Maximal Defective Clique Enumeration

QIANGQIANG DAI, RONG-HUA LI, MEIHAO LIAO, and GUOREN WANG, Beijing Institute of Technology, China

Maximal clique enumeration is a fundamental operator in graph analysis. The model of clique, however, is typically too restrictive for real-world applications as it requires an edge for every pair of vertices. To remedy this restriction, practical graph analysis applications often resort to find relaxed cliques as alternatives. In this work, we investigate a notable relaxed clique model, called s-defective clique, which allows at most s edges to be missing. Similar to the complexity of maximal clique enumeration, the problem of enumerating all maximal s-defective cliques is also NP-hard. To solve this problem, we first develop a new polynomial-delay algorithm based on a carefully-designed reverse search technique, which can output two consecutive results within polynomial time. To achieve better practical efficiency, we propose a branch-and-bound algorithm with a novel pivoting technique. We prove that the time complexity of this algorithm depends only on $O(\alpha_s^n)$ or $O(\alpha_s^\delta)$ when using a degeneracy ordering optimization, where α_s is a positive real number strictly less than 2, and δ ($\delta < n$) is the degeneracy of the graph. To our knowledge, this is the first algorithm that can break the $O(2^n)$ time complexity to enumerate all maximal s-defective cliques ($s > 0$). We also develop several new pruning techniques to further improve the efficiency of our branch-and-bound algorithm to enumerate all relatively-large maximal s-defective cliques. In addition, we further generalize our pivot-based branch-and-bound algorithm to enumerate all maximal subgraphs satisfying a hereditary property. Here we call a graph meeting the hereditary property if all its subgraphs have the same property as itself. Finally, extensive experiments on 11 datasets demonstrate the efficiency, effectiveness, and scalability of the proposed solutions.

CCS Concepts: • **Mathematics of computing → Graph enumeration**.

Additional Key Words and Phrases: cohesive subgraph, s-defective clique, reverse search, branch-and-bound

ACM Reference Format:
Qiangqiang Dai, Rong-Hua Li, Meihao Liao, and Guoren Wang. 2023. Maximal Defective Clique Enumeration. *Proc. ACM Manag. Data* 1, 1, Article 77 (May 2023), 26 pages. https://doi.org/10.1145/3588931

1 INTRODUCTION

Mining cohesive subgraphs in real-world networks is a fundamental problem in graph analysis. There are numerous applications that can be modeled as a cohesive subgraph mining problem, including detecting communities in social networks [5, 20, 27], mining protein complexes in protein-protein interaction (PPI) networks [24, 50], and statistical analysis in financial networks [8, 9]. The classic maximal clique model which requires an edge for every pair of vertices is widely used to represent cohesive subgraphs, due to its densely-connected interiors and the existence of many advanced approaches to find all maximal cliques [10, 19, 21, 38, 43, 47].

However, the condition that requires all possible relations to exist in the community may be too restrictive for real-world applications, since noises or faults may occur in real-world networks [17]

Authors' address: Qiangqiang Dai, qiangd66@gmail.com; Rong-Hua Li, lironghuabit@126.com; Meihao Liao, mhliao@bit.edu.cn; Guoren Wang, wanggrbit@126.com, Beijing Institute of Technology, Beijing, China.

2836-6573/2023/5-ART77 $15.00
https://doi.org/10.1145/3588931

and the interactions between individuals in a community can be accomplished through non-direct relationships [40]. To remedy this issue, many relaxed clique models have been developed as alternatives to represent cohesive subgraphs, such as s-defective clique [50], k-plex [7, 17, 44, 54], r-clique [6, 35], and γ-quasi-clique [34, 41]. In this paper, we focus mainly on the s-defective clique model, as it can well approximate the clique model [50].

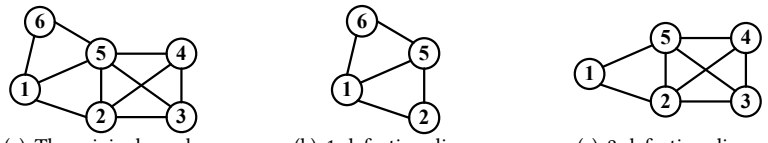
Given a graph G, a subset S of vertices in G is a maximal s-defective clique if (1) the subgraph induced by S has at least $\binom{|S|}{2} - s$ edges, and (2) there does not exist any vertex subset satisfying (1) and containing S. For example, consider the graph G shown in Fig. 1(a). It is easy to check that Fig.1(b) is a maximal 1-defective clique and Fig. 1(c) is a maximal 2-defective clique. Clearly, the clique is the special case of the s-defective clique when $s = 0$. In addition, many relaxed clique models are also closely related to the s-defective clique. For instance, an s-defective clique must be an $(s + 1)$-plex, where a k-plex is a subgraph in which every vertex is adjacent to all but at most k vertices [44]. For any s-defective clique with size larger than $s + 1$, it is also a 2-clique [35] (a subgraph in which the distance between each pair of vertices is no larger than 2 in the original graph), since the diameter of such an s-defective clique is always no larger than 2 (see Property 2).

Compared to another classic relaxed clique model γ-quasi-clique [41], the s-defective clique model also has its own advantages. Specifically, the key difference between these two models is that the s-defective clique satisfies the *hereditary* property (Property 1 in Section 2), while the γ-quasi-clique does not. As a consequence, s-defective clique has two nice features. The first is that the structure of the s-defective clique is typically more robust than that of the γ-quasi-clique. This is because the removal of any subset from an s-defective clique does not break its structure (i.e., the remaining subgraph is still an s-defective clique). The second is that the algorithms for enumerating s-defective cliques are usually much more efficient than those for enumerating γ-quasi-cliques, since the maximality checking procedure for hereditary subgraph models is much easier than those models that do not satisfy the hereditary property. Based on these analyses, the s-defective clique model can be used in many real-world applications, and three concrete examples are listed below.

Implicit interactions prediction. Intuitively, given any two non-adjacent vertices, if their common neighbors in the group form a clique, then these two vertices are likely to be connected. This means that the missing edges in an s-defective clique have a strong prediction for the implicit interactions in the graph if s is small. Thus, in real-world networks, such as PPI networks, collaboration networks, and social networks, one can make use of s-defective cliques to predict implicit interactions. A notable example is that Yu et al. [50] have successfully applied the s-defective clique model to detect implicit protein interactions in PPI networks, which is mainly based on the fact that the protein complex is often considered as a complete subgraph [3, 23].

Community detection in social networks. In an s-defective clique, it only allows at most s edges to be missing, which ensures that each s-defective clique is densely-connected when s is small. In real-world networks, such as social networks, communication networks, and web graphs, the communities usually can be modeled as tightly-connected subgraphs. Note that the classic clique model has been widely used for these tasks [5, 11, 39, 46]. However, real-life networks are usually constructed from empirical data, and there likely exist several links missed during the data collection procedure [7]. Thus, it is more suitable to adopt the s-defective clique model to identify communities compared to the clique model, as they allow up to s missing links. As shown in our experiments (Exp-7 and Exp-8), the s-defective clique model can indeed detect meaningful communities that cannot be identified by the classic clique model.

Statistical analysis in financial networks. In financial networks, such as the stock market network, the vertices and edges can be represented as financial instruments and the correlations



(a) The original graph (b) 1-defective clique (c) 2-defective clique

Fig. 1. A running example graph G.

between instruments, respectively. Real-world applications often need to identify groups of financial instruments in such networks whose prices may collectively fluctuate. The classic clique model has been successfully used for these applications [8, 9]. Since a clique is a special case of an s-defective clique, the use of s-defective clique can be a good alternative for these applications. Moreover, compared to the classic clique model, a striking advantage of the s-defective clique model is that it can also identify the indirectly-linked but closely-related financial instruments.

Although the maximal s-defective clique model can be used for many applications, an efficient algorithm for enumerating all maximal s-defective cliques on large graphs is still lacking. To our knowledge, the only existing algorithm to identify maximal s-defective cliques was proposed in [50]. The main idea of this algorithm is that it first makes use of the Bron-Kerbosch (BK) algorithm [10] to enumerate all maximal cliques and then determines whether each possible combination of maximal cliques can form an s-defective clique. Clearly, such an algorithm is rather inefficient and requires very high memory overheads to store the maximal cliques. More importantly, this algorithm typically cannot obtain all maximal s-defective cliques of a graph. For instance, consider the graph G shown in Fig. 1(a), it is easy to see that $\{v_1, v_2, v_3, v_5\}$ is a maximal 1-defective clique. Such a maximal 1-defective clique, however, does not belong to any pair of maximal cliques, thus cannot be identified by the algorithm proposed in [50].

To overcome this problem, we develop two novel approaches to enumerate all maximal s-defective cliques. We show that both of our proposed algorithms have non-trivial worst-case time complexity guarantees. To our knowledge, our algorithms are the best algorithms so far in terms of the worst-case time complexity. The main contributions of our work are summarized as follows.

A novel polynomial-delay algorithm. Our first solution for enumerating all maximal s-defective cliques is a polynomial-delay algorithm based on a reverse search technique [2], which can output any two consecutive results within polynomial time. Specifically, we first propose a novel graph-like structure to characterize the relationships of all maximal s-defective cliques in a given graph. Such a new structure provides us with guidelines on which possible results can be jumped from one to the other. Based on this, we then propose a reverse search procedure to recursively enumerate each result. We prove that the delay of any two consecutive results output by our algorithm is bounded in $O(n^2 \frac{\Delta^{2s+1}}{4^s})$ time, where n is the number of vertices of the graph, Δ is the size of the maximum s-defective clique, and s is a small constant. To our knowledge, this is the first theoretical result on enumerating maximal s-defective cliques within polynomial-delay time.

New branch-and-bound enumeration algorithms. Although our polynomial-delay algorithm has a nice theoretical property, its practical performance may be inefficient for large real-world graphs. To obtain a more practical solution, we propose a new branch-and-bound enumeration algorithm with a carefully-designed pivoting technique. We prove that the time complexity of our branch-and-bound algorithm depends mainly on $O(\alpha_s^n)$, where α_s is a positive real number and strictly less than 2. To our knowledge, this is the first algorithm that breaks the $O(2^n)$ barrier of the basic branch-and-bound algorithm for maximal s-defective clique enumeration. To boost the efficiency, we also develop an improved branch-and-bound algorithm with a degeneracy-ordering optimization technique. We show that the worst-case time complexity of such an improved algorithm depends mainly on $O(\alpha_s^\delta)$. Here δ denotes the degeneracy of the graph which is often

very small for large real-world graphs [21, 31]. In addition, several non-trivial and efficient pruning techniques are also proposed to further improve the performance of our algorithms.

A general pivoting paradigm. Inspired by the proposed pivot-based branch-and-bound algorithms, we further develop a general pivoting paradigm for enumerating all maximal subgraphs that satisfy the hereditary property. Here we call a graph meeting the hereditary property if all its subgraphs have the same property as itself (e.g., both cliques and s-defective cliques satisfy the hereditary property). We believe that the proposed general pivoting paradigm could be of independent interests, which can provide useful guidelines to devise efficient pivot-based algorithms for other maximal hereditary subgraph enumeration problems.

Extensive experimental evaluations. We conduct extensive experiments to evaluate the efficiency, effectiveness, and scalability of the proposed algorithms on 11 real-world graphs. The experimental results show that (1) the proposed polynomial-delay algorithm performs well in terms of time delay and also achieve good practical performance when processing small dense graphs; (2) the proposed pivot-based algorithms consistently achieve the best performance, which can be several orders of magnitude faster than the baseline algorithms. In addition, three concrete case studies further illustrate the effectiveness of the studied model. For reproducibility purpose, the source code of this work is available at https://github.com/qq-dai/DefectiveClique.

2 PROBLEM STATEMENT

Let $G = (V, E)$ be the undirected and unweighted graph, where V and E are the set of vertices and edges respectively. Denote by $n = |V|$ and $m = |E|$ the number of vertices and edges of G, respectively. The complementary graph of G is denoted by $\overline{G} = (V, \overline{E})$, where each edge $(u, v) \in \overline{E}$ if and only if $(u, v) \notin E$. For a vertex v, the set of neighbors in G is defined as $N_v(G) = \{u \in V | (v, u) \in E\}$, and the degree of v in G is $d_v(G) = |N_v(G)|$. Similarity, we use $\overline{N}_v(G)$ and $\overline{d}_v(G)$ to denote the set of neighbors and the degree of v in $\overline{G}$ respectively. Given a vertex subset S of V, we let $G(S) = (S, E_S)$ be the subgraph of G induced by S, where $E_S = \{(u, v) \in E | u, v \in C\}$. If the context is clear, we abbreviate $N_v(G(S))$ to $N_v(S)$ and $d_v(G(S))$ to $d_v(S)$ respectively.

A vertex subset S of V is a clique if every pair of vertices is adjacent to each other (i.e., $(u, v) \in E$ for each $u, v \in S$). We refer to a clique S as a maximal clique if no vertex in $V \setminus S$ is adjacent to all vertices in S. The problem of finding all maximal cliques on a graph has been well studied [10, 19, 21, 38, 43, 47]. However, as we discussed in Section 1, the concept of clique may be overly restrictive for many real-world applications [40], thus relaxed clique models are often used as alternatives in practice [6, 7, 17, 34, 50]. In this paper, we focus mainly on a notable relaxed clique model, called s-defective clique, which was first proposed in [50].

Definition 1 (Maximal s-defective clique). Given a graph G and an integer s, a maximal s-defective clique S is a subset of vertices of G if (1) the subgraph $G(S)$ induced by S contains at least $\binom{|S|}{2} - s$ edges, and (2) there does not exist any vertex subset satisfying (1) and containing S.

Clearly, an s-defective clique is a clique when $s = 0$, which indicates that the clique is a special case of the s-defective clique. We then show that the s-defective clique has the following two useful properties, which will be used to devise our enumeration algorithms. Note that due to the space limit, all detailed proofs are omitted in this paper. We provide a full version with all proofs in [18].

Property 1 (Hereditary). Given an arbitrary s-defective clique S of G, for each subset H of S, H is also an s-defective clique of G.

By Property 1, it is easy to check that an s-defective clique S is maximal in G if there is no vertex in $V \setminus S$ that can be added to S.

Property 2. Given an s-defective clique S of G, the diameter of $G(S)$ is at most 2 if $|S| \geq s + 2$.

By Property 2, each s-defective clique with size no less than $s + 2$ must be densely-connected, since the diameter of the subgraph is no larger than 2. Thus, in real-world applications, we are often interested in enumerating all maximal s-defective cliques with size q no less than $s + 2$. To obtain a good relaxation of a clique, the parameter s in the s-defective clique model is generally very small (e.g., $s \leq 5$), so that the size constraint $q \geq s + 2$ is easy to meet for relatively-large maximal s-defective cliques. In this paper, we also investigate the problem of enumerating all relatively-large maximal s-defective cliques. Below, we formally define our problems.

Problem definition. Given an undirected graph G and a parameter $s \geq 1$, the goal of this paper is to enumerate: (1) all maximal s-defective cliques, and (2) all maximal s-defective cliques with size q no less than $s + 2$ (i.e., $q \geq s + 2$).

It has been known that finding the maximum s-defective clique is NP-hard [13, 48], thus the problem of enumerating all maximal s-defective cliques is also NP-hard. Therefore, there does not exist a polynomial-time algorithm to solve our problems unless NP = P.

Can existing solutions be used to solve our problems? Note that many advanced techniques have been developed to solve the problems of enumerating maximal cliques [10, 19, 21, 38, 43, 47] and maximal k-plexes [7, 17, 44, 49, 54], which are closely related to our problem. Unfortunately, we show that none of them can be used to solve our problem. The detailed explanations are given as follows.

The notable solutions for enumerating maximal cliques are the classic Bron-Kerbosch (BK) algorithm [10] and its variants [21, 38, 47] which use a pivoting technique to reduce recursive calls. The idea of such a pivoting technique is that given a recursion with S and C, if a vertex $v \in C$ is selected as a pivot, then only vertices in $C \setminus N_v(G)$ are used to expand S, where S and C are the current partial clique and the candidate set used to expand S, respectively. Thus, many branches can be pruned by such a pivoting approach. However, such an idea cannot be used to enumerate maximal s-defective cliques, since for a vertex $v \in C$, some neighbor vertices of v in C probably form a maximal s-defective clique with S. For instance, given a graph shown in Fig. 1(a) and a recursive call with $S = \{v_6\}$ and $C = \{v_1, v_2, v_3, v_4, v_5\}$, then we can see that $\{v_4, v_5, v_6\}$ is a maximal 1-defective clique. If v_2 is selected as a pivot, the result $\{v_4, v_5, v_6\}$ can be ignored by such a pivoting technique, because $\{v_4, v_5\} \subset N_{v_2}(G)$. Since $(v_4, v_5) \in E$, it is also difficult to design algorithms by checking s missing edges in $G(\{v_4, v_5\})$ to identify whether v_4 and v_5 should expand $\{v_6\}$. Thus, we cannot use this pivoting technique to solve our problems.

In maximal k-plex enumeration, the state-of-the-art approaches are developed in [49, 54], which pick the vertex $v \in C$ that has the smallest degree in $G(S \cup C)$ to perform the branch-and-bound enumeration procedure. This idea is based on the observation that if the smallest degree of vertices in $G(S \cup C)$ is no less than $|S \cup C| - k$, then the current set $S \cup C$ is exactly a k-plex. Therefore, no further sub-recursive calls are needed, and thus the current recursive call can be terminated. However, for maximal s-defective clique enumeration, there is no similar result. The only available result is that if the smallest degree of vertices in $G(S \cup C)$ is no less than $|S \cup C| - 1$, then $S \cup C$ is an s-defective clique. However, in the recursive calls of enumerating maximal s-defective cliques, the subgraph $G(S \cup C)$ is often difficult to meet this property. This is because for any maximal s-defective clique S ($s \geq 1$), the smallest degree of vertices in $G(S)$ is probably $|S| - 1 - s$. Thus, the techniques developed in [49, 54] are also not applicable to enumerate all maximal s-defective cliques.

Based on the above analyses, new approaches need to be developed to efficiently solve the maximal s-defective clique enumeration problems. In the following sections, we will present two different types of algorithms to address the issues.

3 A POLYNOMIAL-DELAY ALGORITHM

In this section, we develop an output-sensitive algorithm to enumerate all maximal s-defective cliques whose time complexity mainly relies to the number of results. A nice feature of this algorithm is that it can output two consecutive maximal s-defective cliques within polynomial time. Such an algorithm is inspired by the classic reverse search technique [2], which was originally developed for enumerating vertices in polyhedra. Below, we first briefly describe the reverse search technique and then present our solutions.

3.1 A Brief Overview of Reverse Search

To make use of the reverse search technique for enumeration, a graph-like structure which captures the relationships of all enumeration results must be defined first, which is shown in Definition 2.

Definition 2 ([2]). Given a graph G, let C be the set of all objects of G that needs to be output. Then, the graph-like structure is defined as follows:

- *Root: a unique root object in G can be found in polynomial time.*
- *Neighbors: the set of neighbors of an object $S \in C$, denoted by $\Gamma(S)$, i.e., it is possible for S to jump to each object in $\Gamma(S)$.*
- *Parent: projection from a non-root object S to its unique neighbor by a local search function $f(S)$, which satisfies that the root object can be found by a finite number of local search functions from S, i.e., $f^k(S) = f(f(...f(S))) = root$, where k is a finite positive integer.*

Based on Definition 2, it is easy to see that all objects in C form a connected graph, revealing which objects are allowed to jump from one to another. Then, with the parent relationship and the local search function, any non-root object in C has a unique and acyclic path from it to the root object, which forms a spanning tree. Thus, all objects can be output by a depth-first search (DFS) starting from the root object in the graph. Such an enumeration technique is called reverse search which is detailed in Algorithm 1. As shown in [2], Algorithm 1 has several interesting theoretical properties.

Theorem 3.1. *Let $t(f)$ and $t(\Gamma)$ be the time used to compute f and Γ. Then, the time complexity of Algorithm 1 is $O(K\omega(t(\Gamma) + t(f)))$, where $K = |C|$ and $\omega = max_{S \in C}|\Gamma(S)|$.*

Theorem 3.2. *Algorithm 1 is polynomial delay if both f and Γ can be computed within polynomial time.*

3.2 The Proposed Reverse Search Algorithm

As shown in Algorithm 1, the framework of the reverse search technique is quite implicit; and the definition of a suitable graph-like structure for different problems is also quite challenging. In the following, we propose a novel graph-like structure to characterize the relationships among all maximal s-defective cliques of a graph G, based on which a new enumeration algorithm is proposed.

Devising the graph-like structure. Given a set S of vertices, we denote by $S_{<v}$ the subset of vertices in S that are smaller than v, i.e., $S_{<v} = \{u \in S | u < v\}$. Similarly, $S_{\leq v}$ is defined as $S_{\leq v} = \{u \in S | u \leq v\}$. For two vertex sets S_1 and S_2 of G, we call $S_1 < S_2$ if S_1 is lexicographically smaller than S_2. Then, given a set S of G, we use $Extend(S)$ to denote the lexicographically smallest maximal s-defective clique that contains S. For instance, given a graph shown in Fig. 1(a) and $s = 1$, suppose that $S = \{v_2, v_5\}$, we have $Extend(S) = \{v_1, v_2, v_3, v_5\}$. Based on these notations, we define the graph-like structure for enumerating all maximal s-defective cliques as follows.

Algorithm 1: The reverse search framework [2].

Input: The graph G.
Output: All objects C of G.
Let S_0 be a root object in G;
$ReverseSearch(S_0)$;
Function: $ReverseSearch(S)$
  Output S as a solution;
  foreach $S' \in \Gamma(S)$ **do**
    if $f(S') = S$ **then** $ReverseSearch(S')$;

Definition 3 (Root). Given a graph G, we define the root node of G as the maximal s-defective clique which is the lexicographically smallest among all maximal s-defective cliques, i.e., $root = Extend(\{v_1\})$.

Definition 4 (Neighbors). Let $\mathcal{C}$ be the set of all maximal s-defective cliques of G. For a maximal s-defective clique S, its neighbor set is defined as $\Gamma(S) = \{S' \in \mathcal{C} | S' \cap S \neq \emptyset\}$.

Given a maximal s-defective clique S of G, we refer to a vertex $v \in S$ as the *critical vertex*, denoted by π_S, if v is smallest in S such that $Extend(S_{\leq v}) = S$, i.e., $\pi_S = argmin_{v \in S}\{Extend(S_{\leq v}) = S\}$. Then, we define the parent of maximal s-defective clique S in G.

Definition 5 (Parent). Given a graph G and two maximal s-defective cliques A and B, we call A the parent of B if $A \in \Gamma(B)$ and $A = Extend(B_{<v_p})$, where $v_p = \pi_B$.

Example 1. Given a graph G shown in Fig. 1(a), let $S_1 = \{v_1, v_2, v_4, v_5\}$ and $S_2 = \{v_2, v_3, v_4, v_5\}$ be the two maximal 1-defective cliques of G. For a maximal 1-defective clique $S_3 = \{v_4, v_5, v_6\}$, we can see that the parent of S_3 is S_1 instead of S_2. Although both S_1 and S_2 are the neighbors of S_3, the critical vertex π_{S_3} of S_3 is v_6 and we have $Extend(S_{3<v_6}) = Extend(\{v_4, v_5\}) = \{v_1, v_2, v_4, v_5\} = S_1$. Thus, only S_1 can be the parent of S_3.

Determining all neighbors. Although the graph-like structure of maximal s-defective cliques of G has been established, it is still unclear how to compute all neighbors of a maximal s-defective clique. Our solution to tackle this issue is based on the following observation. Note that in Algorithm 1, only the child nodes of a maximal s-defective clique S (here the child nodes represent the maximal s-defective cliques whose parent is S) have an opportunity to enter into the next recursion (which are what we exactly want), while the other neighbors can be ignored. Thus, we only need to find all possible child nodes of a maximal s-defective clique instead of all neighbor nodes.

By Definition 5, for the parent node A of a maximal s-defective clique S, we have the following two relationships: 1) $\pi_S \in S \setminus A$ and 2) $S_{<\pi_S} \subset A$. This means that, for any two maximal s-defective cliques A and B adjacent to each other, if they do not satisfy these two relationships, A (or B) must not be the child node of B (or A). For example, consider a maximal 1-defective clique $S_1 = \{v_1, v_2, v_4, v_5\}$ shown in Fig. 1(a), and a neighbor node $S_2 = \{v_2, v_3, v_4, v_5\}$ of S_1. We can see that $\pi_{S_2} = v_4$ and $S_{2<v_4} = \{v_2, v_3\} \not\subset S_1$, Then, S_2 is not the child node of S_1 and such a node can be omitted if S_1 being processed.

As a consequence, all possible child nodes of the maximal s-defective clique can be computed by the following approach. Given a parent node S, for each vertex $u \notin S$ of G, we first find each possible subset $S' \subseteq S_{<u}$ satisfying that $S' \cup \{u\}$ is an s-defective clique. Then, we use the $Extend$ function to guarantee the maximality so that the possible child nodes of S containing $S' \cup \{u\}$ can be obtained. Interestingly, we also notice that $S' \cup \{u\}$ can be maximal in $G(S_{<u} \cup \{u\})$, because

Algorithm 2: $GenerateNbrs(S, u)$

```
1  N_S ← ∅; X ← S \ N_u(G); P ← S ∩ N_u(G);
2  for each I ⊆ X s.t. |I| ≤ s do
3      S' ← I ∪ P ∪ {u};
4      if S' is the maximal in G(S ∪ {u}) then
5          Add S' into N_S;
6      else if the missing edges in G(S') is larger than s then
7          Let s' be the number of missing edges in G(S') (s' ≤ s + |I|);
8          for each I' ⊆ P s.t. |I'| ≤ s' − s do
9              S'' ← S' \ I';
10             if S'' is maximal in G(S ∪ {u}) then
11                 add S'' into N_S;
12 return N_S;
```

for any child node B of S with $\pi_B = u$, it always satisfies that $B_{<\pi_B} \subset S$ and $B_{<\pi_B}$ is maximal in $G(V_{<\pi_B})$. The detailed procedure is shown in Algorithm 2.

In Algorithm 2, it admits two parameters $S_{<u}$ and u, where S is a maximal s-defective clique and u is a vertex of G such that $u \notin S$ (here we assume that all vertices in S are less than u). Then, the algorithm first initializes three sets $\mathcal{N}_S$, X, and P (line 1), where X and P are subset of S in which each vertex is non-adjacent and adjacent to u, respectively. Note that the set $P \cup \{u\}$ must be an s-defective clique in $G(S \cup \{u\})$ since all vertices in P are adjacent to u and P is an s-defective clique. This means that a maximal s-defective clique in $G(S \cup \{u\})$ must contain no vertex or at most s vertices in X. Therefore, the algorithm recursively selects a subset I from X such that the size of I does not exceed s to detect whether $S' = I \cup P \cup \{u\}$ is a maximal s-defective clique in $G(S \cup \{u\})$ (lines 2-11). If not, we need to remove a subset I' in P from S' to eliminate conflicts (lines 6-10). Here the size of I' is no larger than $|I|$ since the missing edges in $G(S')$ is no larger than $s + |I|$.

The overall algorithm. Equipping with the graph-like structure and the technique for computing all possible child nodes of an s-defective clique, we can devise the reverse search algorithm for enumerating all maximal s-defective cliques. The detailed algorithm is outlined in Algorithm 3. In Algorithm 3, it first labels each vertex in V with v_1 to v_n to ensure the lexicographical order of each maximal s-defective clique in G (line 1). Then, the algorithm determines the root node with Definition 3 (line 2) and invokes the DFS enumeration procedure *Enum* (line 3).

In *Enum*, it first determines an output order of maximal s-defective cliques (line 10 and line 17), which is determined by the depth of the current recursion (the main purpose of this trick is to reduce the output delay between consecutive solutions as used in [45]). Then, the procedure iteratively selects a vertex $u \in V$, satisfying that $u \notin S$ and $u > \pi_S$, to continue the computations (lines 11-16). Specifically, it first invokes Algorithm 2 to compute all possible child nodes of S (which may not be maximal) (line 12) and then calls the *Extend* function to maximize them (line 14). After that, the procedure checks each possible child node R whether it is really a child of S. If so, we continue the recursion with such a child node (line 16). Finally, The procedure terminates until all maximal s-defective cliques have been visited. An illustrative example for enumerating all maximal s-defective cliques of a graph G is shown in Fig. 2.

Example 2. Given a graph G shown in Fig. 1(a) and a parameter $s = 1$, the graph-like structure for each maximal 1-defective clique of G can be determined by definitions, which is shown in Fig. 2(a). The reverse search starts from the root node $S_1 = \{v_1, v_2, v_3, v_5\}$, and computes all possible child nodes of S_1

Algorithm 3: The proposed reverse search algorithm for enumerating all maximal s-defective cliques.

Input: The graph G and a parameter s.
Output: All maximal s-defective cliques of G.

```
Let v_1, ..., v_n be the vertices in V;
S ← Extend({v_1});
Enum(S, 0);
Function: Extend(S)
    for i = 1 to n do
        if S ∪ {v_i} is the s-defective clique s.t. v_i ∉ S then
            S ← S ∪ {v_i};
    return S;
Function: Enum(S, i)
    if i is odd then Output S;
    forall u in V \ S s.t. u > π_S do
        N_S ← GenerateNbrs(S_<u, u);
        for each S' ∈ N_S do
            R ← Extend(S');
            if R ≥ S' s.t. S = Extend(S'_<u) then
                Enum(R, i + 1);
    if i is even then Output S;
```

using Algorithm 2. For the first possible child node $S_2 = \{v_1, v_2, v_4, v_5\}$, it can be seen that the parent of S_2 is exactly S_1. Then, the algorithm continues the recursive call with S_2 and finds the possible child nodes of S_2. Since $\pi_{S_2} = v_4$, two possible child nodes $S_3 = \{v_1, v_2, v_5, v_6\}$ and $S_6 = \{v_4, v_5, v_6\}$ of S_2 are detected, where only the parent of S_6 is S_2. The algorithm further processes the recursive call with S_6. Since no possible child node of S_6 is found, the algorithm backtracks to S_1 and finds the second possible child node S_3. The next recursive call with S_3 is also processed since the parent of S_3 is S_1. Finally, the algorithm terminates until the remaining two child nodes $S_4 = \{v_2, v_3, v_4, v_5\}$ and $S_5 = \{v_3, v_5, v_6\}$ of S_1 are found. It can bee seen that the parent node of S_6 is not S_1, thus S_6 is ignored in the recursive call with S_1.

Below, we analyze the correctness and complexity of Algorithm 3 in the following theorems.

Theorem 3.3. *Algorithm 3 correctly outputs all maximal s-defective cliques of G.*

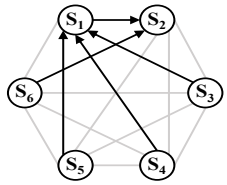
Theorem 3.4. *The time complexity of Algorithm 2 is $O(\frac{\triangle^{2s+2}}{4^s})$, where $\triangle$ is the size of the maximum s-defective clique of G.*

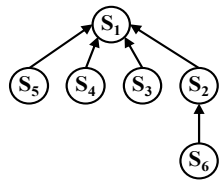
Theorem 3.5. *Algorithm 3 is a polynomial-delay algorithm with $O(n^2 \frac{\triangle^{2s+1}}{4^s})$ delay. The space complexity of Algorithm 3 is $O(n\triangle)$.*

Discussions. Note that Algorithm 3 cannot be extended to solve problem (2), i.e., enumerating maximal s-defective cliques with the size-constraint. This is because the reverse search algorithm needs to construct a graph-like structure (Definition 2) and make use of their parent relationships to guide the recursive calls. However, in the developed graph-like structure, there inevitably exist some maximal s-defective cliques whose parent is an s-defective clique with size less than q. If the

Result ID	Vertices
S_1	v_1, v_2, v_3, v_5
S_2	v_1, v_2, v_4, v_5
S_3	v_1, v_2, v_5, v_6
S_4	v_2, v_3, v_4, v_5
S_5	v_3, v_5, v_6
S_6	v_4, v_5, v_6

(a) The graph-like struture





(b) Enumeration tree

Fig. 2. A running example for enumerating all maximal 1-defective cliques using reverse search.

parents with the size less than q are ignored in the enumeration procedure, the connectivity of the graph-like structure will be broken. Therefore, all available results whose parents are ignored will be missed, which implies that Algorithm 3 cannot be used to solve problem (2). We will present new techniques to solve problem (2) in Section 4.

Since the notion of s-defective clique satisfies the hereditary property (Property 1), there are two reverse search frameworks [15, 16] available for solving our problem (1), i.e., the framework for enumerating maximal subgraphs satisfying the hereditary property [15] and the framework for enumerating maximal subgraphs satisfying strongly accessible properties [16]. However, the framework used in [15] needs to store all maximal results in the main memory to avoid redundant outputs, thus it usually cannot handle the graphs that have a large number of results. The framework in [16], also defines a general graph-like structure that is the same as Definition 2 to guide the recursive processes, which means that our proposed techniques can also be extended to the framework in [16] to solve the problem of enumerating maximal s-defective cliques. Since the graph-like structures are the same, the worst-case delay complexity of this framework is the same as that of Algorithm 3. As a consequence, it is sufficient to use the basic search framework [2] to solve our problem.

4 NEW PIVOT-BASED ALGORITHMS

Although the proposed reverse search algorithm has a nice theoretical property, its practical performance may be not very well, as each maximal s-defective clique has too many neighbors. In this section, we propose several branch-and-bound enumeration algorithms based on a novel pivoting technique. Interestingly, we show that the time complexity of all the proposed pivot-based enumeration algorithms can be bounded by $O(n\alpha_s^n)$, where α_s is a real number strictly less than 2. For instance, if $s = 0$, $s = 1$ and $s = 2$, we have $\alpha_0 = 1.618$, $\alpha_1 = 1.839$, and $\alpha_2 = 1.928$, respectively.

In the rest of this paper, we will focus more on problem (2) as defined in Section 2, i.e., enumerating all s-defective cliques with size q no less than $s + 2$. This is because small s-defective cliques are often of no practical use in real-world applications. Moreover, as shown in Section 2, the diameter of a maximal s-defective clique with size $q \geq s + 2$ is no larger than 2, thus it is more desirable to represent a densely-connected community. It is important to note that although we focus on problem (2), all the proposed branch-and-bound enumeration techniques can also be easily adapted to enumerate all maximal s-defective cliques (i.e., for problem (1)). Below, we first present a basic branch-and-bound enumeration algorithm, which forms a basis for developing our advanced techniques.

4.1 A Basic Branch-and-Bound Algorithm

The main idea of the basic branch-and-bound algorithm is that each s-defective clique either contains a particular vertex v in G or does not contain v. Thus, given a graph G, the original problem can be divided into two sub-problems. The first sub-problem is to enumerate all maximal s-defective cliques containing a vertex v in G; and another one is to enumerate all maximal s-defective cliques

Algorithm 4: The basic branch-and-bound algorithm.

Input: The graph G and two parameters s and $q \geq s + 2$.
Output: All relatively-large maximal s-defective cliques of G.

```
1  BranchEnum(∅, V, ∅);
2  Function: BranchEnum(S, C, X)
3      if C = ∅ then
4          if X = ∅ s.t. |S| ≥ q then Output S;
5          return;
6      Select a vertex v from C, and let S' ← S ∪ {v};
7      C' ← {u ∈ C \ {v} | S' ∪ {u} is an s-defective clique};
8      X' ← {u ∈ X | S' ∪ {u} is an s-defective clique};
9      BranchEnum(S', C', X');
10     BranchEnum(S, C \ {v}, X ∪ {v});
```

excluding v. Such a method can be recursively applied to enumerate all maximal s-defective cliques, and the pseudocode is shown in Algorithm 4.

In Algorithm 4, it invokes the *BranchEnum* procedure to enumerate all maximal s-defective cliques. This procedure admits three parameters: S, C, and X, where S is a partial s-defective clique, C is the candidate set in which each vertex is used to expand S, and X is the exclusion set containing all vertices that have been processed form C. Initially, the sets S and X are the empty sets while C is set to V. Then, in each recursive call, the algorithm first checks if S is maximal (lines 3-5). If not, it selects a vertex v from C to perform two sub-recursive calls. The first one is the enumeration of all maximal s-defective cliques containing $S \cup \{v\}$ (line 9); and the other one is invoked to enumerate all maximal s-defective cliques that contain S but not v (line 10). Note that before invoking the first recursive call, the sets C and X need to be updated to ensure that $S \cup \{v, u\}$ forms an s-defective for each u in C and X (lines 7-8). Whenever the set C is empty, the algorithm terminates.

It is easy to verify that Algorithm 4 can correctly enumerate all maximal s-defective cliques of G. The worst-case time complexity of Algorithm 4 is $O(n\triangle 2^n)$, which is analyzed in Theorem 4.1.

Theorem 4.1. *Algorithm 4 outputs all maximal s-defective cliques of G in $O(n\triangle 2^n)$ time.*

It can be seen that the basic algorithm is very inefficient because there are a large number of branches that produces non-maximal s-defective cliques. In the following, we will propose several efficient techniques to reduce the unnecessary computations.

4.2 A Novel Pivot-based Enumeration Algorithm

To speed up the basic branch and bound algorithm, the key point is to determine the enumeration branches that definitely produce non-maximal s-defective cliques. We observe that only the sub-recursive calls for enumerating all maximal s-defective cliques that exclude v may generate non-maximal s-defective cliques. The reason is as follows. Suppose that C is the candidate set and a maximal s-defective clique $A \subseteq C$ containing v is generated by the first sub-recursive call (i.e., the sub-recursion for enumerating all maximal s-defective cliques containing v). It is easy to see that the set $A \setminus \{v\} \subseteq C$ will be included in the candidate set of the second sub-recursive call (the sub-recursion for enumerating all maximal s-defective cliques excluding v). Clearly, a non-maximal s-defective clique $A \setminus \{v\}$ will be explored by the second sub-recursive call which results in unnecessary computations. To optimize the second sub-recursive call, we propose a novel pivoting technique which is described as follows.

The pivoting technique. The main idea of this technique is based on the fact that for any maximal s-defective clique B that does not contain v, there must be a vertex $u \in B$ such that $u \notin A$, where A is an arbitrary s-defective clique containing v. Therefore, given a candidate set $C \setminus \{v\}$ with $A \subseteq C$, if we select the vertex u ($u \notin A$) to expand the partial s-defective clique S in the sub-recursive call to enumerate all maximal s-defective cliques excluding v, then all s-defective cliques contained in $A \setminus \{v\}$ definitely cannot be generated by such a sub-recursive call. This is because any maximal s-defective clique generated by the sub-recursive call must contain a vertex u, and the set $A \setminus \{v\}$ does not contain the vertex u. Moreover, we further notice that any vertex in $A \setminus \{v\}$ is not necessary to be used to expand S (in the sub-recursive call that enumerates all maximal s-defective cliques excluding v). Since for any maximal s-defective clique in $C \setminus \{v\}$, there must be a vertex in $C \setminus A$ that can be used to expand to obtain it. This indicates that all maximal s-defective cliques that exclude v can be generated by only expanding the vertices in $C \setminus A$. Based on this idea, we develop a novel pivoting technique for enumerating all maximal s-defective cliques, which is shown in the following lemma.

Lemma 1. Given three sets S, C, and X in a recursive call, let v be a vertex in C with $S \subseteq N_v(G)$. If the first sub-recursive call is used to enumerate all maximal s-defective cliques containing v, then all vertices in $C \cap N_v(G)$ are no need to expand S in the other sub-recursive call.

We note that directly applying the pivoting technique in each recursive call may not significantly improve the efficiency. For example, let us reconsider the graph G shown in Fig. 1(a). Let $S = \emptyset$, $C = \{v_1, v_2, ..., v_6\}$, and $X = \emptyset$ be the three input sets of the recursive call to enumerate all maximal 1-defective cliques. If we select the vertex v_2 as the pivot vertex, the first sub-recursive call with $S' = \{v_2\}$ and $C' = \{v_1, v_3, ..., v_6\}$ is invoked to enumerate all maximal 1-defective clique containing v_2. Then, based on our pivoting technique (Lemma 1), the second sub-recursive call with $S = \emptyset$ and $C = \{v_1, v_3, ..., v_6\}$ only selects $v_6 \in C \setminus N_{v_2}(G) = \{v_6\}$ as the pivot vertex and then enumerates all maximal 1-defective cliques containing v_6. We observe that all maximal 1-defective cliques have been generated so far. However, if we further perform the pivoting technique, the vertices in $C \setminus N_{v_6}(G) = \{v_3, v_4\}$ will be used to expand $S = \emptyset$ in the sub-recursive call that is invoked with $S = \emptyset$ and $C = \{v_1, v_3, v_4, v_5\}$ to enumerate the maximal 1-defective cliques, which incurs redundant computations.

Pivot-based branching rule. Interestingly, we find that only a part of recursive calls in the branch-and-bound algorithm is required to perform the pivoting technique. Let the top recursive call be the root branch. For each recursive call, we refer to the sub-recursive call that enumerates maximal s-defective cliques containing a particular vertex in the candidate set as the first-child, and the other sub-recursive call as the second-child. Then, we propose the following *pivot-based branching rule*: in the branch-and-bound algorithm, only the root branch and the first-child of each recursive call require to perform the pivoting technique, while for the second-child of each recursive call, we inherit the pivoting results (i.e., the candidate vertices that can be used to expand) from its parent branch to continue the computation.

The pivot-based branching algorithm. With the proposed pivot-based branching rule, we then develop a novel pivot-based branch-and-bound algorithm. Unlike the basic branch-and-bound algorithm, we need to partition the candidate set into two disjoint sets C_1 and C_2. The benefits of doing this are that (i) when performing the pivot-based technique in a recursion, C_1 and C_2 are the candidate sets that need and do not need to be used to expand S, respectively; (ii) it can also be used to distinguish whether the current recursion is the first or second-child branch, i.e., if C_2 is not empty, it must be the second-child branch. The detailed pseudocode is shown in Algorithm 5.

In Algorithm 5, it invokes the *PivotEnum* procedure to perform recursive calls, which requires five parameters: S, r, C_1, C_2, and X, where the parameters C_1 and C_2 are the two disjoint candidate

Algorithm 5: The pivot-based branch-and-bound algorithm.

Input: The graph G and two parameters s and $q \geq s + 2$.
Output: All relatively-large maximal s-defective cliques of G.

```
1  PivotEnum(∅, 0, {(v, 0)|v ∈ V}, ∅, ∅);
2  Function: PivotEnum(S, r, C1, C2, X)
3      if C1 = ∅ then
4          if C2 ∪ X = ∅ and |S| ≥ q then Output S;
5          return;
6      Select a pivot element (v, cv) from C1 such that cv is maximum among all elements in C1;
7      if cv = 0 and C2 = ∅ then
8          C2 ← {(u, cu) ∈ C1|u ∈ Nv(G)};
9          C1 ← {(u, cu) ∈ C1|u ∉ Nv(G)};
10     C1' ← UpdateSet(S ∪ {v}, r + cv, v, C1 ∪ C2);
11     X' ← UpdateSet(S ∪ {v}, r + cv, v, X);
12     PivotEnum(S ∪ {v}, r + vc, C1', ∅, X');
13     PivotEnum(S, r, C1 \ {(v, cv)}, C2, X ∪ {(v, cv)});
14 Function: UpdateSet(S, r, v, C)
15     C' ← ∅;
16     foreach (u, cu) ∈ C do
17         if u ∉ Nv(G) then cu ← cu + 1;
18         if v ≠ u and r + cu ≤ s then
19             C' ← C' ∪ {(u, cu)};
20     return C';
```

sets respectively, and r is a parameter to record the number of missing edges in $G(S)$. Initially, the sets S, C_2, and X are set to empty, while C_1 contains all vertices in V (line 1). Note that each element of the three sets C_1, C_2, and X is a pair (v, c_v), where v is the vertex and c_v is the number of non-neighbors of v in S. Here the purpose of using such a *pair representation* is to improve the efficiency for updating the sets C_1, C_2, and X. Specifically, when a vertex v is added to S, for each (u, c_u) in C_1, C_2, and X, we can determine whether u can form an s-defective clique with $S \cup \{v\}$ by simply checking whether u and v are adjacent, since c_u has recorded the number of non-neighbors of u in S. Thus, the overall updating time is linear by using this trick, which is detailed in the procedure *UpdateSet* (lines 14-20).

In each recursive call, if $C_1 \cup C_2 \cup X$ is empty, the algorithm outputs S as a result (line 4). Otherwise, the algorithm recursively performs the branch-and-bound procedure if $C_1 \neq \emptyset$ (lines 6-13). More specifically, it first picks a pivot element (v, c_v) in C_1, where c_v is maximum among all elements in C_1 (line 6). If such a pivot vertex v satisfies $S \subseteq N_v(G)$ and also $C_2 = \emptyset$ (lines 7), the algorithm makes use of Lemma 1 to partition the original C_1 into two new candidate sets C_1 and C_2 such that all vertices in C_1 are the neighbors of v and C_2 does not contain any neighbors of v (lines 8-9). Otherwise, the original sets C_1 and C_2 are directly used. The algorithm then obtains the sets $C_1' \subset (C_1 \cup C_2)$ and $X' \subseteq X$ by invoking the *UpdateSet* procedure, and invokes the first sub-recursive call to expand S with v (lines 10-12). Note that in this sub-recursive call, all candidate vertices are included in C_1', while C_2' is empty. This is because only the second sub-recursive call may produce redundant computations. After that, the algorithm moves the pivot vertex (v, c_v) from C_1 to X and makes use of the second sub-recursive call to continue the recursions until the set C_1 is empty (line 3 and line 13). Below, we give an example to illustrate the basic idea of this algorithm.

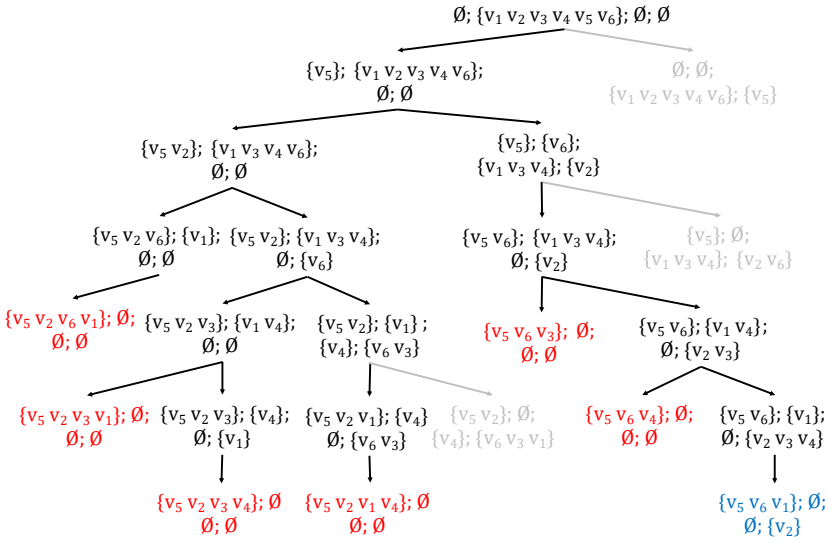


Fig. 3. The enumeration tree of our pivot-based enumeration algorithm with $s = 1$ (The sets used in each recursive call correspond to the partial 1-defective clique S, candidate set C_1, candidate set C_2, and exclusion set X from left to right).

Example 3. Given a graph G shown in Fig. 1(a) and a parameter $s = 1$. Algorithm 5 first initializes the sets S, C_2, and X to empty, and the set C_1 to V. In the root branch, we assume that v_5 is the pivot vertex. The algorithm then divides the candidate set into $C_1 = \emptyset$ and $C_2 = \{v_1, ..., v_4, v_6\}$, since all vertices in $V \setminus \{v_5\}$ are the neighbors of v_5. Thus, the second-child with $S = \emptyset$, $C_1 = \emptyset$, and $C_2 = \{v_1, ..., v_4, v_6\}$ will terminate immediately because $C_1 = \emptyset$. For the first-child with $S = \{v_5\}$ and $C_1 = \{v_1, ..., v_4, v_6\}$, another pivot vertex v_2 is further selected, which leads to that its candidate sets can be partitioned as $C_1 = \{v_6\}$ and $C_2 = \{v_1, v_3, v_4, v_5\}$. Subsequently, this recursion further generates two sub-recursive calls to continue the branch-and-bound procedure. For a new second-child, it can only expand $S = \{v_5\}$ with v_6, since C_1 contains only v_6. Such a second-child is finished until all maximal s-defective cliques containing $\{v_5, v_6\}$ are obtained, and the whole algorithm terminates when all vertices in C_1 for each recursion are computed. The complete enumeration tree of Algorithm 5 is shown in Fig. 3.

We analyze the correctness and complexity of Algorithm 5 in the following theorems.

Theorem 4.2. *Algorithm 5 outputs all maximal s-defective cliques of G exactly once and no non-maximal s-defective clique is output.*

Proof sketch. By Lemma 1, we can derive that all maximal s-defective cliques are output by Algorithm 5. Then, when all maximal s-defective cliques containing $v \in C_1$ are identified, v needs to be moved to X, thus we can avoid non-maximal and redundant results. This is because the current S is a possible result only if $X = \emptyset$. □

Theorem 4.3. *Given a graph G with n vertices, the time complexity of Algorithm 5 is bounded by $O(n\alpha_s^n)$, where α_s is the largest real root of function $x^{s+3} - 2x^{s+2} + 1 = 0$ and it is strictly smaller than 2. In particular, when $s = 0$, 1, and 2, we have $\alpha_s = 1.618, 1.839$, and 1.928, respectively.*

Proof sketch. Denote by $T(n)$ the total number of recursive calls. By Lemma 1, we can divide the set $C = C_1 \cup C_2$ into $C_1 = C \setminus N_v(G)$ and $C_2 = C \cap N_v(G)$ when the vertex $v \in C$ is the pivot.

If $|C_1| \leq s$, we can obtain a recurrence of $T(n) \leq \sum_{i=1}^{s+1} T(n-i)$. Otherwise, $T(n) \leq \sum_{i=1}^{s+2} T(n-i)$ is obtained, since the current S admits at most s missing edges. When the vertex $v \in C$ is not the pivot vertex, we have $S \setminus N_v(G) \neq \emptyset$, which also can derive that $T(n) \leq \sum_{i=1}^{s+2} T(n-i)$ based on a restriction that there are at most s missing edges in S. Thus, the worst-case recurrence is $T(n) \leq \sum_{i=1}^{s+2} T(n-i)$. By the result in [22], we can derive the results as shown in the theorem. □

Theorem 4.4. *The space complexity of Algorithm 5 is $O(n\triangle)$, where $\triangle$ is the size of the maximum s-defective clique of G.*

Remark. Note that based on Lemma 1, only the vertex $v \in C_1$ satisfying $S \subseteq N_v(G)$ is considered as the pivot vertex. If the conditions in line 7 of Algorithm 5 hold in a recursive call, then every vertex v in C_1 satisfies $S \subseteq N_v(G)$ and can be used as a pivot vertex. To achieve good efficiency, we can re-select a vertex that has the most number of neighbors in the candidate set C_1 as the pivot vertex.

Discussions. Note that the pivot-based enumeration techniques used in Algorithm 5 are different from the approaches for enumerating cliques, k-plexes, and the maximum s-defective cliques.

As we analyzed before, existing clique enumeration techniques [10, 21, 38, 47] and k-plex enumeration techniques [49, 54] cannot be used to solve our problem. Thus, in this paper, we present a new pivoting technique (Lemma 1). In the recursive enumeration procedure, we also develop a new binary branch-and-bound framework, which is different from the existing methods based on the set enumeration framework [10, 21, 38, 47].

Although the algorithm for enumerating all k-plexes in [54] achieves a similar time complexity to ours (Algorithm 5), the techniques used are fundamentally different. In this paper, we present a new pivoting technique to partition the candidate set into two disjoint subsets C_1 and C_2, and only select the vertices in C_1 to expand S. By Theorem 4.3, our approach can achieve a recurrence of $T(n) \leq \sum_{i=1}^{s+2} T(n-i)$. However, for maximal k-plex enumeration, [54] picks the vertex $v \in C$ with the smallest degree in $G(S \cup C)$ to generate sub-branches, thus obtaining a recurrence of $T(n) \leq \sum_{i=1}^{k+1} T(n-i)$ as shown in [54]. Since $k \geq 1$ and $s \geq 0$, the branching factor for these two approaches happens to be the same. Moreover, if using an adaptation of the method in [54] to solve our problem, the worst-case time complexity of this adapted approach cannot reach our bound. This is because the recursive call cannot terminate if the smallest degree in $G(S \cup C)$ is no less than $|S \cup C| - 1 - s$.

In addition, we also note that the algorithm proposed in [13] for finding the maximum s-defective clique is closely related to the techniques in [54], thus it is also different from ours. Specifically, in each recursion, [13] first determines whether $|C \setminus N_S(G)| > s$ holds, where $N_S(G)$ is the common neighbors of vertices in S, i.e., $N_S(G) = \{u \in C | S \subseteq N_u(G)\}$. If $|C \setminus N_S(G)| > s$, the algorithm uses the branching rule in [54] for enumeration. Otherwise, the algorithm makes use of the basic binary branching rule for enumeration (similar to Algorithm 4). As shown in [13], the time complexity of this approach is much worse than ours. Thus, using the technique in [13] to solve our problem cannot achieve good performance, which is also confirmed in our experiments.

4.3 Optimization Techniques

In this subsection, we propose several optimization techniques to further improve the efficiency of the proposed pivot-based algorithm. The first optimization is a degeneracy ordering for enumerating all maximal s-defective cliques. With the help of this technique, the time complexity of the proposed pivot-based algorithm can be further improved to $O(n^{s+2}\alpha_s^{\delta})$. Here δ is the degeneracy of a given graph G [32], which is often very small in real-world graphs [21, 31]. The other optimizations are

Algorithm 6: The improved pivot-based algorithm.

Input: The graph G and two parameters s and $q \geq s + 2$.
Output: All relatively-large maximal s-defective cliques of G.

1 Let $v_1, v_2, ..., v_n$ be the degeneracy ordering of vertices in V;
2 **for** $i = 1$ *to* n **do**
3 $C \leftarrow V_{>v_i} \cap N_{v_i}(G)$; $\overline{C} \leftarrow V_{>v_i} \setminus N_{v_i}(G)$; $X \leftarrow V_{<v_i}$;
4 **foreach** $I \subseteq \overline{C}$ *s.t.* $|I| \leq s$ **do**
5 $I \leftarrow I \cup \{v_i\}$; $s' \leftarrow$ the number of missing edges in $G(I)$;
6 **if** $s' \leq s$ **then**
7 Obtain $C' \subseteq \{(u, c_u) | u \in C\}$ and $X' \subseteq \{(u, c_u) | u \in \overline{C} \cup X \setminus I\}$ such that for each $(u, c_u) \in C' \cup X'$, $c_u = |I \setminus N_u(G)|$ and $s' + c_u \leq s$;
8 $PivotEnum(I, s', C', \emptyset, X')$;

aimed to reduce the size of the candidate set in enumerating all relatively-large maximal s-defective cliques. Below, we first introduce the concept of the degeneracy ordering.

Definition 6 (Degeneracy ordering). Given a graph G, the degeneracy ordering of vertices in V is an ordering of $v_1, v_2, ..., v_n$ such that for each v_i, its degree is minimum in $G(\{v_i, ..., v_n\})$.

To obtain the degeneracy ordering of the vertices in a graph G, we can use a peeling technique to iteratively remove the lowest-degree vertex from G [4]. Such a vertex-removal ordering is the degeneracy ordering, which can be computed in $O(m + n)$ time [4].

The key idea of our improved pivot-based algorithm is based on an observation that given a vertex $v \in V$, each maximal s-defective clique including v contains at most s non-neighbors of v. This indicates that to obtain all maximal s-defective cliques including v, it only needs to enumerate all maximal s-defective cliques containing $I \cup \{v\}$, for each I satisfying $|I| \leq s$ and $I \subseteq \overline{N}_v(G)$. Here the candidate set of $I \cup \{v\}$ must be a subset included in $N_v(G)$. Thus, we obtain an improved algorithm which is shown in Algorithm 6.

In Algorithm 6, it first computes the degeneracy ordering of the vertices in V (line 1). Then, for each vertex v_i in V, it computes all maximal s-defective cliques containing v_i (lines 2-8). Before performing recursive calls, the algorithm partitions the task into several sub-tasks based on our previous observation (lines 4-7). In particular, the algorithm first divides the candidate set into two disjoint sets C and $\overline{C}$, where C ($\overline{C}$) includes all neighbors (non-neighbors) of v_i that occur after v_i in the degeneracy ordering (line 3). In line 3, the set $V_{>v_i}$ is defined as $\{v_j \in V | j > i\}$. Next, it selects each subset I of $\overline{C}$ with $|I| \leq s$ and enumerates all maximal s-defective cliques that contain $I \cup \{v_i\}$ (lines 4-8). Note that the candidate set of the recursion with $I \cup \{v_i\}$ only contains the neighbors of v_i, since for any maximal s-defective clique S containing v_i, there always exists a recursion with $I \cup \{v_i\}$ satisfying $I \subset \overline{C}$ and $I = S \setminus N_{v_i}(G)$. Finally, the algorithm terminates when all vertices in V have been processed.

Theorem 4.5. *The time and space complexity of Algorithm 6 are $O(n^{s+2}\alpha_s^{\delta})$ and $O(n * min(\triangle, \delta))$, respectively, where α_s is the largest real root of $x^{s+3} - 2x^{s+2} + 1 = 0$ which is strictly smaller than 2, and δ is the degeneracy of the graph G.*

Optimization techniques for problem (2). Here we further propose several additional optimization techniques for enumerating all maximal s-defective cliques with size no less than q ($q \geq s + 2$).

Algorithm 7: The general pivot-based algorithm.

Input: A graph G.
Output: All maximal $\mathcal{P}$-subgraphs of G.

```
GenEnum(∅, V, ∅, ∅);
Function: GenEnum(S, C1, C2, X)
    if C1 = ∅ then
        if C2 ∪ X = ∅ then Output S;
        return;
    Select a pivot vertex v from C1; C ← C1 ∪ C2 \ {v};
    Obtain the sets C' ⊆ C and X' ⊆ X such that for each u ∈ C' ∪ X', S ∪ {v, u} is a P-subgraph;
    GenEnum(S ∪ {v}, C', ∅, X');
    if C2 = ∅ then
        Partition C into two disjoint subsets C'1 and C'2 such that all vertices in C'2 are ignored when
          expanding S; C1 ← C'1, C2 ← C'2;
    GenEnum(S, C1 \ {v}, C2, X ∪ {v});
```

Lemma 2. Given any s-defective clique S with the size of q ($q = |S|$), for each pair of vertices u and v in S, we have $|N_v(S) \cap N_u(S)| \geq q - s - 2$ if $(u, v) \in E$ and $|N_v(S) \cap N_u(S)| \geq q - s - 1$ if $(u, v) \notin E$.

Before performing the recursive procedure to enumerate the maximal s-defective cliques with size no less than q ($q \geq s + 2$), we can make use of Lemma 2 to remove unnecessary vertices in the candidate set. The details are as follows. Let C be the candidate set and $S = \{v\}$ be the partial result, we first partition the set C into two subsets C_1 and C_2 such that $C_1 = C \cap N_v(G)$ and $C_2 = C \setminus N_v(G)$. Then, we iteratively remove the vertices in C_1 whose degree in $G(C_1)$ is less than $q - s - 2$. Let $C_1' \subseteq C_1$ be the remaining vertices satisfying $d_u(C_1') \geq q - s - 2$ for each u in C_1'. We next remove each vertex u in C_2 that satisfies $|N_u(G) \cap C_1'| < q - s - 1$. After that, all the remaining vertices form the new candidate set of a recursive call. Clearly, such a pruning process can be done in linear time with respect to the number of edges in $G(C)$.

Interestingly, we can derive a more general result based on Lemma 2 as shown in the following lemma.

Lemma 3. Given any s-defective clique S with the size of q ($q = |S|$), for each subset $R \subseteq S$, we refer to $N_R(S)$ as the number of common neighbors of R in S, then we have $|N_R(S)| \geq q - |R| - s + r$, where r is the number of missing edges in $G(R)$.

Based on Lemma 3, an early termination criteria can be further used in each recursive call of our pivot-based algorithm. Specifically, in each recursive call, there always exists a vertex v in the candidate set C_1 selected to expand the partial s-defective clique S. Before expanding S with v, we first compute the common neighbors of $S \cup \{v\}$ in the candidate set. Then, we make use of Lemma 3 to determine whether the sub-recursive call with $S \cup \{v\}$ needs to be computed. If not, we can early terminate the computation. Note that in Algorithm 5 and Algorithm 6, we have already known the common neighbors of $S \cup \{v\}$ using the *UpdateSet* procedure. Therefore, such an early termination trick can be implemented in constant time in each recursive call.

5 A GENERAL PIVOTING PARADIGM

In this section, we generalize our pivot-based algorithm to enumerate all maximal subgraphs that satisfy the hereditary property. Given a graph property $\mathcal{P}$, we call the graph property $\mathcal{P}$ to be hereditary on the subgraph $G(S)$ if both $G(S)$ and all subgraphs of $G(S)$ satisfy property $\mathcal{P}$. For

convenience, we refer to a subgraph as a $\mathcal{P}$-subgraph if it satisfies the hereditary property $\mathcal{P}$. Clearly, the basic branch-and-bound algorithm can enumerate all maximal $\mathcal{P}$-subgraphs of G. Such an algorithm, however, is inefficient because its worst-case time complexity is $O(f(n)2^n)$, where $f(n)$ is a polynomial function with respect to (w.r.t.) n. To improve the efficiency, we present a general pivot-based branch-and-bound algorithm which is shown in Algorithm 7.

Similar to our pivot-based algorithm for enumerating all maximal s-defective cliques, Algorithm 7 also needs two disjoint candidate sets C_1 and C_2, and it only expands the partial $\mathcal{P}$-subgraph S using the vertices in C_1 (line 1 and line 6). Specifically, in each recursion, the algorithm selects a pivot v from C_1 to expand S, and then recursively enumerates all maximal $\mathcal{P}$-subgraphs that contains v (lines 7-8). To enumerate all the maximal $\mathcal{P}$-subgraphs excluding v, the algorithm first checks whether $C_2 = \emptyset$. If so, the algorithm partitions the current candidate set $C = C_1 \cup C_2 \setminus \{v\}$ into two new candidate sets C_1' and C_2' such that each vertex in C_2' is no need to expand S (lines 9-10). After that, the algorithm invokes the recursive procedure to enumerate all maximal $\mathcal{P}$-subgraphs excluding v (line 11).

The remaining technical issue needed to be addressed in Algorithm 7 is how to partition the candidate set C into two disjoint subsets C_1' and C_2' in line 10. Below, we propose a general partition approach to solve this issue.

Lemma 4. In each recursion, we let S be the partial $\mathcal{P}$-subgraph and C be the candidate set. Suppose that A is an arbitrary maximal $\mathcal{P}$-subgraph containing $S \cup \{v\}$ that has already been identified by the algorithm, where $A \subseteq C \cup S \cup \{v\}$. Then, based on A, we can obtain a valid partition such that $C_2 = A \setminus (S \cup \{v\})$ and $C_1 = C \setminus C_2$.

By Lemma 4, it is easy to construct a partition if we have a maximal $\mathcal{P}$-subgraph containing $S \cup \{v\}$. So, the question is how to obtain such a subgraph in each recursion? In effect, Algorithm 7 can obtain such a subgraph *for free*. That is, the results returned by the recursive call in line 8 are exactly the maximal $\mathcal{P}$-subgraphs that contain $S \cup \{v\}$. More interestingly, we prove that the time complexity of such a general pivot-based algorithm can be bounded by $O(f(n)\alpha^n)$, where α is a real number no larger than 2.

Theorem 5.1. *Let p be the maximum size of C_1 in line 11 of Algorithm 7. Then, the time complexity of Algorithm 7 is bounded by $O(f(n)\alpha^n)$, where $f(n)$ is a polynomial function w.r.t. n, and α is the maximum real root of $x^{p+2} - 2x^{p+1} + 1 = 0$, which is no larger than 2.*

6 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the efficiency and effectiveness of the proposed algorithms. Below, we first describe the experimental setup and then report our results.

6.1 Experimental Setup

Algorithms. We implement five algorithms, denoted by RSA, Basic, MDEC, Pivot, and Pivot+, to enumerate all maximal s-defective cliques, where RSA is the polynomial-delay algorithm shown in Algorithm 3, Basic is the basic branch-and-bound algorithm presented in Algorithm 4, MDEC is an adaptation of the maximum s-defective clique detection algorithm in [13] by removing all maximum s-defective clique pruning rules and remaining the branching technique, Pivot is the proposed pivot-based branch-and-bound algorithm presented in Algorithm 5, and Pivot+ is the improved Pivot algorithm shown in Algorithm 6. Here Basic and MDEC are used as baselines for comparison.

To enumerating relatively-large maximal s-defective cliques (with size no less than a given threshold q), we also implement two additional algorithms, called Pivot2 and Pivot2+. Pivot2 is a variant of Pivot which initializes the candidate set with 2-hop neighbors of a vertex v to enumerate

Table 1. Real-world graph datasets.

Datasets	n	m	$d_{\max}$	δ
scc-enron	146	9,828	145	119
scc-rt-lolgop	273	4,510	249	41
web-google	1,299	2,773	59	17
web-edu	3,031	6,474	104	29
ca-GrQc	4,158	13,422	81	43
web-spam	4,767	37,375	477	35
dblp-2010	226,413	1,432,920	238	74
ca-citeseer	227,320	1,628,268	1,372	86
wikipedia2009	1,864,433	9,014,630	2,624	66
flixster	2,523,386	15,837,602	1,474	68
socfb-B-anon	2,937,612	41,919,708	4,356	63

Table 2. Running time of various algorithms on small real-world graphs (in seconds).

Datasets	s	#Nums	RSA	Basic	MDEC	Pivot	Pivot+	s	#Nums	RSA	Basic	MDEC	Pivot	Pivot+
scc-enron	1	7.97×10^2	0.08	INF	INF	**0.01**	0.47	3	6.14×10^5	58.92	INF	INF	**0.22**	98.71
	2	2.36×10^4	2.09	INF	INF	**0.04**	7.52	4	1.46×10^7	1604.38	INF	INF	**2.18**	1062.73
scc-rt-lolgop	1	3.27×10^4	1.19	INF	INF	**0.01**	0.08	3	1.06×10^7	513.2	INF	INF	**1.32**	12.84
	2	6.77×10^5	25.98	INF	INF	**0.08**	1.19	4	9.29×10^7	4580.25	INF	INF	**10.09**	98.33
web-google	1	8.41×10^5	77.24	3.49	3.11	0.69	**0.37**	3	3.68×10^8	INF	2777.89	1350.03	305.72	**96.33**
	2	3.56×10^6	266.03	10.57	7.51	2.84	**1.72**	4	3.84×10^8	INF	3772.05	1392.68	**340.16**	411.99
web-edu	1	4.59×10^6	1101.43	178.63	423.71	9.00	**3.25**	3	4.65×10^9	INF	67828.74	41373.21	9404.94	**2397.92**
	2	1.94×10^7	3079.57	241.86	436.84	36.63	**19.42**	4	4.78×10^9	INF	INF	41860.55	**9445.79**	11414.93
ca-GrQc	1	8.63×10^6	2832.71	INF	INF	23.53	**7.86**	3	1.21×10^{10}	INF	INF	INF	33316.87	**8452.04**
	2	5.55×10^7	35055.38	INF	INF	96.18	**50.52**	4	1.35×10^{10}	INF	INF	INF	**33560.66**	40650.56
web-spam	1	1.26×10^7	5164.93	98.96	248.15	35.16	**13.55**	3	1.86×10^{10}	INF	INF	INF	57014.37	**15403.21**
	2	1.84×10^8	INF	557.83	812.66	165.25	**108.17**	4	2.71×10^{10}	INF	INF	INF	**62245.90**	75138.01

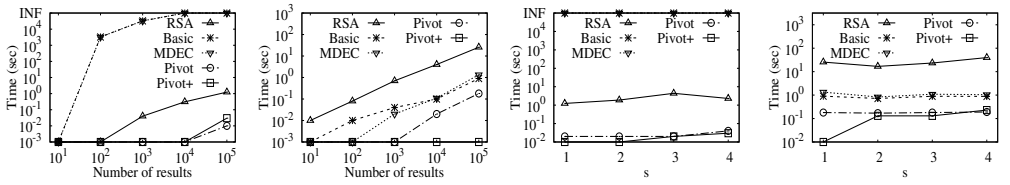
maximal s-defective cliques containing v. Pivot2+ is an improved Pivot2 algorithm equipped with the optimization techniques shown in Lemma 2 and Lemma 3. Note that both Pivot2 and Pivot2+ are equipped with the degeneracy ordering technique. All algorithms are implemented in C++, and tested on a PC with one 2.2 GHz CPU and 128GB of RAM running CentOS.

Datasets. We use 11 real-world networks to test the performance of proposed algorithms, including collaboration networks, web graphs, social networks, scientific computing networks, and so on. The detailed statistics of these datasets are shown in Table 1, where d_{max} and δ denote the maximum degree and the degeneracy of the graph respectively. Note that in the experiments, we use 6 small real-world graphs (the first six datasets in Table 1) to evaluate the performance of different algorithms for enumerating all maximal s-defective cliques, because both RSA and Basic are often too expensive to enumerate all results on large graphs. We use 5 large graphs (the last five datasets in Table 1) to evaluate the performance of different algorithms for enumerating relatively-large maximal s-defective cliques. All datasets can be downloaded from https://networkrepository.com/index.php.

Parameters. For all algorithms, we set the parameter s to be an integer falling in the interval $[1, 4]$ with a default value of 2. To enumerate relatively-large maximal s-defective cliques, we set the threshold parameter q to be an integer falling in the interval $[8, 20]$ with a default value of 12. We will study the effect of our algorithms with varying these two parameters.

6.2 Efficiency Testing

Exp-1: Results of enumerating all maximal s-defective cliques. In this experiment, we evaluate the performance of various algorithms to enumerate all maximal s-defective cliques. Table 2 shows the runtime of RSA, Basic, MDEC, Pivot, and Pivot+, on 6 small real-world graphs, respectively.



(a) lolgop ($s = 1$) (b) web-edu ($s = 1$) (c) lolgop (Output 10^5 results) (d) web-edu (Output 10^5 results)

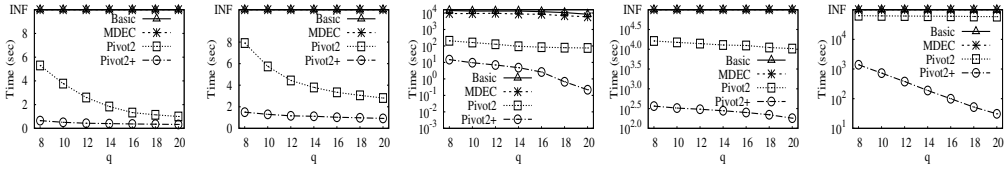
Fig. 4. Delay of various algorithms on real-world graphs.

The column #Nums in Table 2 denotes the number of maximal s-defective cliques in each graph with a specified parameter s. Note that we set the runtime of an algorithm to "INF" if it cannot terminate within 24 hours. As can be seen, both Pivot and Pivot+ substantially outperform Basic and MDEC. This result indicates that the proposed pivoting technique indeed plays a crucial role in pruning unnecessary computations of the enumeration procedure. Specifically, our algorithms Pivot and Pivot+ are orders of magnitude faster than Basic and RSA, which demonstrates the high-efficiency of our pivot-based algorithms. We also notice that the performance of Basic and MDEC is comparable on all tested datasets. This result suggests that the adaption of the existing maximum s-defective clique algorithm [13] cannot work well for the maximal s-defective clique enumeration problem, because there still exist a large number of non-maximal s-defective cliques explored by the algorithm.

When comparing the two pivot-based algorithms Pivot and Pivot+, we can observe that if s is no larger than 3, Pivot+ is often faster than Pivot on most datasets. This result confirms that the optimization techniques developed in Section 4.3 is very effective when s is small. When s gets larger, Pivot+ may generate more non-maximal s-defective cliques in the recursive procedure (Line 8 of Algorithm 6), thus degrading the performance of the algorithm. In practice, we suggest to use Pivot+ to enumerate all maximal s-defective cliques of a graph if $s \leq 3$, otherwise it is better to use Pivot.

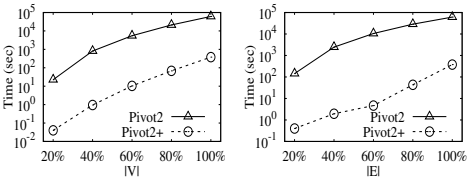
We can also observe that RSA works well when s is small, but it is often very costly when $s \geq 3$. Moreover, we can see that for small dense graphs, RSA is often much faster than Basic and MDEC when $s \leq 2$. For example, on scc-enron, RSA can output all maximal 1-defective cliques using only 0.08 seconds, while Basic and MDEC cannot terminate within 24 hours. This result indicates that RSA is not only with theoretical interests, but it also works well in small real-world graphs given that $s \leq 2$. In addition, since RSA can output two consecutive results within polynomial time, it might be useful for the applications that only requires a part of results.

Exp-2: Time delay of various algorithms. Fig. 4 shows the runtime of each algorithm on lolgop and web-edu with varying s and varying the number of results. The results on other datasets are consistent. From Fig. 4, we can see that the runtime of RSA scales linearly with respect to (w.r.t.) the number of results, which confirms our polynomial-delay time complexity analysis of RSA in Section 3.2. Moreover, the pivot-based algorithms Pivot and Pivot+, although they cannot guarantee a polynomial-delay time complexity in theory, always achieve the best performance among all tested algorithms under all parameter settings. This is mainly attributed to the fact that the proposed pivoting technique avoids a large number of redundant computations. However, the runtime of Basic and MDEC on some datasets, such as lolgop, may increases exponentially with the number of returned results, indicating that these baselines involve a large amount of redundant computations. In addition, when specifying the desired number of output results (e.g 10^5 results), the runtime of all our algorithms increases very smoothly as s increases. These results further confirm that our algorithms are also useful for applications that only need to obtain a part of results even when $s > 2$.



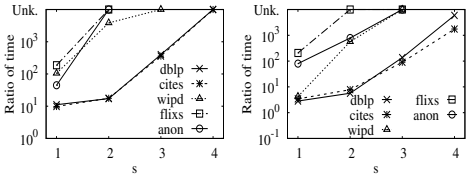
(a) dblp-2010 ($s = 2$) (b) ca-citeseer ($s = 2$) (c) wikipedia2009 ($s = 2$) (d) flixster ($s = 2$) (e) socfb-B-anon ($s = 2$)

Fig. 5. Runtime of different algorithms with varying q on large real-world graphs.



(a) $s = 2, q = 12$ (b) $s = 2, q = 12$

Fig. 6. Scalability testing (socfb-B-anon).



(a) $q = 12$ (b) $q = 20$

Fig. 7. The ratios of the runtime of enumerating maximal $(s+1)$-plexes to the runtime of enumerating maximal s-defective cliques on various datasets with the same parameters.

Exp-3: Results of enumerating all relatively-large maximal s-defective cliques. Fig. 5 shows the results of Basic, MDEC, Pivot2, and Pivot2+ on 5 large real-world graphs with $s = 2$ and varying q. The results are consistent for other s values. Note that for a fair comparison, in this experiment, we also make use of the Property 2 to reduce the size of the candidate set in algorithms Basic and MDEC. That is to say, the only difference between Basic (MDEC) and Pivot2 in this experiment is whether the proposed pivoting technique is used or not. From Fig. 5, we can observe that Pivot2+ consistently outperforms the other algorithms on all datasets. Specifically, under most parameter settings, Pivot2+ can be more than two orders of magnitude faster than Pivot2 and Basic. This result indicates that the proposed pivoting technique is indeed very effective in improving the performance of enumerating maximal s-defective cliques. Moreover, when comparing Pivot2 and Basic (MDEC), we can see that Basic (MDEC) cannot handle most large real-world graphs, while Pivot2 is still very efficient to process all graphs with most parameter settings. This result indicates that the proposed pivoting technique indeed plays an important role in reducing unnecessary computations.

Exp-4: Scalability testing. To evaluate the scalability of the proposed pivot-based algorithms, we generate eight subgraphs by randomly sampling 20-80% vertices or edges from the largest dataset socfb-B-anon. Similar results can also be observed on the other datasets. We run our algorithms on these subgraphs and report their runtime. Fig. 6 depicts the results of Pivot2 and Pivot2+ with $s = 2$. As can be seen, the runtime of the proposed pivot-based algorithms increases smoothly as $|V|$ or $|E|$ increases. This result indicates that the proposed pivot-based algorithms scale well on real-world graphs. In addition, Pivot2+ always shows excellent performance in all parameter settings compared to Pivot2, which further demonstrates that Pivot2+ is capable of handling large real-world graphs.

6.3 Effectiveness Testing

In this subsection, we evaluate the effectiveness of our solutions. To this end, we compare the s-defective clique model with the k-plex model which is a well-known relaxed clique model and is also closely-related to the s-defective clique model. Specifically, a subset C of V is called a k-plex if every vertex in $G(C)$ has at least $|C| - k$ neighbors. When $k = 1$, the k-plex is a clique which

Table 3. Prediction accuracy on a PPI network (CORE).

s	$(s+1)$-plexes		s-defective cliques	
	#Nums	Accuracy	#Nums	Accuracy
1	206	**0.621**	160	0.6
2	673	0.513	569	**0.614**
3	2192	0.382	1390	**0.608**
4	29246	0.182	3530	**0.465**

Table 4. Statistical analysis on stock markets.

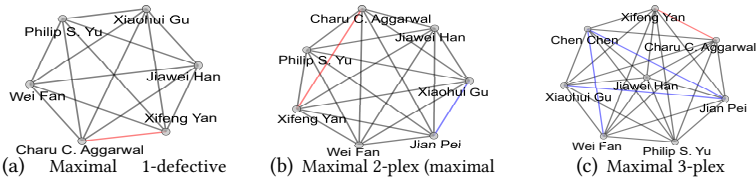
θ	Market graph size			Maximum size (varying s)				
	n	m	d_{max}	0	1	2	5	10
0.4	2784	218956	1166	116	117	118	120	123
0.5	2154	56218	694	57	58	59	60	63
0.6	1254	15488	260	38	39	39	41	42
0.7	562	3908	47	18	18	19	20	22
0.8	224	606	16	8	8	8	9	10

corresponds to the case of $s = 0$ for the s-defective clique. Moreover, it is easy to check that any s-defective clique is also an $(s+1)$-plex. Therefore, the k-plex model is a very good baseline for our experiments.

Exp-5: Runtime of enumerating different relaxed cliques. In this experiment, we compare the runtime of our pivot-based algorithm (Pivot2+) and that of Plex (the state-of-the-art maximal $(s+1)$-plexes enumeration algorithm in [49, 54]). Fig. 7 shows the results on each dataset with varying s. The vertical coordinate in the figures represents the ratios of the runtime of Plex to the runtime of Pivot2+. "Unk." denotes that the Plex algorithm cannot complete the computations within 24 hours. Here we also abbreviate the name of each dataset to reduce the space usage in the figures. From Fig. 7, we can observe that Pivot2+ exhibits much better performance than Plex on all datasets. More specifically, Pivot2+ can be more than two orders of magnitude faster than Plex on most parameter settings, and the gap between the runtime of Plex and Pivot2+ increases dramatically as s increases. The results confirm that the s-defective clique model is indeed easier to compute than the k-plex model.

Exp-6: Protein interaction prediction. In this case study, we test the effectiveness of the s-defective clique model to predict implicit protein interactions in a protein-protein interaction (PPI) network. As shown in [3, 23], a protein complex is usually in the form of a complete subgraph (i.e., each pair of proteins in a protein complex is interacting with each other). Due to this property, the problem of predicting implicit protein interactions is often considered as a *clique complement* problem, which is slightly different from the classic link prediction problem. Thus, to identify implicit protein interactions, it is better to use a relaxed clique model in which all missing edges can be considered as implicit protein interactions. The s-defective clique is a good choice for this goal. Since the k-plex is a widely-used relaxed clique model and is also closely related to the s-defective clique model, it can be served as a very good baseline for this case study. In the experiment, we first use the proposed algorithms and the algorithms in [49, 54] to compute s-defective cliques and $(s+1)$-plexes with size no less than 10, respectively. Then, we add all missing edges in each detected relaxed clique and evaluate the accuracy of edges detected by the relaxed clique models. Table 3 shows the prediction accuracy of these two different models (i.e, the ratio of the number of true positive edges to all added edges) on CORE [26], which contains 2708 vertices and 7123 edges and the ground truth protein interactions can be determined by the MIPS protein database [26]. From Table 3, we observe that the s-defective clique model is often more accurate than the $(s+1)$-plex model with varying s (except for $s = 1$, and in this case the s-defective clique model is also comparable to the k-plex model). Moreover, with the increase of s, the accuracy gap between the s-defective clique model and the $(s+1)$-plex model tends to be larger. This result demonstrates the high effectiveness of our solutions for protein interaction prediction in PPI networks.

Exp-7: Collaboration relationship prediction on DBLP. Here we evaluate the effectiveness of the s-defective clique model to predict the collaboration relationship on DBLP (https://dblp.uni-trier.de). To this end, we use the DBLP collaboration network before 2011, and evaluate the prediction



(a) Maximal 1-defective clique (b) Maximal 2-plex (maximal 2-defective clique) (c) Maximal 3-plex

Fig. 8. Collaboration relationship prediction on DBLP (red and blue edges are the true and false positive collaborators respectively).

accuracy using the full DBLP dataset. Fig. 8 shows the results of Prof. Jiawei Han's communities which are generated by the s-defective clique model and the $(s+1)$-plex model. As can be seen, the s-defective clique model can exactly predict a new collaboration relationship (Prof. Charu C. Aggarwal and Prof. Xifeng Yan in Fig. 8(a)) with $s = 1$, while the $(s+1)$-plex model will introduce a false-positive collaboration relationship (Prof. Jian Pei and Prof. Xiaohui Gu in Fig. 8(b)). Moreover, when increasing s, (i.e, $s = 2$), the result obtained by the 3-plex model (Fig. 8(c)) yields more false positive relationships compared to the 2-defective clique model (Fig. 8(b)). This result further confirm the high effectiveness of the s-defective clique model for implicit interaction prediction.

Exp-8: Statistical analysis on stock markets. We further apply the s-defective clique to conduct a statistical analysis on a US stock market dataset[1], which contains the daily trading information for 7,195 financial instruments. The market graphs are constructed by the method used in [8, 9]. Specifically, for each pair of instruments i and j, if the correlation coefficient C_{ij} between them is no less than a given threshold θ, we simply add an edge between i and j, which represents that the instruments i and j have a similar price fluctuation behavior. Denote by $P_i(t)$ the price of a financial instrument on day t and $R_i(t) = ln(P_i(t)/P_i(t-1))$ the logarithm of the return of instrument i on the day from $t-1$ to t. Then, the correlation coefficient between a pair of instruments i and j can be defined as $C_{ij} = \frac{E(R_iR_j)-E(R_i)E(R_j)}{\sqrt{Var(R_i)Var(R_j)}}$, where $E(R_i)$ and $Var(R_i)$ are the expectation and variance of the returns of instrument i over a given period of days, respectively. As can be seen, the value of C_{ij} lies between -1 to 1, and the larger C_{ij} is, the more correlative the price fluctuations of i and j are. Clearly, a cohesive subgraph on a market graph represents a set of instruments with similar price fluctuations. In our experiment, we use instrument trading prices from 2000 to 2006 to construct our market graphs. Table 4 shows statistics of the market graphs with varying θ and the exact sizes of the maximum s-defective cliques on each market graph with varying s. From Table 4, we observe that the sizes of the maximum s-defective cliques are relatively large, which implies that there are a large number of instruments whose price fluctuations behave similarly over time in stock markets. Moreover, as the increase of s, the size of maximum s-defective clique increases accordingly, which means that some indirectly-linked (but closely-related) financial instruments can also be detected by the s-defective clique model. For instance, when $\theta = 0.6$, the size of maximum clique ($s = 0$) is 38, which are the instruments related to *real estate*, while increasing s to 5, three additional instruments "cpt.us", "bxp.us", and "els.us" can be detected by the s-defective clique. These additional instruments are very relevant to the *real estate investment*.These results indicate that the s-defective clique model is a powerful tool in the application of statistical analysis on stock markets.

7 RELATED WORKS

Maximal clique enumeration. The maximal clique enumeration problem had been well studied in recent decades [10, 12, 19, 21, 38, 47]. Perhaps, the most impressive technique is the Bron-Kerbosch

[1]https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

(BK) algorithm [10], which is a backtracking enumeration algorithm with a pivoting technique. Following this work, many interesting variants of the BK algorithms were proposed. For example, Tomita et al. [47] proposed an $O(3^{n/3})$ BK algorithm with a careful-designed pivoting technique. Such an algorithm was shown to be optimal, as there exists an graph having $O(3^{n/3})$ maximal cliques in the worst case [36]. Eppstein et al. [21] presented an improved BK algorithm with the time complexity closely related to the degeneracy of the graph [32]. In addition, a refined pivot-selection technique [38] and a dominated-set based technique [12] were also developed to improve the BK algorithm when processing dense graphs. For sparse graphs, however, the bit-parallelism techniques were proposed to improve the BK algorithm [19, 43]. Recently, the maximal clique model was also generalized to other types of graphs. Notable examples include maximal cliques on uncertain graphs [30, 37, 55], maximal bipartite cliques on bipartite graphs [1, 28, 52], maximal cliques on temporal graphs [25, 42], and maximal signed cliques on signed graphs [14, 29]. Unlike all these mentioned maximal clique algorithms, this paper focuses mainly on the s-defective clique model and developing efficient techniques to enumerate all maximal s-defective cliques.

Maximal relaxed clique enumerations. In real-world applications, using the clique model to mine cohesive subgraphs may be overly strict due to the presence of noises in graph datasets. Many relaxed clique models have been developed as alternatives. Except for the s-defective clique, several representative relaxed clique models include k-plex [7, 17, 44, 54], s-bundle [40, 53], s-clique [6, 35], and γ-quasi-clique [34, 41]. Existing algorithms for enumerating all these relaxed cliques can be classified into two categories. The first category of algorithms are to convert the original problem to the problem of enumerating all maximal relaxed cliques in each almost-satisfying subgraphs [15] if the relaxed clique model admits the hereditary property, such as the techniques presented in [6, 7]. The other category of algorithms are based on the set enumeration technique or the branch-and-bound technique to check whether each subset of a given graph is the maximal relaxed clique [17, 34, 41, 53, 54]. In addition, some of these models were also extended to different types of graphs. For instance, [51] developed an algorithm to enumerate all maximal k-plexes in bipartite graphs. [33] studied the problem of finding stable quasi-cliques in temporal graphs. In this work, we focus on developing efficient solutions with nontrivial time complexity guarantees to enumerate all and relatively-large s-defective cliques.

8 CONCLUSION

In this paper, we systematically investigate the maximal s-defective clique enumeration problem, and propose several novel and efficient algorithms to solve this problem. The first proposed algorithm is an output-sensitive algorithm based on a carefully-developed reverse search technique, which can return any consecutive solutions within polynomial time. The second proposed algorithm is a more practical algorithm based on a branch-and-bound enumeration and a novel pivoting technique. We prove that the time complexity of our pivot-based branch-and-bound algorithm can break the $O(2^n)$ worst-case enumeration complexity. We also develop several new pruning techniques to further improve the efficiency of our pivot-based branch-and-bound algorithm. Additionally, we also extend our pivot-based branch-and-bound algorithm to enumerate all maximal subgraphs that satisfy a hereditary property. Finally, the results of comprehensive experiments demonstrate the efficiency, effectiveness, and scalability of the proposed approaches.

ACKNOWLEDGMENTS

This work was partially supported by (i) National Key Research and Development Program of China 2020AAA0108503, (ii) NSFC Grants U2241211, 62072034, U1809206 and (iii) CCF-Huawei Populus Grove Fund. Rong-Hua Li is the corresponding author of this paper.

REFERENCES

[1] Aman Abidi, Rui Zhou, Lu Chen, and Chengfei Liu. 2020. Pivot-based Maximal Biclique Enumeration. In *IJCAI*. 3558–3564.

[2] David Avis and Komei Fukuda. 1996. Reverse Search for Enumeration. *Discret. Appl. Math.* 65, 1-3 (1996), 21–46.

[3] Gary D. Bader and Christopher WV Hogue. 2002. Analyzing yeast protein–protein interaction data obtained from different sources. *Nature biotechnology* 20, 10 (2002), 991–997.

[4] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O(m) Algorithm for Cores Decomposition of Networks. *CoRR* cs.DS/0310049 (2003).

[5] Punam Bedi and Chhavi Sharma. 2016. Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 6, 3 (2016), 115–135.

[6] Rachel Behar and Sara Cohen. 2018. Finding All Maximal Connected s-Cliques in Social Networks. In *EDBT*. 61–72.

[7] Devora Berlowitz, Sara Cohen, and Benny Kimelfeld. 2015. Efficient Enumeration of Maximal k-Plexes. In *SIGMOD*. 431–444.

[8] Vladimir Boginski, Sergiy Butenko, and Panos M Pardalos. 2005. Statistical analysis of financial networks. *Computational statistics & data analysis* 48, 2 (2005), 431–443.

[9] Vladimir Boginski, Sergiy Butenko, and Panos M Pardalos. 2006. Mining market data: A network approach. *Computers & Operations Research* 33, 11 (2006), 3171–3184.

[10] Coenraad Bron and Joep Kerbosch. 1973. Finding All Cliques of an Undirected Graph (Algorithm 457). *Commun. ACM* 16, 9 (1973), 575–576.

[11] Sergiy Butenko and Wilbert E. Wilhelm. 2006. Clique-detection models in computational biochemistry and genomics. *Eur. J. Oper. Res.* 173, 1 (2006), 1–17.

[12] Frédéric Cazals and Chinmay Karande. 2006. *Reporting maximal cliques: new insights into an old problem.* Ph. D. Dissertation. INRIA.

[13] Xiaoyu Chen, Yi Zhou, Jin-Kao Hao, and Mingyu Xiao. 2021. Computing maximum k-defective cliques in massive graphs. *Comput. Oper. Res.* 127 (2021), 105131.

[14] Zi Chen, Long Yuan, Xuemin Lin, Lu Qin, and Jianye Yang. 2020. Efficient Maximal Balanced Clique Enumeration in Signed Networks. In *WWW*. 339–349.

[15] Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. 2008. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *J. Comput. Syst. Sci.* 74, 7 (2008), 1147–1159.

[16] Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. 2019. Listing Maximal Subgraphs Satisfying Strongly Accessible Properties. *SIAM J. Discret. Math.* 33, 2 (2019), 587–613.

[17] Alessio Conte, Tiziano De Matteis, Daniele De Sensi, Roberto Grossi, Andrea Marino, and Luca Versari. 2018. D2K: Scalable Community Detection in Massive Networks via Small-Diameter k-Plexes. In *KDD*. 1272–1281.

[18] Qiangqiang Dai, Rong-Hua Li, Meihao Liao, and Guoren Wang. 2023. Maximal Defective Clique Enumeration (full version). https://github.com/qq-dai/DefectiveClique/blob/main/DefectiveClique-full.pdf

[19] Naga Shailaja Dasari, Desh Ranjan, and Mohammad Zubair. 2014. pbitMCE: A bit-based approach for maximal clique enumeration on multicore processors. In *ICPADS*. 478–485.

[20] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. 2007. Community detection in large-scale social networks. In *WebKDD workshop*. 16–25.

[21] David Eppstein, Maarten Löffler, and Darren Strash. 2010. Listing All Maximal Cliques in Sparse Graphs in Near-Optimal Time. In *ISAAC*, Vol. 6506. 403–414.

[22] V. Fomin Fedor and Kratsch Dieter. 2010. *Exact Exponential Algorithms.* Springer.

[23] Anne-Claude Gavin, Markus Bösche, Roland Krause, Paola Grandi, Martina Marzioch, Andreas Bauer, Jörg Schultz, Jens M. Rick, Anne-Marie Michon, Cristina-Maria Cruciat, et al. 2002. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature* 415, 6868 (2002), 141–147.

[24] Eric Harley, Anthony Bonner, and Nathan Goodman. 2001. Uniform integration of genome mapping data using intersection graphs. *Bioinformatics* 17, 6 (2001), 487–494.

[25] Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. 2016. Enumerating maximal cliques in temporal graphs. In *ASONAM*. 337–344.

[26] George Kollios, Michalis Potamias, and Evimaria Terzi. 2013. Clustering Large Probabilistic Graphs. *IEEE Trans. Knowl. Data Eng.* 25, 2 (2013), 325–336.

[27] Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: a comparative analysis. *Physical review E.* 80, 5 (2009), 056117.

[28] Jinyan Li, Guimei Liu, Haiquan Li, and Limsoon Wong. 2007. Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms. *IEEE Trans. Knowl. Data Eng.* 19, 12 (2007), 1625–1637.

[29] Rong-Hua Li, Qiangqiang Dai, Lu Qin, Guoren Wang, Xiaokui Xiao, Jeffrey Xu Yu, and Shaojie Qiao. 2018. Efficient Signed Clique Search in Signed Networks. In *ICDE*. 245–256.
[30] Rong-Hua Li, Qiangqiang Dai, Guoren Wang, Zhong Ming, Lu Qin, and Jeffrey Xu Yu. 2019. Improved Algorithms for Maximal Clique Search in Uncertain Networks. In *ICDE*. 1178–1189.
[31] Rong-Hua Li, Qiushuo Song, Xiaokui Xiao, Lu Qin, Guoren Wang, Jeffrey Xu Yu, and Rui Mao. 2022. I/O-Efficient Algorithms for Degeneracy Computation on Massive Networks. *IEEE Trans. Knowl. Data Eng.* 34, 7 (2022), 3335–3348.
[32] Don R. Lick and Arthur T. White. 1970. k-Degenerate graphs. *Canadian Journal of Mathematics* 22, 5 (1970), 1082–1096.
[33] Longlong Lin, Pingpeng Yuan, Rong-Hua Li, Jifei Wang, Ling Liu, and Hai Jin. 2022. Mining Stable Quasi-Cliques on Temporal Networks. *IEEE Trans. Syst. Man Cybern. Syst.* 52, 6 (2022), 3731–3745.
[34] Guimei Liu and Limsoon Wong. 2008. Effective Pruning Techniques for Mining Quasi-Cliques. In *ECML/PKDD*, Vol. 5212. 33–49.
[35] R. Duncan Luce. 1950. Connectivity and generalized cliques in sociometric group structure. *Psychometrika* 15, 2 (1950), 169–190.
[36] John W. Moon and Leo Moser. 1965. On cliques in graphs. *Israel journal of Mathematics* 3, 1 (1965), 23–28.
[37] Arko Mukherjee, Pan Xu, and Srikanta Tirthapura. 2017. Enumeration of Maximal Cliques from an Uncertain Graph. *IEEE Trans. Knowl. Data Eng.* 29, 3 (2017), 543–555.
[38] Kevin A. Naudé. 2016. Refined pivot selection for maximal clique enumeration in graphs. *Theor. Comput. Sci.* 613 (2016), 28–37.
[39] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* 435, 7043 (2005), 814–818.
[40] Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko. 2013. On clique relaxation models in network analysis. *Eur. J. Oper. Res.* 226, 1 (2013), 9–18.
[41] Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *KDD*. 228–238.
[42] Hongchao Qin, Rong-Hua Li, Guoren Wang, Lu Qin, Yurong Cheng, and Ye Yuan. 2019. Mining Periodic Cliques in Temporal Networks. In *ICDE*. 1130–1141.
[43] Pablo San Segundo, Jorge Artieda, and Darren Strash. 2018. Efficiently enumerating all maximal cliques with bit-parallelism. *Comput. Oper. Res.* 92 (2018), 37–46.
[44] Stephen B. Seidman and Brian L. Foster. 1978. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology* 6, 1 (1978), 139–154.
[45] Uno Takeaki. 2003. *Two general methods to reduce delay and change of enumeration algorithms.* Technical Report. Technical Report.
[46] Etsuji Tomita, Tatsuya Akutsu, and Tsutomu Matsunaga. 2011. *Efficient algorithms for finding maximum and maximal cliques: Effective tools for bioinformatics.* IntechOpen.
[47] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.* 363, 1 (2006), 28–42.
[48] Svyatoslav Trukhanov, Chitra Balasubramaniam, Balabhaskar Balasundaram, and Sergiy Butenko. 2013. Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations. *Comput. Optim. Appl.* 56, 1 (2013), 113–130.
[49] Zhengren Wang, Yi Zhou, Mingyu Xiao, and Bakhadyr Khoussainov. 2022. Listing Maximal k-Plexes in Large Real-World Graphs. In *WWW*.
[50] Haiyuan Yu, Alberto Paccanaro, Valery Trifonov, and Mark Gerstein. 2006. Predicting interactions in protein networks by completing defective cliques. *Bioinform.* 22, 7 (2006), 823–829.
[51] Kaiqiang Yu, Cheng Long, Shengxin Liu, and Da Yan. 2022. Efficient Algorithms for Maximal k-Biplex Enumeration. In *SIGMOD*. 860–873.
[52] Yun Zhang, Charles A. Phillips, Gary L. Rogers, Erich J. Baker, Elissa J. Chesler, and Michael A. Langston. 2014. On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. *BMC Bioinform.* 15 (2014), 110.
[53] Yi Zhou, Weibo Lin, Jin-Kao Hao, Mingyu Xiao, and Yan Jin. 2022. An effective branch-and-bound algorithm for the maximum s-bundle problem. *Eur. J. Oper. Res.* 297, 1 (2022), 27–39.
[54] Yi Zhou, Jingwei Xu, Zhenyu Guo, Mingyu Xiao, and Yan Jin. 2020. Enumerating Maximal k-Plexes with Worst-Case Time Guarantee. In *AAAI*. 2442–2449.
[55] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. 2010. Finding top-k maximal cliques in an uncertain graph. In *ICDE*. 649–652.

Received July 2022; revised October 2022; accepted November 2022