Efficient Maximum Defective Clique Search on Real-world Networks

ABSTRACT

....

1 INTRODUCTION

.

2 PROBLEM DEFINITION

Consider an undirected and unweighted graph $G = (V, E)$, where V and E represent the sets of vertices and edges of the graph G, respectively. Let $n = |V|$ and $m = |E|$ denote the number of vertices and edges in G, respectively. For a vertex v of G, we define $N_v(G)$ as the set of neighbors of v in G, i.e., $N_v(G) = \{u \in V | (u, v) \in E\}$. The degree of v in G is the cardinality of $N_v(G)$, denoted by $d_v(G) = |N_v(G)|$. Given a vertex subset S of G, we let $G(S) = (S, E_S)$ be the subgraph of G induced by the subset S, where $E_S = \{(u, v) \in E | u \in S, v \in S\}$. In accordance with [9], the k-defective clique of G is defined as follows.

Definition 1 (k-defective clique). Given a graph G and a non-negative integer k, the subgraph $G(S)$ induced by the vertex set $S \subseteq V$ is a k-defective clique if there exists at least $\binom{|S|}{2} - k$ edges in $G(S)$.

For simplicity, in the rest of this paper, we directly refer the set S as the k-defective clique of G. A k-defective clique S of G is considered maximal if there does not exist any other k-defective clique S' of G such that $S \subset S'$. Furthermore, a k-defective clique S of G is designated as maximum if its size is largest among all maximal k-defective clique of G, where the size of k-defective clique S is defined as the number of vertices it contains. Then, two useful properties of the k-defective clique are described below, which are very helpful in designing the algorithms.

Property 1 (Hereditary [1]). Given a k-defective clique S of G, every subset of S is also a k-defective clique of G.

The hereditary property (Property 1) of a k-defective clique simplifies the maximality check process. Specifically, if a k-defective clique S of G is not the maximal, it implies the existence of a vertex in $V \setminus S$ that can form a larger k-defective clique when combined with S. Thus, this property forms the foundation for the design of our algorithms.

Property 2 (Small Diameter [2]). Given a k-defective clique S of G, the diameter of $G(S)$ is no larger than 2 if $|S| \geq k + 2$.

SIGMOD '24, June 9–15, 2024, Santiago, Chile

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Algorithm 1: Graph coloring

Input: The graph $G = (V, E)$
Output: The color number of each vertex in V

Compute the degeneracy ordering of vertices in G;
$\tau \leftarrow$ a positive number; $\omega \leftarrow 0$;
for *each* $v \in V$ *in reverse order with the degeneracy ordering* **do**
  color$(v) \leftarrow 0$;
  while $\exists u \in N_v(G)$ *with* $color(v) = color(u)$ **do**
    color$(v) \leftarrow$ color$(v) + 1$;
  /* Recoloring */
  if $color(v) \geq \tau$ **then**
    for $c = 0$ *to* $\tau - 1$ **do**
      if $|\{u \in N_v^+(G) | color(u) = c\}| = 1$ **then**
        Let u be the vertex in $N_v^+(G)$ with color$(u) = c$;
        if $\exists c' \in [0, \tau - 1]$ *s.t.* $\nexists w \in N_u(G)$ *with* $color(w) = c'$ **then**
          color$(u) \leftarrow c'$; color$(v) \leftarrow c$;
          break;
  $\omega \leftarrow \max\{\omega, \text{color}(v)\}$
return ω;

Property 2 not only implies the internal density-connected nature (having a diameter of two with a size no less than $k + 2$) of the k-defective clique but also provides an acceleration for enumerating relatively-large maximal k-defective cliques [**?**]. However, the problem of enumerating maximal k-defective clique often suffers from excessively long computational times, as there can be an exponential number of maximal k-defective cliques compared to the number of vertices. To address this challenge, in this paper, we aim to the problem of finding a maximum k-defective clique of G, and the formal problem definition is shown below.

Problem definition. Given a graph G and a non-negative integer k, the goal of this paper is to compute a maximum k-defective clique of G.

2.1 Existing Solutions

As shown in [6, 8], the problem of finding the maximum k-defective clique of a given graph G is NP-complete, thus there is no algorithm with polynomial-time to solve this problem. To our knowledge, there are several solutions have been developed to address the problem [1, 3, 4, 6], which mainly consist of two categories.

Russian doll search based algorithms. The first algorithm for finding the maximum k-defective clique is developed by Trukhanov et al. [6], which is based on a Russian doll search technique [7]. The key idea of such an algorithm is summarized as follows. Let $\{v_1, v_2, ..., v_n\}$ be a total ordering for the vertices in V of G. The problem of finding the maximum k-defective clique from a graph G can be divided into a series of n nested subproblems. Each subproblem represents finding the maximum k-defective clique that includes v_i in the subgraph $G(\{v_i, v_{i+1}, ..., v_n\})$, where $v_i \in V$. Then, the algorithm starts from $i = n$ and iterates down to $i = 1$, processing each subproblem along the way. The final maximum solution is obtained when $i = 1$. In addition, the algorithm employs the branch-and-bound technique, which can be accelerated using the pre-knowledge when solving the i-th subproblem. Specifically, based on the upper bound on the size of the maximum k-defective clique that has been detected by j-th subproblem (where $j > i$), it is possible to determine whether a larger k-defective clique is also likely to contain the vertex v_j. Recently, such the Russian doll search algorithm has also been improved by Gschwinda et al. [4] based on some auxiliary information prior computed.

Branch-and-bound based algorithms. In order to enhance the efficiency of finding the maximum k-defective clique in a graph G, several new enumeration algorithms have been developed [1, 3] based on a branch-and-bound technique [5]. Notably, Chen et al. [1] propose a new branching rule and employ some reduction techniques to improve the search for the maximum k-defective clique. This proposed branching rule prioritizes the vertices that have non-neighbors within the current k-defective clique S to expand S, thus ensuring a maximum of $k + 1$ subbranches in each recursive call. Furthermore, the authors establish that the worst-case time complexity of this technique for finding maximum k-defective clique is bounded by $O(P(n)\gamma_k^n)$, where $P(n)$ is a polynomial function related to n and γ_k is a real-number less than 2. To further reduce unnecessary computations, an improved branch-and-bound enumeration algorithm is presented in [3] based on some newly developed branch pruning techniques. To our knowledge, this algorithm represents the current state-of-the-art for solving the problem of finding the maximum k-defective cliques.

Nevertheless, the these existing solutions still suffer from significant computational time when applied to real-world graphs. This issue arises primarily due to the insufficient tightening of bounding techniques in these algorithms, which leads to a proliferation of unnecessary computations during the branch-and-bound procedure. Additionally, the efficiency of the employed branching rules in swiftly identifying the maximum k-defective clique is also limited. Furthermore, it is noteworthy that the worst-case time complexity of most existing solutions remains bounded by $O(P(n)2^n)$, and only one approach [1] achieves a worst-case time complexity of $O(P(n)\gamma_k^n)$, where $\gamma_k < 2$. However, even for small values of k, the value of γ_k is nearly equal to 2. For instance, as reported in [1], when $k = 2$ and 3, the corresponding values of γ_k are 1.996 and 1.999, respectively, which are unacceptably high for the problem of finding the maximum k-defective clique in graph G.

Hence, there is a pressing need to develop more efficient algorithms to finding the maximum k-defective clique of real-world graphs. In the subsequent sections, we first introduce some new bounding techniques and subsequently present our approaches that enable the efficient discovery of the maximum k-defective clique.

3 BOUNDING TECHNIQUES

Let κ and $\kappa(C)$ denote the sizes of maximum k-defective clique in graph G and the subgraph $G(C)$, respectively. In this section, we start to explore both typical and novel upper bounds for κ and $\kappa(C)$, which play crucial role in accelerating the computations related to the maximum k-defective clique.

Degree-based upper bound. The first upper bound is simply derived from the degree information of vertices in G, and the detailed result is presented the following lemma.

Lemma 1. For a given graph G, the size of the maximum k-defective clique of G containing a vertex $v \in V$ is no larger than $d_v(G) + k + 1$. Consequently, we can establish $\kappa \leq \max_{v \in V} d_v(G) + k + 1$.

Proof. This lemma is clearly established based on the definition of k-defective clique. □

Core-based upper bound. Next, we introduce a tighter upper bound for κ and $\kappa(C)$ based on a well-established concept of k-core [**?**]. The formal definition of k-core is provided as follows.

Definition 2. Given a graph G, the subgraph $G(C)$ of G induced by the vertex set C is a k-core of G if $d_v(C) \geq k$ for every v in C.

Let C_k represent k-core subgraph of G. The core number of a vertex v in G, denoted by $core_v(G)$, is defined as the maximum value of k such that v belongs to the k-core subgraph C_k of G, i.e., $core_v(G) = \max\{k \mid v \in C_k\}$. Based on this concept, we can derive a core number based upper bound, as shown below.

Lemma 2. For a given graph G, the size of the maximum k-defective clique in G containing a vertex $v \in V$ is bounded by $core_v(G) + k + 1$. Consequently, we can establish $\kappa \leq \max_{v \in V} core_v(G) + k + 1$.

Proof. It is evident that any k-defective clique S of G is also a $(|S| - k - 1)$-core subgraph of G, as per the definitions provided in Definition 1 and Definition 2. Consequently, if a k-defective clique S in G includes a vertex v, it follows that the size of such a k-defective clique is bounded by $core_v(G) + k + 1$. Thus, this lemma is established. □

To compute the core number for each vertex in a given graph G, the traditional peeling technique described in [**?**] can be employed. This technique follows an iterative process where the vertices are sequentially removed from the remaining subgraph based on their degree, in a non-decreasing order starting from $k = 0$ up to $k = n$. The core number of a vertex is assigned to the current value of k at the time of its removal. The total time-consuming for this technique is bounded by $O(m)$, indicating its high efficiency in generating the core number based upper bound.

Color-based upper bound. Furthermore, we present a method to refine the upper bound for κ and $\kappa(C)$ using a graph coloring technique. Here we begin by providing the fundamental definition of graph coloring below.

Definition 3. Given a graph G, the graph coloring is to assign a color number for each vertex v of G, denoted by $color_v(G)$, such that any two adjacent vertices have different colors. Formally, for every $(u, v) \in E$, it should hold that $color_v(G) \neq color_u(G)$.

Denote by ω and $\omega(C)$ the number of distinct colors in G and the subgraph $G(C)$ of G induced by C, respectively. Based on the concept of graph coloring, we can establish the following upper bound for κ and $\kappa(C)$.

Lemma 3. Given a coloring of the graph G and a subset C within G, the size of the maximum k-defective clique in G and $G(C)$ can be bounded by $\omega + k$ and $\omega(C) + k$, respectively.

Proof. Given a k-defective clique S in G, we can partition it into two subsets, namely S_1 and S_2, satisfying $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$. We assume that every vertex in S_1 has a different color than the other vertices in S_1, while every vertex in S_2 shares the same color with at least one vertex in S_1. Formally, for each $v \in S_1$ (resp. $u \in S_2$), it is true that $\{w \in S_1 \mid color_v(G) = color_w(G)\} = \emptyset$ (resp. $\{w \in S_1 \mid color_u(G) = color_w(G)\} \neq \emptyset$). Based on the definition of graph coloring, it is evident that if two vertices u and v in G have the same color (i.e., $color_v(G) = color_u(G)$), there exists no edge connecting them (i.e., $(u, v) \notin E$). Consequently, we can deduce that $|S_2| \leq k$, as having more than k vertices in S_2 would violate the definition of k-defective clique. Moreover, the number of distinct colors in G is denoted as ω, it follows that $|S_1| \leq \omega$. Thus, the size of the maximum k-defective clique in G is bounded by $\omega + k$, establishing this lemma. □

Lemma 3 highlights the importance of finding a smaller value for ω in order to achieve a better upper bound. However, it is worth noting that determining the smallest value of ω for graph coloring is a computationally challenging problem, known to be NP-hard [?]. Thus, many heuristic approaches have been extensively investigated to address graph coloring [?]. In this paper, we adopt a widely used degeneracy ordering to color the graphs. Specifically, the degeneracy ordering [?] is defined as follows.

Definition 4. Given a graph G with the vertex set V, the degeneracy ordering is a permutation $(v_1, v_2, ..., v_4)$ of vertices in V such that for each vertex v_i, its degree is smallest in the subgraph of G induced by $\{v_i, v_{i+1}, ..., v_n\}$.

The degeneracy ordering, similar to the method for computing the k-core of a graph, can be acquired using the peeling technique [?]. In this method, the vertex removal ordering aligns with the degeneracy ordering, and it can be executed with a time complexity of $O(m)$. Then, we can systematically assign colors to each vertex v_i of graph G in a reverse order of the degeneracy ordering, commencing from $i = n$ and concluding at $i = 1$. Consequently, the upper bound based on graph coloring can be computed efficiently in $O(m)$ time.

Denote by δ the maximum core number of G, i.e., δ is the maximum value of k such that there exist a non-empty k-core in G. When combining with Lemma 1-3, we then have the following relations.

Lemma 4. Given a graph G, let κ be the size of the maximum k-defective clique of G. We can derive that $\kappa \leq \omega + k \leq \delta + k + 1 \leq d_{max} + k + 1$, where d_{max} is the maximum degree of vertices in G.

Proof. It is evident that $d_{\max} \geq \delta$ and $d_{\max} \geq \omega - 1$ are clearly established. Thus, here we only prove the correctness of $\omega \leq \delta + 1$. Assume that the vertex ordering $\{v_1, v_2, ..., v_n\}$ corresponds to the degeneracy ordering. Based on this ordering, we can obtain that for each vertex v_i, the degree of v_i is the smallest among all vertices in the subgraph $G(v_i, v_{i+1}, ..., v_n)$ induced by the set $v_i, v_{i+1}, ..., v_n$, where $i \in [1, n]$. Denote by $d^+_{v_i}$ the degree of v_i in $G(\{v_i, v_{i+1}, ..., v_n\})$. It can be seen that the subgraph $G(\{v_i, v_{i+1}, ..., v_n\})$ also forms a $d^+_{v_i}$-core, implying that $d^+_{v_i} \leq \delta$. Moreover, once all vertices in $\{v_{i+1}, v_{i+2}, ..., v_n\}$ have been colored, when it comes to coloring the vertex v_i, we can see that the color number assigned to v_i is at most $d^+_{v_i} + 1$. Consequently, we establish $\omega \leq \delta + 1$. Hence, the lemma is proven. □

The following example illustrates the above mentioned upper bounds.

Example 1. ...

Improved color-based upper bound. We observe that the proposed technique for obtaining the upper bound is still not sufficiently tight. For example, let us consider a graph G depicted in Fig. ??, which has been colored using the degeneracy ordering. Given a subset of vertices $C = *******$ in G, it becomes apparent that the number of distinct colors utilized in the subgraph $G(C)$ is **. According to Lemma 3, we deduce that the upper bound for $\kappa(C)$ is *** when $k = 3$, which exceeds the size of the vertex set C. Furthermore, it is evident that the subgraph $G(C)$ itself is not a k-defective clique of G when $k = **$, implying that the upper bound for $\kappa(C)$ can be tightened to 5. To address this concern, we next develop a new color-based upper bound.

Given a vertex subset S of G, let $\kappa(S, C)$ be the size of the maximum k-defective clique in $G(S \cup C)$ that includes all vertices in S. We define $\overline{d}_v(S)$ as the number of non-neighbors of vertex v in S, given by $\overline{d}_v(S) = |S \setminus N_v(S)|$. Additionally, we define $c_v(S)$ as the number of other vertices in S that share the same color with vertex v, denoted as $c_v(S) = |\{u \in S \setminus \{v\} | color_v(G) = color_u(G)\}|$. With these definitions in place, we state the following lemma.

Lemma 5. Consider a graph G and a non-maximal k-defective clique S of G. If there exists a vertex set $D \in V \setminus S$ that can form a larger k-defective clique with S, then for each vertex $v \in D$, there are at least $\overline{d}_v(S) + c_v(D)$ non-neighbors of v in $G(S \cup D)$.

Proof. This lemma is clearly established since each vertex in $D \setminus \{v\}$ that has the same color as v is not the neighbor of v. □

Based on Lemma 4, we then have the following upper bound for $\kappa(S, C)$.

Lemma 6. Denote by $\overline{d}(S)$ the total number of missing edges in $G(S)$, i.e., $\overline{d}(S) = \frac{1}{2}\sum_{v \in S}(|S| - d_v(S) - 1)$. Given two sets S and C, let D be the largest subset of C satisfying $\sum_{v \in D}(\overline{d}_v(S) + \frac{1}{2}c_v(D))\}) \leq k - \overline{d}(S)$. When computing the maximum k-defective clique containing S in the subgraph $G(S \cup C)$, we can state that $\kappa(S, C) \leq |S| + |D|$.

Proof. Given the subset D of C, we observe that there are at least $\frac{1}{2}\sum_{v \in D} c_v(D)$ missing edges in $G(D)$, as any pair of vertices with the same color in D must not have an edge between them. Moreover, each vertex v in D has $\overline{d}_v(S)$ non-neighbors in S. Consequently, when the set D is incorporated into the current k-deficient clique S, there will be a minimum of $\sum_{v \in D}(\overline{d}_v(S) + \frac{1}{2}c_v(D)\})$ additional missing edges. Given that D is the largest subset of C satisfying $\sum_{v \in D}(\overline{d}_v(S) + \frac{1}{2}c_v(D)\}) \leq k - \overline{d}(S)$, we can conclude that the size of maximum k-defective clique containing S in the subgraph $G(S \cup C)$ is at most $|S| + |D|$. This completes the proof of the Lemma. □

Example 2. ...

Algorithm 2: Upperbound1(S, C, k)

/* Supposing that each vertex in $S \cup C$ is colored based on the degeneracy ordering */

1 $D \leftarrow \emptyset$; $s \leftarrow$ the missing edges in $G(S)$;

2 **while** $C \neq \emptyset$ **do**

3 $\quad v \leftarrow$ a vertex in C with minimum $\overline{d}_v(S) + c_v(D)$;

4 $\quad$ **if** $s + \overline{d}_v(S) + c_v(D) > k$ **then break**;

5 $\quad s \leftarrow s + \overline{d}_v(S) + c_v(D)$;

6 $\quad D \leftarrow D \cup \{v\}$; $C \leftarrow C \setminus \{v\}$;

7 **return** $|S| + |D|$;

Based on Lemma 6, we can derive an algorithm, as shown in Algorithm 2, to compute the upper bound of $\kappa(S, C)$. This algorithm employs a heuristic approach to determine the largest subset D of C. Initially, the algorithm sets D as an empty set (line 1). Then, it iteratively selects a vertex v in C with the smallest value of $\overline{d}_v(S) + c_v(D)$ and adds it to the current subset D (lines 2-6). Once v is selected, the missing edges in $G(S \cup D)$ increase by $\overline{d}_v(S) + c_v(D)$, and v is moved from C to D (lines 5-6). The algorithm terminates and outputs $|S| + |D|$ as the upper bound of $\kappa(S, C)$ when either C becomes empty or the missing edges in $G(S \cup D)$ would violate the definition of a k-defective clique (lines 2 and 4). The following Theorem presents the correctness of Algorithm 2.

Theorem 3.1. *Algorithm 2 correctly computes the upper bound of $\kappa(S, C)$.*

Proof. On the contrary, we assume that there exist a subset D' of C with $|D'| > |D|$ that satisfies $\sum_{v \in D'} (\overline{d}_v(S) + \frac{1}{2} c_v(D'))\}) \leq k - \overline{d}(S)$. Denote by $D_1 = D' \setminus D$. It is easy to see that there must exist two vertices $v \in D$ and $u \in D_1$ satisfying $\overline{d}_v(S) + c_v(D \setminus \{v\}) > \overline{d}_u(S) + c_u(D \setminus \{v\})$, or D is the largest result. If $color_v(G) = color_u(G)$, we obtain that $\overline{d}_v(S) > \overline{d}_u(S)$ and the vertex u must contain in D if $v \in D$ according to our algorithm. If $color_v(G) \neq color_u(G)$, we have $\overline{d}_v(S) + c_v(D) = \overline{d}_v(S) + c_v(D \cup \{u\})$ and the vertex u will be also prior to push into D compared to u. However, we note that $u \notin D$, which is contradictory. Thus, we proved this lemma. □

Theorem 3.2. *The time complexity of Algorithm 2 is bounded by $O(kn + \overline{m})$, where $\overline{m}$ is the number of missing edges in $G(C)$.*

Proof. Initially, each vertex v in C needs to be sorted with the size of $\overline{d}_v(S)$, which takes $O(kn)$ times using a bin sort. Moreover, when a vertex v in C is pushed into D, each vertex in $C \setminus D$ that have the same color with v, which takes at most $O(\overline{d}_v(C))$ time. Thus, the total time consumes of lines 2-6 in Algorithm 2 requires $O(\overline{m})$ time. Thus, this theorem is proved. □

Note that by preprocessing of lemma 1-3, the subgraph $G(S \cup C)$ is usually relatively dense for real-world graphs. This means that the missing edges in $G(C)$ cannot be very large. Thus, Algorithm 2 is very time efficiency in computing the upper bound of $\kappa(S, C)$, which is further evidenced by our practical experiments (in Sec. **??**).

Algorithm 3: A heuristic search by expanding

Input: The graph $G = (V, E)$ and a parameter $k \geq 0$

Output: A near maximum k-defective clique S^* in G

$v \leftarrow$ the vertex in V with the maximum core number; $S^* \leftarrow \{v\}$;

Expanding S^* with vertices in $N_v^2(G)$ in a degree ordering;

$G \leftarrow (|S^*| - s)$-core of G;

$V \leftarrow$ a total order of vertices in G (degree, degeneracy, and color);

for $i = n$ *to* 1 *s.t.* $core(v_i) \geq |S^*| - s$ **do**

$\quad S \leftarrow \{v_i\}$; $C_1 \leftarrow N_{v_i}^+(G)$; $C_2 \leftarrow \emptyset$;

$\quad$ Removing all vetices in C_1 with degree less than $|S^*| - 1 - s$ in $G(C_1)$;

$\quad$ **for** *each* $v_j \in N_{v_i}^{=2}(G)$ *s.t.* $j > i$ **do**

$\quad\quad$ **if** $d_{v_j}(C_1) \geq |S^*| - s$ **then**

$\quad\quad\quad C_2 \leftarrow C_2 \cup \{v_j\}$

$\quad$ **while** $S \cup C_1 \cup C_2$ *is not an* s*-defective clique* **do**

$\quad\quad v \leftarrow$ a vertex in $C_1 \cup C_2$ with maximum core number;

$\quad\quad S \leftarrow S \cup \{v\}$ and remove all vertices in $C_1 \cup C_2$ that cannot form a larger s-defective clique with $S \cup \{v\}$;

$\quad\quad$ **if** $\exists u \in C_1 \cup C_2$ *with* $d_u(S \cup C_1) < |S^*| - s$ **then**

$\quad\quad\quad$ Remove u from $C_1 \cup C_2$;

$\quad\quad$ **if** $\exists u \in S$ *with* $d_u(S \cup C_1) < |S^*| - s$ **then**

$\quad\quad\quad C_1 \leftarrow \emptyset$; $C_2 \leftarrow \emptyset$; **break**;

$\quad$ **if** $|S^*| < |S \cup C_1 \cup C_2|$ **then** $S^* \leftarrow S \cup C_1 \cup C_2$;

return S^*;

4 THE PROPOSED ALGORITHMS

In this section, we

Russian doll search.

Theorem 4.1. *Let $\{v_1, v_2, ..., v_n\}$ be an ordering of vertices in V. Assume that $|S^*|$ is the maximum size of the k-defective clique in $G(\{v_i, v_{i+1}, ..., v_n\})$. Then the maximum size of the k-defective clique in $G(\{v_{i-1}, v_i, v_{i+1}, ..., v_n\})$ is at most $|S^*| + 1$.*

Theorem 4.2 (New pivoting). *Let $\mathcal{B}(S, C)$ be the recursive call to find the maximum k-defective clique in $G(S \cup C)$. Let v be a vertex in C with $\overline{d}_v(S) \leq 1$, then the maximum k-defective clique either contains v or a vertex in $C \setminus N_v(G)$.*

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

Pruning branches.

Algorithm 4: Russian doll search

Input: The graph $G = (V, E)$ and a parameter k
Output: The maximum k-defective clique S^* of G

A heuristic search to compute a near maximum k-defective clique S^* of G;
$G \leftarrow (|S^*| - s)$-core of G;
$V \leftarrow$ a total order of vertices in G (degree, degeneracy, and color);
for $i = n$ *to* 1 *s.t.* $core(v_i) \geq |S^*| - s$ **do**
  if $|S^*| < k + 1$ **then** $Branch(\{v_i\}, V^+ \setminus \{v_i\}, true)$;
  else
    $S \leftarrow \{v_i\}; C_1 \leftarrow N^+_{v_i}(G); C_2 \leftarrow \emptyset$;
    Remove vetices in C_1 with degree less than $|S^*| - s$ in $G(C_1)$;
    for *each* $v_j \in N^{=2}_{v_i}(G)$ *s.t.* $j > i$ **do**
      if $d_{v_j}(C_1) > |S^*| - s$ **then** $C_2 \leftarrow C_2 \cup \{v_j\}$;
    Color vertices in $G(S \cup C_1 \cup C_2)$ (Recoloring $\tau = |S^*| - s + 1$);
    $Branch(S, C_1 \cup C_2, true)$;

Algorithm 5: $Branch(S, C, falg)$

1 **if** $falg = false$ **then return**;
2 **if** $S \cup C$ *is a* k*-defective clique* **then**
3   **if** $|S \cup C| > |S^*|$ **then** $S^* \leftarrow S \cup C; falg \leftarrow false$;
4   **return**;
5 **if** *upper bound* $\leq |S^*|$ **then return**;
6 $C_1 \leftarrow \{v \in C | \overline{d}_v(S) \leq 1\}; C_2 \leftarrow C \setminus C_1$;
7 **for** *each* $v \in C_1$ *s.t.* $\overline{d}_v(S \cup C) = 1$ **do**
8   $S \leftarrow S \cup \{v\}; C \leftarrow C \setminus \{v\}; C_1 \leftarrow C_1 \setminus \{v\}$;
9 **if** $\exists v \in C_1$ *s.t.* $\overline{d}_v(S \cup C) = 2$ **then**
10   $C \leftarrow \{u \in C | S \cup \{u, v\}$ is a k-defective clique$\}$;
11   $Branch(S \cup \{v\}, C \setminus \{v\}, falg)$;
12 **else if** $\exists v \in C_1$ *s.t.* $\overline{d}_v(S \cup C) = 3$ **then**
13   $Branch(S \cup \{v\}, C \setminus \{v\}, falg)$;
14   Denote by $D \leftarrow C \cap \overline{N}_v(G)$;
15   **if** $|D \cap C_1| = 2$ **then**
16   Let u, w be the vertices in $D \cap C_1$;
17   **if** $\overline{d}_u(S) + \overline{d}_w(S) = 0$ *and* $(u, w) \in E$ **then**
18       $Branch(S \cup \{u, w\}, C \setminus \{v\} \cap N(\{u, w\}), falg)$;
19   **else if** $\overline{d}_v(S) = 1$ *and* $|D \cap C_1| = 1$ **then**
20   Let u be the vertex in $D \cap C_1$;
21   **if** $\overline{d}_u(S) = 0$ **then**
22       $Branch(S \cup \{u\}, C \setminus \{v\} \cap N(\{u\}), falg)$;
23 **else if** $C_2 \neq \emptyset$ **then**
24   Let v be the vertex in C with maximum $\overline{d}_v(S)$;
25   $Branch(S \cup \{v\}, C \setminus \{v\}, falg)$;
26   $Branch(S, C \setminus \{v\}, falg)$;
27 **else**
28   Let v be the vertex in C_1 with maximum degree;
29   **for** *each* $u \in C \setminus N_v(G)$ *s.t.* $\overline{d}_v(S \cup C) > 3$ **do**
30     $C_1 \leftarrow C_1 \setminus \{u\}; C \leftarrow C \setminus \{u\}$;
31     $Branch(S \cup \{u\}, C, falg)$;
32   Run lines 12-22;

Theorem 4.3 ($\overline{d}_v(S \cup C) \leq 2$). *Denote by $\kappa(S, C)$ be the size of maximum k-defective clique in $G(S \cup C)$. If there exists a vertex $v \in C$ with $\overline{d}_v(S \cup C) \leq 2$, then $\kappa(S, C) \leq \kappa(S \cup \{v\}, C \setminus \{v\})$, where $\overline{d}_v(S \cup C) = |\overline{N}_v(S \cup C)|$ and $v \in \overline{N}_v(G)$.*

Theorem 4.4 ($\overline{d}_v(S \cup C) = 3$). *Denote by $\kappa(S, C)$ be the size of maximum k-defective clique in $G(S \cup C)$. Let v be a vertex in C with $\overline{d}_v(S \cup C) = 3$ and $\overline{d}_v(S) \leq 1$. Denote by $u, w \in N_v(S \cup C)$. Then, we have the following results:*

(1) If $u \in C$ with $\overline{d}_u(S) \geq 1$ (or $w \in C$ with $\overline{d}_w(S) \geq 1$), $\kappa(S, C) \leq \kappa(S \cup \{v\}, C \setminus \{v\})$.

(2) If $\overline{d}_v(S) = 1$ and $u \in C$, $\kappa(S, C) \leq \max\{\kappa(S \cup \{v\}, C \setminus \{v\}), \kappa(S \cup \{u\}, C \setminus \{v\} \cap N_u(G))\}$.

(3) If $u, w \in C$ with $(u, w) \notin E$, $\kappa(S, C) \leq \kappa(S \cup \{v\}, C \setminus \{v\})$.

(4) If $u, w \in C$ with $(u, w) \in E$, $\kappa(S, C) \leq \max\{\kappa(S \cup \{v\}, C \setminus \{v\}), \kappa(S \cup \{u\}, C \setminus \{v\} \cap N_u(G) \cap N_w(G))\}$.

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

Theorem 4.5. *The time complexity of the algorithm is bounded by $O(m\gamma_k^n)$, where γ_k is the maximum real root of $x^{k+2} - \sum_{i=1}^{k} x^{k+2-i} - 1 = 0$.*

Proof. Key idea is to prove a recurrence of $T(n) = \sum_{i=1}^{k} T(n - i) + T(n - k - 2)$, where $k \geq 1$. ◻

.

5 EXPERIMENTS

5.1 Experimental Setup

5.2 Experimental Results

6 RELATED WORKS

7 CONCLUSION

REFERENCES

[1] 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.

Algorithm 6: Branch and reduction search

Input: The graph $G = (V, E)$ and a parameter k
Output: The maximum k-defective clique S^* of G

A heuristic search to compute a near maximum k-defective clique S^* of G;
$G \leftarrow (|S^*| - s)$-core of G;
$V \leftarrow$ a total order of vertices in G (degree, degeneracy, and color);
while $|V| > |S^*|$ **do**
    $v \leftarrow$ a vertex in V with the minimum degree;
    if $|S^*| < k + 1$ **then** $Branch(\{v\}, V \setminus \{v\})$;
    else
        if $d_v(G) \leq |S^*| - s$ *or* $cn(N_v(G)) < |S^*| - s$ **then**
            Remove v from V and G; **continue**;
        $S \leftarrow \{v_i\}; C_1 \leftarrow N_v(G); C_2 \leftarrow \emptyset$;
        Remove vetices in C_1 with degree less than $|S^*| - s$ in $G(C_1)$;
        for *each* $u \in N_v^{=2}(G)$ **do**
            if $d_u(C_1) > |S^*| - s$ **then** $C_2 \leftarrow C_2 \cup \{u\}$;
        Color vertices in $G(S \cup C_1 \cup C_2)$ (Recoloring $\tau = |S^*| - s + 1$);
        $Branch(S, C)$;
    Remove v from V and G;

Table 1: Real-world graph datasets.

Datasets	n	m	$d_{\max}$	δ

[2] Qiangqiang Dai, Rong-Hua Li, Meihao Liao, and Guoren Wang. 2023. Maximal Defective Clique Enumeration. *Proc. ACM Manag. Data* 1, 1 (2023), 77:1–77:26.

[3] Jian Gao, Zhenghang Xu, Ruizhi Li, and Minghao Yin. 2022. An Exact Algorithm with New Upper Bounds for the Maximum k-Defective Clique Problem in Massive Sparse Graphs. In *AAAI*. 10174–10183.

[4] Timo Gschwind, Stefan Irnich, and Isabel Podlinski. 2018. Maximum weight relaxed cliques and Russian Doll Search revisited. *Discret. Appl. Math.* 234 (2018), 131–138.

[5] Ailsa H. Land and Alison G. Doig. 2010. An Automatic Method for Solving Discrete Programming Problems. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. 105–132.

[6] 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.

[7] Gérard Verfaillie, Michel Lemaître, and Thomas Schiex. 1996. Russian Doll Search for Solving Constraint Optimization Problems. In *AAAI*. 181–187.

[8] Mihalis Yannakakis. 1978. Node- and Edge-Deletion NP-Complete Problems. In *STOC*. 253–264.

[9] 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.