood Significance) The colorful neighborhood significance of a vertex u, denoted by $f_{cn}(u)$, is the sum of significance of u and the maximum vertex significance for each color in $N(u)$, i.e.,*

$$f_{cn}(u) = f(u) + \sum_{\mathcal{C} \in \mathsf{colors}(N(u))} \max_{v \in N(u) | \mathsf{color}(v) = \mathcal{C}} f(v).$$

Lemma 5. *Given a (k,θ)-significant clique C, we have $f_{cn}(u) \geq \theta$ for every $u \in V_C$.*

Example 6. We give an example to explain Definition 7. Considering the vertex v_{17} in Fig. 2, we have three distinct colors in $N(v_{17})$. For the yellow color, we have two vertices v_5 and v_{15} in $N(v_{17})$. The labels of them are the same, and the largest statistic for the yellow color is $f(\mathsf{B}) = 4$. Each other color in $N(v_{17})$ has only one vertex. As a result, we have $f_{cn}(v_{17}) = f(v_{17}) + f(v_5) + f(v_{10}) + f(v_3) = 5.625$. By contrast, the neighborhood significance of v_{17} is $f_n(v_{17}) = 9.625$.

Based on Definitions 6 and 7, we give an extended version of (k,θ)-significant core.

Definition 8. *(Colorful Significant Core) Given a colored graph G, an integer k and a positive real value θ, a Colorful (k,θ)-Significant Core (CSC for short) is a maximal connected subgraph of G in which every vertex u satisfies (i) $deg_c(u) \geq k$, and (ii) $f_{cn}(u) \geq \theta$.*

Example 7. We give an example of the (3,6)-*CSC* in Fig. 4. Two vertices v_6 and v_{17} are removed from the graph. Based on Lemmas 4 and 5, they can never be in any (4,6)-clique.

The pruning effectiveness of (k,θ)-*CSC* is guaranteed to be stronger than that of (k,θ)-*SC* as shown below.

Theorem 4. *A colorful (k,θ)-significant core must be contained in a (k,θ)-significant core.*

Proof. Given a colorful (k,θ)-significant core C, for each vertex $u \in V_C$, we have $deg_c(u) \geq k$ and $f_{cn}(u) \geq \theta$. Obviously, $deg(u) \geq deg_c(u) \geq k$, and $f(u) \geq f_{cn}(u) \geq \theta$, thus, C is contained in a (k,θ)-significant core. □

*Colorful SC Computation.*Algorithm 2 presents the pseudocode for colorful (k,θ)-significant core computation. The strategy is similar to computing k-cores. Line 3 initializes the colorful degree and the colorful neighborhood significance for each vertex u. Line 4 adds u to the queue if it cannot be in the (k,θ)-CSC. The procedure invalid() is used to identify the validity of a vertex according to Lemmas 4 and 5. After popping an invalid vertex u in Line 6, we update neighbors of u if necessary. Let the color of u be $\mathcal{C}$. In Line 9, we decrease $deg_c(u)$ by one if no vertex has the color $\mathcal{C}$ in $N(v)$ after removing u. In Line 10, let $N_{\mathcal{C}}(v)$ be the set of all neighbors of v with the color $\mathcal{C}$. If u has the largest chi-square statistic in $N_{\mathcal{C}}(v)$, let s be the second largest chi-square statistic in $N_{\mathcal{C}}(v)$. We set $s = 0$ if there exists only u in $N_{\mathcal{C}}(v)$. We update $f_{cn}(v)$ to $f_{cn}(v) - f(u) + s$. We do not change $f_{cn}(v)$ if u is not the vertex with the largest chi-square statistic in $N_{\mathcal{C}}(v)$.

Algorithm 2. CSC(G, θ, k)

```
  // $f_n(u) = f(u) + \sum_{v \in N(u)} f(v)$, $f_{cn}(u)$ is colorful $f_n(u)$,
     $deg_c(u)$ is colorful $deg(u)$
 1: $Q \leftarrow$ initialized an empty queue;
 2: foreach $u \in V$ do
 3:    compute $f_{cn}(u)$, and $deg_c(u)$;
 4:    if invalid$(u)$ then $Q.push(u)$;
 5: while $Q \neq \emptyset$ do
 6:    $u \leftarrow Q.pop()$;
 7:    foreach $v \in N(u)$ do
 8:       if invalid$(v)$ then continue;
 9:       update $deg_c(v)$;
10:       update $f_{cn}(v)$;
11:       if invalid$(v)$ then $Q.push(v)$;
12:    remove $u$ and all connected edges;
 1: Procedure invalid$(u)$ :
 2:    if $deg_c(u) < k$ then return $true$;
 3:    if $f_{cn}(u) < \theta$ then return $true$;
 4:    return $false$;
```

Implementation and Complexity Analysis. The key step in Algorithm 2 is to efficiently maintain the colorful degree (Line 9) and colorful neighborhood significance (Line 10) of each vertex. For the colorful degree of a vertex u, we use a hash table to store the number of vertices for each color in colors$(N(u))$. Initializing the hash table for u takes $O(deg(u))$ time, and locating the value for a specific color number takes constant average and amortized time.

Next, we discuss the implementation to update the colorful neighborhood significance. For each color $\mathcal{C} \in$ colors$(N(v))$, we sort all vertices with the color $\mathcal{C}$ in a non-increasing order of their chi-square statistics, which takes $O(deg(v) \log deg(v))$ time. Given an invalid neighbor u, we also use a hash table to locate u in the sorted list and mark the position of u as empty. If u is not the first vertex in the corresponding color, $f_{cn}(v)$ does not need to be updated. If u is the first vertex, we iteratively search the following positions with the same color until finding a nonempty vertex w. We update $f_{cn}(v)$ to $f_{cn}(v) - f(u) + f(w)$, where $f(w) = 0$ if w does not exist. Note that it may take several movements to find w. However, the total number of operations for each vertex v in Line 10 of Algorithm 2 is bounded by $O(deg(v))$ since we always move forward to locate the second largest chi-square statistic.

Example 8. In Fig. 5, we give an example of the data structure to maintain $f_{cn}(v_{15})$ in the graph of Fig. 2. v_{15} has six neighbors with three distinct colors. $f_{cn}(v_{15})$ is initialized as the sum of $f(v_{15})$ and the chi-square statistics of all the first vertices of distinct colors. Assume that the vertex v_6 is removed, and we need to update $f_{cn}(v_{15})$. We first use the hash function to locate the the position of v_6. By checking the previous vertex, v_6 is not the first vertex in the table with the same color. Therefore, we set the position of v_6 as empty and do not change $f_{cn}(v_{15})$.

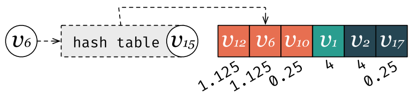


Fig. 5. The data structure for v_{15} to maintain $f_{cn}(v_{15})$.

We provide the theoretical analysis of Algorithm 2 in the following theorem.

Theorem 5. *The time complexity of Algorithm 2 is $O(m \cdot \log deg_{max})$, where deg_{max} is the maximum degree in the graph. The space complexity of Algorithm 2 is $O(m)$.*

4.3 Pruning via Significant Truss

Recall that Section 4.2 extends (k,θ)-SC and strengthens the vertex pruning rule. In this subsection, we will strengthen the (k,θ)-SC from the perspective of edge reduction. In other words, we consider whether two connected vertices can be in the same (k,θ)-clique or not. The lemma for the structural edge reduction is given as follows, which further applies the k-core concept to the ego-network of each vertex.

Lemma 6. *Given a k-clique C and an arbitrary vertex $u \in V_C$, for every vertex $v \in N_C(u)$, the vertex v is contained in a $(k-2)$-core of $G[N(u)]$ [23].*

Theorem 6. *Two vertices u and v cannot be contained in the same (k,θ)-clique if v is not in a $(k-2)$-core of the neighborhood subgraph $G[N(u)]$.*

Based on Theorem 6, if the vertex v is not in a $(k-2)$-core of $G[N(u)]$, we remove the edge (u,v), which guarantees that u and v cannot be enumerated in the same clique. Given the $O(m)$ time to compute the k-core, a straightforward method to recursively remove all unpromising edges in Theorem 6 takes $O(m \cdot h_{max})$ time, where h_{max} is the largest number of edges in neighborhood subgraphs. To remedy the cost, we give the following lemma.

Lemma 7. *Given a vertex u and a vertex v in $N(u)$, the degree of v in $G[N(u)]$ is equivalent to the number of triangles that contain the edge (u,v).*

Based on Lemma 7, removing all the edges not satisfying the condition in Theorem 6 is equivalent to computing the k-truss in graph, which is formally defined as follows.

Definition 9. *(k-Truss) Given a graph G and an integer k, a k-truss is a maximal connected subgraph in which every edge is contained in at least $k-2$ triangles [52].*

Let S be the maximal subgraph such that for each pair of connected vertices u and v in V_S, v is in $(k-2)$-core of $G[N(u)]$. Then, S is a k-truss of G.

Example 9. We consider the graph in Fig. 6. Given the vertex v_{12} and $k = 4$, the degree of v_{15} in the neighborhood

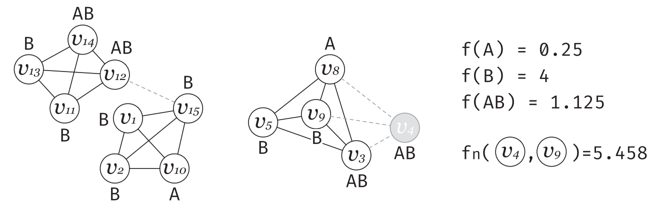


Fig. 6. Pruning G via (4,6)-significant truss.

subgraph of v_{12} is 0. In other words, there is no triangle containing the edge (v_{12}, v_{15}). Therefore, v_{12} and v_{15} cannot be in the same $(4, \theta)$-clique according to Theorem 6.

Similar to the neighborhood significance, we define the support significance for the statistical edge reduction.

Definition 10. *(Support Significance) The support significance of an edge (u, v), denoted by $f_n(u, v)$, is the sum of $f(u, v)$ and the chi-square statistics of all common neighbors of u and v, i.e., $f_n(u,v) = f(u,v) + \sum_{w \in N(u) \cap N(v)} f(w)$.*

Lemma 8. *Given a (k, θ)-significant clique C, we have $f_n(u, v) \geq \theta$ for every edge $(u, v) \in E_C$.*

Example 10. We consider the edge (v_4, v_9) in the graph of Fig. 6. There are two common neighbors — v_3 and v_8. We have $f_n(v_4, v_9) = f(v_4, v_9) + 1.125 + 0.25 = 5.458$. Based on Lemma 8, the edge (v_4, v_9) cannot be in any (k, θ)-significant clique if $\theta > 5.458$.

The support of an edge (u, v), denoted by $sup(u, v)$, is the number of triangles that contains (u, v). Based on Definitions 10 and 9, we define a new statistical cohesive subgraph model as follows.

Definition 11. *(Significant Truss) Given a graph G, an integer k and a positive real value θ, (k, θ)-Significant Truss (ST for short) is a maximal connected subgraph of G in which every edge (u, v) satisfies (i) $sup(u, v) \geq k - 2$, and (ii) $f_n(u, v) \geq \theta$.*

Example 11. An example of the pruning result by applying (4,6)-significant truss is given in Fig. 6. The edge (v_{12}, v_{15}) and all the edges connected to v_4 are removed.

The following theorem shows that (k, θ)-ST is guaranteed to have stronger pruning effectiveness than (k, θ)-SC.

Theorem 7. *A (k, θ)-significant truss must be contained in a $(k - 1, \theta)$-significant core.*

Proof. Given a (k, θ)-significant truss C and an arbitrary vertex $u \in C$, for each neighbor $v \in N(u) \cap C$, we have $f_n(u, v) \geq \theta$ (Lemma 8). Thus, $f_n(u) \geq \theta$. Also, we have $sup(u, v) \geq k - 2$ by which we can get $deg(u) \geq k - 1$. So, C is contained in a $(k - 1, \theta)$-significant core. □

ST Computation. We give the pseudocode for computing (k, θ)-significant truss in Algorithm 3. The idea is similar to that of truss decomposition [52]. The complexity of Algorithm 3 is summarized below.

Theorem 8. *The time complexity and space complexity of Algorithm 3 are $O(\alpha m)$ and $O(m)$, respectively.*

Proof. Lines 1–5 initialize the support and the support significance of each edge. The time complexity of enumerating all triangles is $O(\sum_{(u,v) \in E} \min(deg(u), deg(v)))$, i.e., $O(\alpha \cdot m)$, where $\alpha (\alpha < m^{0.5})$ is the graph arboricity and equals the minimum number of forests to cover all edges in the graph [53]. Lines 6–8 takes $O(m)$ time. We update necessary edges after (u, v) is removed in Line 10. We use a hash set to maintain all neighbors of each vertex. As a result, the time complexity for Lines 9–20 is $O(\alpha \cdot m)$. The total time complexity is $O(\alpha \cdot m)$. Note that we never store any triangle during the algorithm, thus the space complexity is $O(m)$. □

Remark: Similar to Section 4.2, we can further embed the color-based bound in the concept of (k, θ)-significant truss and improve the pruning effectiveness. However, maintaining such colorful supports and statistics for each edge incurs large space usage and processing time. Therefore, we only compute (k, θ)-significant truss in our final algorithm.

Algorithm 3. ST(G, θ, k)

```
1: $sup(u,v) \leftarrow 0, f_n(u,v) \leftarrow f(u,v)$;
2: foreach enumerated $\triangle_{u,v,w} \in G$ do
3:     $sup(u,v) \leftarrow sup(u,v) + 1$;
4:     $f_n(u,v) \leftarrow f_n(u,v) + f(w)$;
5:     repeat two lines above for $(u,w)$ and $(v,w)$;
6: $Q \leftarrow$ initialized an empty queue;
7: foreach $(u,v) \in E_G$ : invalid$(u,v,\theta,k-2)$ do
8:     $Q.push((u,v))$;
9: while $Q \neq \emptyset$ do
10:    $(u,v) \leftarrow Q.pop()$;
11:    if $deg(u) > deg(v)$ then swap $u$ and $v$;
12:    foreach $w \in N(u)$ do
13:        if $w \in N(v)$ then
14:            if invalid$(u,w,\theta,k)$ then continue;
15:            $sup(u,w) \leftarrow sup(u,w) - 1$;
16:            $f_n(u,w) \leftarrow f_n(u,w) - f(v)$;
17:            if invalid$(u,w,\theta,k)$ then
18:                $Q.push((u,w))$;
19:            repeat five lines above for $(v,w)$;
20:    remove $(u,v)$ from $E_G$;
21: remove all isolated vertices from $V_G$;
1: Procedure invalid$(u,v,\theta,k)$ :
2:     if $sup(u,v) < k-2$ then return $true$;
3:     if $f_n(u,v) < \theta$ then return $true$;
4:     return $false$;
```

5 The Final Algorithm

Our final algorithm to enumerate maximal (k, θ)-significant cliques is given in Algorithm 4. We reduce the input graph G by computing the $(k - 1, \theta)$-CSC and the (k, θ)-ST in Line 2 and Line 3, respectively. The order of CSC and ST does not affect the result. However, as CSC requires less computation than ST, we apply CSC first. Different from SigClique, the recursive enumeration procedure Enum* in SigClique* computes relaxed colorful $(k - 1, \theta)$-significant cores ($RCSC$) in Line 3 instead of $(k - 1)$-cores (Line 2) and color numbers (Line 4) of Enum.

Relaxed CSC. Given a graph G, an integer k and a real value θ, let S and S' be the vertices in (k, θ)-SC and (k, θ)-CSC, respectively. An induced subgraph of S'' is called a relaxed colorful (k,θ)-significant core if $S' \subseteq S'' \subseteq S$. The motivation of computing $RCSC$s is to hold the same $O(m)$ time complexity as k-core computation but bring stronger pruning effectiveness.

Recall that in Algorithm 2 to compute (k,θ)-CSCs, the dominating cost is to sort the neighbors of each vertex, which is used to efficiently update the colorful neighborhood significance (Fig. 5). To reduce the time complexity from $O(m \cdot \log deg_{max})$ to $O(m)$, we do not update the colorful neighborhood significance f_{cn} for each vertex in the iteration but just remove all vertices u with $f_{cn}(u) < k$ in the first round. Specifically, we modify Algorithm 2 to compute $RCSC$s in the following three parts. First, in Line 3, we additionally compute $f_n(u)$. Second, we replace Line 10 with "update $f_n(u)$". Third, we additionally check whether $f_n(u) < \theta$ in the procedure invalid. As a result, we derive a subgraph which satisfies the conditions of SC but possibly contains some vertices u with $f_{cn}(u) < \theta$. Due to the space limitation, we omit the detailed pseudocode of RCSC. Without the sorted data structure, the time complexity of RCSC reduces to $O(m)$, and the pruning effectiveness is at least the same as Line 2 and Line 4 in Enum.

Algorithm 4. SigClique$^*(G(V,E),\theta,k)$

```
1:  color $G$ based on the degeneracy order;
2:  CSC$(G,\theta,k-1)$;
3:  ST$(G,\theta,k)$;
4:  Enum$^*(V,\emptyset,\theta,k)$;
1:  **Procedure** Enum$^*(R,I,\theta,k)$ :
2:    RCSC$(R,\theta,k-1)$;
3:    **if** $R \cap I \neq I$ **then return**;
4:    **if** $R$ *is a* $(k,\theta)$*-significant clique* **then**
5:      **if** IsMax$(R,\bigcap_{v\in R} N(v),\theta)$ **then** output $R$;
6:      **return**;
7:    **if** $R = I$ **then return**;
8:    pick a vertex $u$ from $R \setminus I$;
9:    Enum$^*(I \cup N_R(u) \cup \{u\}, I \cup \{u\},\theta,k)$;
10:   Enum$^*(R \setminus \{u\}, I,\theta,k)$;
```

6 Experiments

In this section, we evaluate the efficiency and effectiveness of our proposed algorithms. We implement our algorithms with four versions according to the pruning techniques, namely SigClique, SigClique-SC, SigClique-CSC, and SigClique*. SigClique is the Algorithm 1 with basic k-core and graph coloring reduction techniques. SigClique-SC utilizes the significant core technique to prune the unpromising vertices. SigClique-CSC is the algorithm with the colorful significant core pruning rule. SigClique* contains all the pruning techniques. We also implement kClist, a variant of the k-clique listing algorithm [47]. We add our significance computation and maximality testing to make it able to enumerate sigcliques. All the algorithms are implemented in C++, and all the experiments are conducted on a Linux machine with 3.7 GHz Xeon CPU and 64 GB memory.

Datasets. We evaluate our algorithms with seven real-world datasets as showed in Table 1. EUmail is a communication network that contains the email sending and receiving information from a large European research institution. Amazon is a co-purchasing network from Amazon.com, where the vertices represent products, and the edges between them mean the two products are frequently purchased together. DBLP is a co-authorship network in which each vertex represents an author, and there is an edge between two vertices if they have co-authored at least three papers. Both Youtube and Hyves are social networks. CiteSeer and Patent are citation networks. On each graph, we randomly assign each vertex with 1-3 labels chosen from a 4-label alphabet. EUmail, Amazon, Youtube, and Patent are downloaded from the Stanford Large Network Dataset Collection.[2] CiteSeer and Hyves are downloaded from the Koblenz Network Collection.[3] DBLP is extracted from the computer science bibliography DBLP.[4]

TABLE 1
Network Statistics (deg_{max} is the Maximum Degree, c is the Core Value)

Dataset	n	m	deg_{max}	c
EUmail	265,214	420,045	7,636	37
Amazon	334,863	925,872	549	6
CiteSeer	384,413	1,751,463	1739	15
DBLP	556,533	1,311,766	373	25
Youtube	1,134,890	2,987,624	28,754	51
Hyves	1,402,673	2,777,419	31,883	39
Patent	3,774,768	16,518,948	793	64

Parameters. Our algorithms have two parameters, namely θ and k. The parameter θ is selected from the set $\{8, 10, 12, 14\}$ with the default value $\theta = 10$, and the parameter k is ranging from $\{3, 5, 7, 9\}$ with a default value $k = 5$. Unless otherwise specified, a parameter is set as the default value when we vary the other one. Regarding the efficiency evaluation, we prepare four labels $\Sigma = \{\mathsf{A},\mathsf{B},\mathsf{C},\mathsf{D}\}$. We randomly and independently assign a set of labels in Σ for each vertex in the graph. We assign the same proportion for all labels, and the expected label distribution is $P = \{0.25, 0.25, 0.25, 0.25\}$.

6.1 Efficiency Evaluation

Exp-1: Overall Efficiency of Maximal Significant Clique Enumeration. Table 2 compares the running time of SigClique* with SigClique and kClist under the default parameter setting. On the right side, we also show the number and the maximum size of corresponding maximal significant cliques. We can see that SigClique* is the fastest on most datasets. Compared to SigClique, the speedup in EUmail is small (from 8 seconds to 7 seconds) for the following reasons. First, EUmail has the smallest size in all datasets, and the improvement room is limited. Second, the average degree in EUmail is relatively small, which means the basic k-core pruning has been very effective. On Patent, SigClique* takes about 151 seconds while SigClique cannot finish in 5 hours. kClist is faster on Amazon and CiteSeer as both datasets have small core values and numbers of cliques, which means the listing can be fast. On other datasets with larger core values and numbers of cliques, our SigClique* is much faster.

2. https://snap.stanford.edu/
3. http://konect.cc/networks/
4. https://dblp.uni-trier.de/

TABLE 2
Overall Running Time of Enumerating on All the Datasets and the Number and the Maximum Size of Corresponding Maximal Significant Cliques

Dataset	SigClique	SigClique*	kClist	# sigcliques	k_{max}
EUmail	8 s	**7 s**	13 s	1198	15
Amazon	362 s	2 s	1 s	31	6
CiteSeer	340 s	8 s	2 s	137	9
DBLP	224 s	**74 s**	204 s	1388	23
Youtube	469 s	**86 s**	138 s	6036	16
Hyves	1605 s	**219 s**	525 s	998	14
Patent	-	**151 s**	177 s	1790	9

Exp-2: Running Time of Different Pruning Strategies With Varying Parameters. We evaluate the running time of our algorithms by varying the input parameter k and θ. Due to the space limitation, we only report the results on two representative datasets, i.e., Youtube and Patent. The results on other datasets perform similar trends. Given the default $k = 5$, we vary θ from 8 to 14 in Figs. 7a and 7b. We only have the line for SigClique* on Patent since the algorithms cannot finish in 5 hours under other settings. We can see a downward trend for the algorithms on both Youtube and Patent when θ increases. On Patent, the running time of SigClique* is about 1.25 hours when $\theta = 8$ and drops to 25 seconds when $\theta = 14$. Fig. 7b also reveals the effectiveness of our several pruning techniques. SigClique is the slowest algorithm under all θ values since it only uses the basic k-core and the graph coloring pruning rules to reduce the search space. SigClique-SC starts to consider statistical pruning, which is a little faster than SigClique. When θ is small, the running time of SigClique-SC is similar to SigClique, because the degree pruning dominates in the algorithm. The gap between SigClique-SC and SigClique proves the effectiveness of the significant core model. SigClique-CSC is the second fastest algorithm in the figure since it adopts a stronger pruning model than SigClique-SC. The gap between SigClique-CSC and SigClique-SC proves the effectiveness of the colorful significant core model. SigClique* contains all optimizations, which is the fastest. The gap between SigClique* and SigClique-CSC proves the effectiveness of the significant truss model.

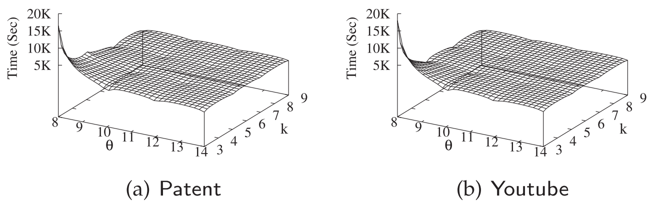


Fig. 8. Running time of SigClique* by varying θ and k.

Given the default $\theta = 10$, we vary k from 3 to 9 in Figs. 7c and 7d. The results are similar to those when varying θ. For example, on Patent, SigClique* takes 175 seconds and 12 seconds when $k = 3$ and $k = 9$, respectively. In Fig. 7d, the running time of SigClique is almost the same as that of SigClique-SC when k is large, because θ does not change and the degree pruning is the dominating rule. Fig. 8 shows the running time of SigClique* on Patent and Youtube while varying the θ and k.

Exp-3: Number of Vertices After Reduction. To further evaluate our pruning techniques, we report the number of vertices after performing the reduction rules (e.g., core, significant core, colorful significant core, and significant truss) and before invoking the Enum procedure in Fig. 9. As shown in Figs. 9a and 9b, when varying θ, we can see that the number of vertices for SigClique never changes since only the structural pruning is performed in SigClique. Note that even though the significant truss is an edge pruning model, many vertices become isolated and are removed during the edge removal. The results show that our final pruning techniques are extremely effective. For example, in Fig. 9a, the number of vertices is about 194 thousand when $\theta = 8$ but drops to only about 17 thousand when $\theta = 14$.

Exp-4: Number of Maximal Significant Cliques. We report the number of maximal significant cliques under different parameters in Fig. 10. The number of results gradually decreases when θ increases from 8 to 14 on both datasets. For example, in Fig. 10a, we have over 12 thousand maximal (5,8)-significant cliques and 154 maximal (5,14)-significant cliques on Patent. In Fig. 10c, we see a sharp drop from $k = 7$ to $k = 9$. That is because there exist a larger number of (7,10)-cliques or (8,10)-cliques on Patent.

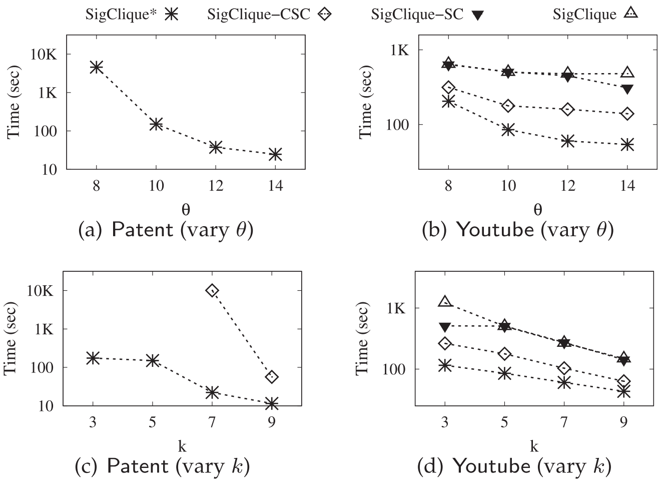


Fig. 7. Running time of different algorithms by varying θ and k.

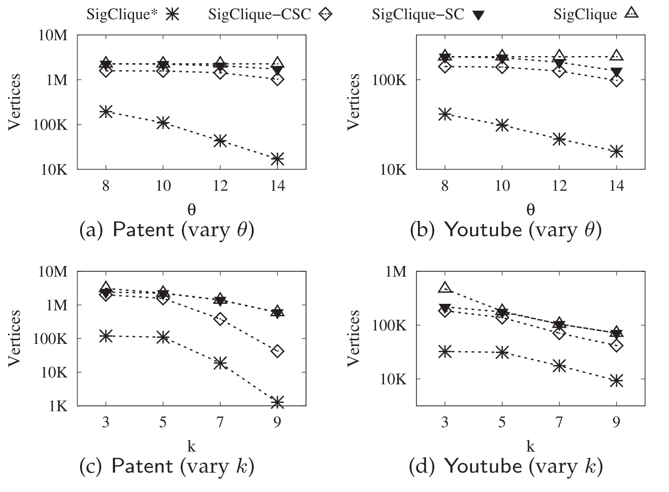


Fig. 9. The number of vertices after pruning by varying θ and k.

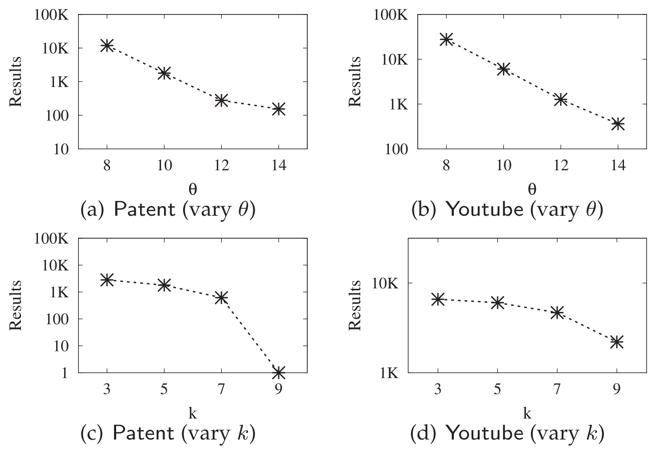


Fig. 10. The number of maximal (θ, k)-cliques.

Exp-5: Scalability Testing. We evaluate the scalability of our final algorithm SigClique* on Patent with the baseline SigClique as a comparison. The results on other datasets show similar trends. We vary the graph size and density by randomly sampling vertices and edges from 20% to 100%, respectively. When sampling vertices, we derive the induced subgraph of the sampled vertices, and when sampling edges, we select the incident vertices of the edges as the vertex set. The results are shown in Fig. 11. We do not have results of SigClique under several settings since it cannot finish in 5 hours. The results show that our algorithm is scalable to large graphs. In Fig. 11a, the time of SigClique* is 0.5 s when sampling 20% vertices and increases to 151 seconds finally.

6.2 Case Study on DBLP

To evaluate the effectiveness of our model, we conduct a case study on DBLP and compare it with the maximal clique model. We generate a collaboration network for the DBLP, where an edge connects two researchers if they have at least three co-authored publications. Regarding the vertex labels, we select the publication data for each researcher from three major conferences in the database area — SIGMOD, PVLDB, and ICDE over the last decade. We assign each vertex the label that indicates the author has a published paper on the corresponding conference. Specifically, for each conference every year, given a researcher u, we assign u a conference name if u has a publication in the conference and an empty value $\emptyset$ otherwise. Note that the assigned labels can be repeated for each researcher.

For example, assume that u has one SIGMOD publication, two PVLDB publications, and no ICDE publication in one year. The added labels in such a year for u are $\{\text{SIGMOD}, \text{PVLDB}, \text{PVLDB}, \emptyset\}$. Given the impact factor of these conferences, we expect the proportion of label frequencies for SIGMOD, PVLDB, ICDE, and $\emptyset$ (no paper published) as $1:2:3:30$. The rationale is that we give a relatively low expected frequency for the highest-ranked conference. If a researcher does not have any publication in the last decade, they will have thirty $\emptyset$ labels. Intuitively, a significant clique with a large chi-square value in our setting may represent a research group with many high-ranked conference publications.

In Figs. 12a, 12b, and 12c, we show all the maximal significant cliques containing the Prof. Jiawei Han with the parameters $k = 5$ and $\theta = 500$. The parameter setting is application-oriented, where generally, the larger the k or θ, the fewer the resulting subgraphs. As we can see, there are three $(5, 500)$-cliques reported by our model. The chi-square statistics of the results in Figs. 12a, 12b, and 12c are 828.639, 589.864, and 722.331, respectively.

In comparison, we also enumerate all the maximal cliques containing Prof. Jiawei Han with at least 5 vertices. There are 33 resulting subgraphs, including two 7-cliques, four 6-cliques, and 27 5-cliques. Due to the large result size, we show some representatives of them in Fig. 13. We can see that the number of maximal cliques is much more than that of the $(5, 500)$-cliques. We find that the main research interests of many resulting authors do not lay in the database area. For example, Prof. Tarek F. Abdelzaher, Prof. Su Lu, and Dr. Shaohan Hu may be mainly interested in the

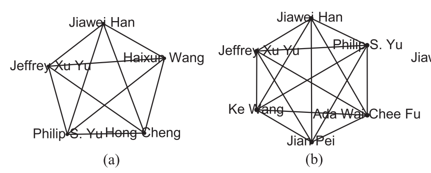


Fig. 12. The maximal (5, 500)-significant cliques of Prof. Jiawei Han on DBLP graph.

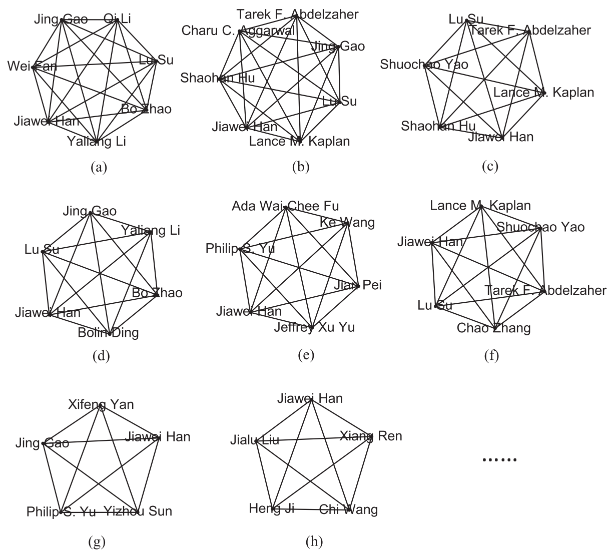


Fig. 13. The maximal cliques containing Prof. Jiawei Han on DBLP graph with at least 5 vertices. (a) and (b) are 7-cliques. (c)-(f) are 6-cliques. (g) and (h) are 5-cliques.

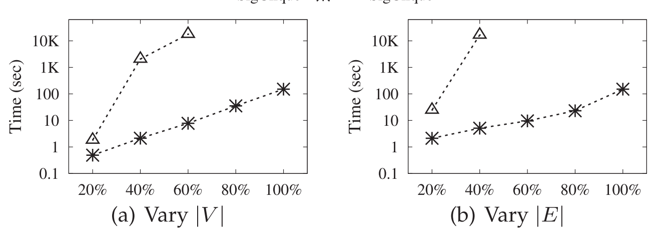


Fig. 11. Scalability testing on Patent.

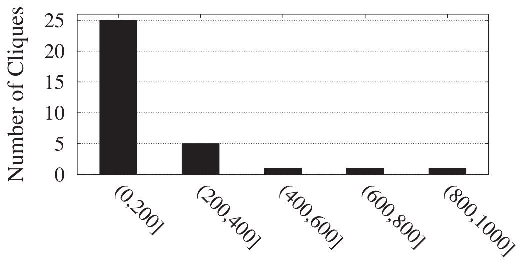


Fig. 14. The chi-square statistic of all the maximal cliques containing Prof. Jiawei Han with at least 5 vertices.

Internet of Things, Cyber-Physical Systems, and Mobile Computing since they have contributed a lot to the corresponding research area. However, they all have co-authored only one PVLDB paper in the past decade. According to historical publications, they should be excluded if we want to find significant database communities.

We have a consistent conclusion when considering the chi-square statistics of the resulting maximal cliques. In Fig. 14, we group all the 33 maximal cliques (with size at least 5) containing Prof. Han into five groups by corresponding chi-square statistics. We can see that the chi-square statistics of most results are relatively low, which means the resulting communities have rather less significance in the database area. For example, there are 25 results whose chi-square statistics lay in the interval (0,200]. By contrast, all the resulting three (5,500)-significant cliques reported by our model have much higher chi-square statistics. Note that the three maximal (5,500)-significant cliques are also included in the results of maximal cliques.

7 Related Works

Significant Sub-Structures on Graphs. Many real-world applications rely on exploiting statistically significant sub-structures on graphs, which include significant paths [54], trees [55], and subgraphs [12], [13]. Zhang et al. propose a sampling method based on modularity to detect significant communities on graphs [56]. He et al. utilize the p-value bound to develop a local search algorithm to find significant subgraphs [57]. A brief survey on significant sub-structures can be found in [58]. However, most previous works are tailored to unlabeled graphs that ignore the label information on vertices. Arora et al. propose a statistically significant connected subgraph model, which depicts the label figure with chi-square statistics [14]. This model may not be applicable to our problem, as the connection between the nodes inside a connected subgraph may be very loose.

Community Modeling. The community models over graphs have been extensively explored. Communities are often modeled by a group of densely connected nodes. In the literature, various community models and algorithms have been proposed, which include clique [24], [43], [44], k-core [40], [51], k-truss [52], k-clan [59], k-plex [60], and so on. In recent years, more models for community detection that consider graph label information have also been developed. Notable examples include the k-core-based attributed community model [37], the truss-based attributed community model [38], the keyword-centric attributed community searching model [39]. However, all the above models never consider the statistical significance of the discovered community, which cannot be directly applied to our problem.

Maximal Clique Enumeration. The clique model has a wide range of applications. The enumeration of all maximal cliques in a graph has long been a popular problem in graph data mining. Many existing works have been put forward to study the problem. Most of them are based on a backtracking diagram [44], [45], [46]. Tomita et al. [45] propose a pivoting technique which is proved to be worst-case optimal. [46] further improves the time complexity of maximal clique enumeration on sparse graphs. Jin et al. [61] propose an approach combining a hybrid data structure and a new pivot selection rule to accelerate the enumeration. [43] and [62] propose I/O efficient and distributed algorithms, respectively. Chang et al. [20] proposes an algorithm to progressively compute maximal cliques in polynomial delay.

8 Conclusion

In this paper, we formulate a new model, called maximal (k,θ)-significant clique, to capture statistically significant cohesive subgraphs in large labeled graphs. We propose a novel branch-and-bound algorithm and several effective pruning rules to enumerate all maximal (k,θ)-significant clique. Extensive experiments are conducted to show the efficiency of our algorithms.

References

[1] L. Freeman, *The Development of Social Network Analysis: A Study in the Sociology of Science*, Vancouver, BC, Canada: Empirical Press, 2004. [Online]. Available: https://books.google.com.au/books?id=VcxqQgAACAAJ
[2] E. Otte and R. Rousseau, “Social network analysis: A powerful strategy, also for the information sciences,” *J. Inf. Sci.*, vol. 28, no. 6, pp. 441–453, 2002.
[3] N. Pržulj, “Biological network comparison using graphlet degree distribution,” *Bioinformatics*, vol. 23, no. 2, pp. e177–e183, 2007.
[4] G. A. Pavlopoulos et al., “Using graph theory to analyze biological networks,” *Bio Data Mining*, vol. 4, no. 1, pp. 1–27, 2011.
[5] M. Tomassini and L. Luthi, “Empirical analysis of the evolution of a scientific collaboration network,” *Physica A: Stat. Mechanics Appl.*, vol. 385, no. 2, pp. 750–764, 2007.
[6] D. Surian, N. Liu, D. Lo, H. Tong, E.-P. Lim, and C. Faloutsos, “Recommending people in developers’ collaboration network,” in *Proc. 18th Work. Conf. Reverse Eng.*, 2011, pp. 379–388.
[7] M. Sozio and A. Gionis, “The community-search problem and how to plan a successful cocktail party,” in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2010, pp. 939–948.
[8] W. Cui, Y. Xiao, H. Wang, and W. Wang, “Local search of communities in large graphs,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 991–1002.
[9] X. Li, M. Wu, C.-K. Kwoh, and S.-K. Ng, “Computational approaches for detecting protein complexes from protein interaction networks: A survey,” *BMC Genomic.*, vol. 11, no. 1, 2010, Art. no. S3.
[10] J. Pei, D. Jiang, and A. Zhang, “Mining cross-graph quasi-cliques in gene expression and protein interaction data,” in *Proc. 21st Int. Conf. Data Eng.*, 2005, pp. 353–354.
[11] H. He and A. K. Singh, “GraphRank: Statistical modeling and mining of significant subgraphs in the feature space,” in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 885–890.
[12] S. Ranu and A. K. Singh, “GraphSig: A scalable approach to mining significant subgraphs in large graph databases,” in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2009, pp. 844–855.
[13] X. Yan, H. Cheng, J. Han, and P. S. Yu, “Mining significant graph patterns by leap search,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 433–444.

[14] A. Arora, M. Sachan, and A. Bhattacharya, "Mining statistically significant connected subgraphs in vertex labeled graphs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 1003–1014.

[15] T. R. Read and N. A. Cressie, *Goodness-of-Fit Statistics for Discrete Multivariate Data*. Berlin, Germany: Springer, 2012.

[16] A. Denise, M. Régnier, and M. Vandenbogaert, "Assessing the statistical significance of overrepresented oligonucleotides," in *Proc. Int. Workshop Algorithms Bioinf.*, 2001, pp. 85–97.

[17] K. Wongpanya, K. Sripimanwat, and K. Jenjerapongvej, "Simplification of frequency test for random number generation based on chi-square," in *Proc. 4th Adv. Int. Conf. Telecommun.*, 2008, pp. 305–308.

[18] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Qual. Rel. Eng. Int.*, vol. 17, no. 2, pp. 105–112, 2001.

[19] J. W. Moon and L. Moser, "On cliques in graphs," *Isr. J. Math.*, vol. 3, no. 1, pp. 23–28, 1965.

[20] L. Chang, J. X. Yu, and L. Qin, "Fast maximal cliques enumeration in sparse graphs," *Algorithmica*, vol. 66, no. 1, pp. 173–186, 2013.

[21] J. Cheng, Y. Ke, A. W. Fu, J. X. Yu, and L. Zhu, "Finding maximal cliques in massive networks by h*-graph," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 447–458.

[22] E. A. Akkoyunlu, "The enumeration of maximal cliques of large graphs," *SIAM J. Comput.*, vol. 2, no. 1, pp. 1–6, 1973.

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

[24] 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.

[25] R.-H. Li et al., "Efficient signed clique search in signed networks," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 245–256.

[26] A. Himmel, H. Molter, R. Niedermeier, and M. Sorge, "Enumerating maximal cliques in temporal graphs," in *Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Mining*, 2016, pp. 337–344.

[27] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Mining maximal cliques from an uncertain graph," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 243–254.

[28] M. B. Jdidia, C. Robardet, and E. Fleury, "Communities detection and analysis of their dynamics in collaborative networks," in *Proc. 2nd Int. Conf. Digit. Inf. Manage.*, 2007, pp. 744–749.

[29] A. Conte, R. De Virgilio, A. Maccioni, M. Patrignani, and R. Torlone, "Finding all maximal cliques in very large social networks," in *Proc. 19th Int. Conf. Extending Database Technol.*, 2016, pp. 173–184.

[30] G. Liu, L. Wong, and H. N. Chua, "Complex discovery from weighted PPI networks," *Bioinformatics*, vol. 25, no. 15, pp. 1891–1897, 2009.

[31] S. A. Moosavi, M. Jalali, N. Misaghian, S. Shamshirband, and M. H. Anisi, "Community detection in social networks using user frequent pattern mining," *Knowl. Inf. Syst.*, vol. 51, no. 1, pp. 159–186, 2017.

[32] A. Prado, M. Plantevit, C. Robardet, and J. Boulicaut, "Mining graph topological patterns: Finding covariations among vertex descriptors," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 9, pp. 2090–2104, Sep. 2013.

[33] L. Chu, Z. Wang, J. Pei, Y. Zhang, Y. Yang, and E. Chen, "Finding theme communities from database networks," *Proc. VLDB Endowment*, vol. 12, no. 10, pp. 1071–1084, 2019.

[34] M. Berlingerio, F. Pinelli, and F. Calabrese, "ABACUS: Frequent pattern mining-based community discovery in multidimensional networks," *Data Mining Knowl. Discov.*, vol. 27, no. 3, pp. 294–320, 2013.

[35] F. Moser, R. Colak, A. Rafiey, and M. Ester, "Mining cohesive patterns from graphs with feature vectors," in *Proc. SIAM Int. Conf. Data Mining*, 2009, pp. 593–604.

[36] A. Prateek, A. Khan, A. Goyal, and S. Ranu, "Mining top-k pairs of correlated subgraphs in a large network," *Proc. VLDB Endowment*, vol. 13, no. 9, pp. 1511–1524, 2020.

[37] Y. Fang, R. Cheng, S. Luo, and J. Hu, "Effective community search for large attributed graphs," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1233–1244, 2016.

[38] X. Huang and L. V. Lakshmanan, "Attribute-driven community search," *Proc. VLDB Endowment*, vol. 10, no. 9, pp. 949–960, 2017.

[39] Z. Zhang, X. Huang, J. Xu, B. Choi, and Z. Shang, "Keyword-centric community search," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 422–433.

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

[41] E. Tomita, "Efficient algorithms for finding maximum and maximal cliques and their applications," in *Proc. Int. Workshop Algorithms Comput.*, 2017, pp. 3–15.

[42] Full version, 2022. [Online]. Available: https://www.dropbox.com/s/suc9ies7z2ibbfk/Computing_Significant_Cliques_in_Large_Labeled_Networks_Full_Version.pdf?dl=0

[43] J. Cheng, Y. Ke, A. W. Fu, J. X. Yu, and L. Zhu, "Finding maximal cliques in massive networks," *ACM Trans. Database Syst.*, vol. 36, no. 4, pp. 21:1–21:34, 2011.

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

[45] 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.

[46] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," *ACM J. Exp. Algorithmics*, vol. 18, 2013, Art. no. 3.1.

[47] M. Danisch, O. D. Balalau, and M. Sozio, "Listing k-cliques in sparse real-world graphs," in *Proc. World Wide Web Conf.*, 2018, pp. 589–598.

[48] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, "Parallel maximum clique algorithms with applications to network analysis," *SIAM J. Sci. Comput.*, vol. 37, no. 5, pp. C589–C616, 2015.

[49] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," 2003, *arXiv:0310049*.

[50] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. New York, NY, USA: Plenum Press, 1972, pp. 85–103.

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

[52] J. Wang and J. Cheng, "Truss decomposition in massive networks," *Proc. VLDB Endowment*, vol. 5, no. 9, pp. 812–823, 2012.

[53] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM J. Comput.*, vol. 14, no. 1, pp. 210–223, 1985.

[54] K. Tsuda, "Entire regularization paths for graph data," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 919–926.

[55] H. He and A. K. Singh, "Efficient algorithms for mining significant substructures in graphs with quality guarantees," in *Proc. IEEE 7th Int. Conf. Data Mining*, 2007, pp. 163–172.

[56] P. Zhang and C. Moore, "Scalable detection of statistically significant communities and hierarchies, using message passing for modularity," *Proc. Nat. Acad. Sci. USA*, vol. 111, no. 51, pp. 18144–18149, 2014.

[57] Z. He, H. Liang, Z. Chen, C. Zhao, and Y. Liu, "Computing exact p-values for community detection," *Data Mining Knowl. Discov.*, vol. 34, pp. 833–869, 2020.

[58] H. Cheng, X. Yan, and J. Han, "Mining graph patterns," in *Frequent Pattern Mining*. Berlin, Germany: Springer, 2014, pp. 307–338.

[59] R. J. Mokken et al., "Cliques, clubs and clans," *Qual. Quantity*, vol. 13, no. 2, pp. 161–173, 1979.

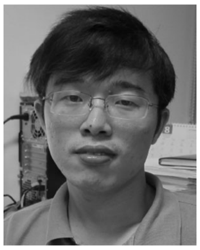
[60] D. Berlowitz, S. Cohen, and B. Kimelfeld, "Efficient enumeration of maximal k-plexes," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 431–444.

[61] Y. Jin, B. Xiong, K. He, Y. Zhou, and Y. Zhou, "On fast enumeration of maximal cliques in large graphs," *Expert Syst. Appl.*, vol. 187, 2022, Art. no. 115915.

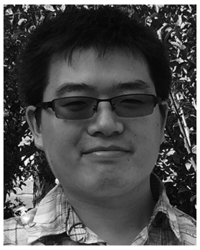
[62] Y. Xu, J. Cheng, A. W.-C. Fu, and Y. Bu, "Distributed maximal clique computation," in *Proc. Int. Congr. Big Data*, 2014, pp. 160–167.

Yu-Xuan Qiu received the BE and ME degrees in computer science from Shenzhen University, in 2015 and 2018, respectively. He is currently working toward the PhD degree with the Australian Artificial Intelligence Institute, University of Technology Sydney. His current research interests include graph data management and mining.

Dong Wen reveived the BEng degree from Nankai University, in 2013, and the PhD degree from the Faculty of Engineering and Information Technology, University of Technology Sydney, in 2019. He is currently a lecturer with the University of New South Wales. His research interests include I/O efficient graph processing and streaming graph analysis.



Rong-Hua Li received the PhD degree from the Chinese University of Hong Kong, in 2013. He is currently an associate professor with the Beijing Institute of Technology (BIT), Beijing, China. Before joining BIT, in 2018, he was an assistant professor with Shenzhen University. His research interests include graph data management and mining, social network analysis, graph computation systems, and graph-based machine learning.



Lu Qin received the BEng degree from the Department of Computer Science and Technology, Renmin University of China, in 2006, and the PhD degree from the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, in 2010. He is currently an associate professor with the Centre for Artificial Intelligence, University of Technology, Sydney. His research interests include big graph analytics and graph query processing.

Michael Yu received his BSc(Hons) degree in computer science from the University of New South Wales, in 2017 and currently working toward the PhD degree under the Data and Knowledge Research Group with the School of Computer Science and Engineering, UNSW. His research interests encompass topics that study large-scale graph data management and mining.

Xuemin Lin (Fellow, IEEE) received the BSc degree in applied math from Fudan University, in 1984, and the PhD degree in computer science from the University of Queensland, in 1992. He is a chair professor and the head of Data and Business Intelligence with the Antai College of Economics & Management, Shanghai Jiaotong University. His current research interests lie in approximate query processing, spatial data analysis, and graph processing and visualization.