KD-Club: An Efficient Exact Algorithm with New Coloring-Based Upper Bound for the Maximum k-Defective Clique Problem

Mingming Jin*, Jiongzhi Zheng*, Kun He†

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
brooklet60@hust.edu.cn

Abstract

The Maximum k-Defective Clique Problem (MDCP) aims to find a maximum k-defective clique in a given graph, where a k-defective clique is a relaxation clique missing at most k edges. MDCP is NP-hard and finds many real-world applications in analyzing dense but not necessarily complete subgraphs. Exact algorithms for MDCP mainly follow the Branch-and-bound (BnB) framework, whose performance heavily depends on the quality of the upper bound on the cardinality of a maximum k-defective clique. The state-of-the-art BnB MDCP algorithms calculate the upper bound quickly but conservatively as they ignore many possible missing edges. In this paper, we propose a novel CoLoring-based Upper Bound (CLUB) that uses graph coloring techniques to detect independent sets so as to detect missing edges ignored by the previous methods. We then develop a new BnB algorithm for MDCP, called KD-Club, using CLUB in both the preprocessing stage for graph reduction and the BnB searching process for branch pruning. Extensive experiments show that KD-Club significantly outperforms state-of-the-art BnB MDCP algorithms on the number of solved instances within the cut-off time, having much smaller search tree and shorter solving time on various benchmarks.

Introduction

Investigating structured subgraphs is a practical task with numerous demands in many optimization problems and real-world applications. The clique model is a famous and well-studied structured subgraph where any two distinct vertices are restricted to be adjacent. However, in many real-world applications, such as biological networks (Yu et al. 2006), social networks (Balasundaram, Butenko, and Hicks 2011), and community detection (Conte et al. 2018; Yang et al. 2022), dense subgraphs need not be complete but allow missing a few connections. Thus, many relaxations clique structures, such as the quasi-clique (Brunato, Hoos, and Battiti 2007), k-plex (Balasundaram, Butenko, and Hicks 2011), k-defective clique (Yu et al. 2006), etc., have been proposed.

Among these relaxations, the k-defective clique allows missing at most k edges over a clique, and the k-plex allows missing at most $k-1$ edges for each vertex. Obviously, the relaxation of the k-defective clique is between the clique and the k-plex, *i.e.*, a clique must be a k-defective clique, and a k-defective clique must be a $(k+1)$-plex. The clique problem has been well studied in the past decades. Recently, the k-plex also attracted much attention (Chang, Xu, and Strash 2022; Wang et al. 2022; Zheng et al. 2023; Jiang et al. 2023; Wang et al. 2023), yet there are relatively few studies on the k-defective clique (Chen et al. 2021; Gao et al. 2022; Dai et al. 2023) which also has wide applications, such as social network analysis (Jain and Seshadhri 2020), transportation (Sherali and Smith 2006), clustering (Yin et al. 2017), and protein interaction prediction (Yu et al. 2006). In this paper, we address the Maximum k-Defective Clique Problem (MDCP), which aims to find the maximum k-defective clique in a given graph, and we focus on its exact solving.

Since the k-defective clique is a relaxation of the clique, the difficulty of solving MDCP is as hard as finding the maximum clique in a given graph, which is a famous NP-hard problem. Some exact algorithms for MDCP have been proposed, coming up with a series of efficient techniques, such as reduction rules and upper bounds. For representative methods, there are the generic RDS algorithm (Verfaillie, Lemaître, and Schiex 1996; Gschwind, Irnich, and Podlinski 2018) and a branch-and-price framework (Gschwind et al. 2021) for various relaxed clique problems, including MDCP, and the most recent MADEC$^+$ (Chen et al. 2021) and KDBB (Gao et al. 2022) algorithms that proposed some new upper bounds and reduction rules. As the state-of-the-art exact algorithms for MDCP, both MADEC$^+$ and KDBB follow the branch-and-bound (BnB) framework (McCreesh, Prosser, and Trimble 2017; Jiang et al. 2018).

A BnB algorithm for MDCP usually maintains a growing partial solution S (*i.e.*, a k-defective clique) and the corresponding candidate vertex set C. Reduction or pruning is performed if the upper bound of the size of the maximum k-defective clique containing S is no larger than the size of the maximum k-defective clique found so far (*i.e.*, lower bound). Therefore, the quality of the upper bound greatly influences the algorithm's efficiency. MADEC$^+$ proposes an upper bound based on graph coloring. Note that an independent set in a graph is a vertex set where any two distinct vertices are non-adjacent. MADEC$^+$ uses graph coloring methods to assign each vertex color with the constraint that adjacent vertices cannot be in the same color, so as to

*These authors contributed equally.
†Corresponding author.

partition the entire subgraph of G induced by $V\backslash S$ into independent sets and then calculates the upper bound. Such an upper bound regards S and $V\backslash S$ as entirely independent and increases significantly with the increase of k.

KDBB proposes some new upper bounds that focus on the missing edges between candidate vertices and vertices in S, which increase softly with the increase of k. Thus, KDBB shows excellent performance for MDCP instances with large values of k. The upper bounds in KDBB also lead to efficient reduction rules, helping KDBB significantly reduce massive sparse graphs. However, these upper bounds are still not very tight since they ignore the missing edges between candidate vertices. In other words, they regard the entire candidate set C as a clique, which is over-conservative.

To address the above issues, we propose a new upper bound based on graph coloring, called **CoLoring-based Upper Bound (CLUB)**, that considers the missing edges between not only vertices in C and vertices in S but also vertices in C themselves. The main idea is that for a subset C' of the candidate set C, we use the graph coloring method to partition C' into r independent sets. Then, when we want to add t vertices (suppose $r < t \leq |C'|$) from C' to S, besides the missing edges between the t vertices and vertices in S that might exist, there must be missing edges between the t added vertices since there must be added vertices belonging to the same independent set. Since CLUB considers the missing edges more thoroughly, it is strictly no worse than the upper bounds proposed in KDBB.

Our proposed CLUB is a generic approach that can be used not only during the BnB searching process to prune the branches but also in preprocessing for graph reduction. Based on CLUB, we propose a new BnB algorithm for MDCP, called **KD-Club** (maximum **K**-**D**efective clique algorithm with **CLUB**). Since the upper bound in MADEC$^+$ increases significantly with the increase of k, MADEC$^+$ does not work well for MDCP instances with large k values. Since the reduction rules in KDBB cannot help it reduce dense graphs, KDBB does not work well for MDCP instances based on dense graphs. Our proposed CLUB significantly outperforms the previous upper bounds for MDCP by considering the connectivity between vertices more thoroughly, helping the BnB algorithm significantly reduce the search space. Therefore, KD-Club has excellent performance and robustness for solving MDCP instances based on either massive sparse or dense graphs, with either small or large k values, as shown in our experiments.

Preliminaries

Definitions

Given an undirected graph $G = (V, E)$, where V is the vertex set and E the edge set, the density of G is $2|E|/(|V|(|V|-1))$. We define $\overline{G} = (V, \overline{E})$ as the complementary graph of G, *i.e.*, $\overline{E} = \{(u, v)|u \in V \wedge v \in V \wedge (u, v) \notin E\}$.

We refer to adjacent vertices as neighbors to each other and denote the set of neighbors to v in G as $N_G(v)$, $|N_G(v)|$ is the degree of v in G, and the common neighbor of two vertices u, v as $N_G(u, v) = N_G(u) \cap N_G(v)$. Moreover, we define $N_G[v] = N_G(v) \cup \{v\}$ and $N_G[u, v] = N_G(u, v) \cup \{u, v\}$. Given a vertex set $V' \subseteq V$, $G[V']$ is defined as the subgraph induced by V'. Given a positive integer k, $V' \subseteq V$ is a k-defective clique if $G[V']$ has at least $\binom{|V'|}{2} - k$ edges.

We use $S \subseteq V$ to denote a growing partial solution (*i.e.*, a growing k-defective clique) in BnB algorithms, and $C \subseteq V\backslash S$ as the corresponding candidate vertex set of S. The size of the maximum k-defective clique in G that includes all vertices in S is denoted by $\omega_{G,k}(S)$, and the size of the maximum k-defective clique in G is $\omega_{G,k}(\emptyset)$.

Revisiting the Reduction Rules of KDBB

The KDBB algorithm defines function $rem_{G,k}(S, v) = k - |E(\overline{G}[S \cup \{v\}])|$ as the remaining number of allowed missing edges of $S \cup \{v\}$ to form a feasible k-defective clique, and function $res_{G,k}(S, v) = \min\{rem_{G,k}(S, v), |C\backslash N_{G[C\cup\{v\}]}[v]|\}$ as the maximum number of extra vertices in C non-adjacent to v that are allowed to be added to $S \cup \{v\}$. KDBB proposes an upper bound of $\omega_{G,k}(S \cup \{v\})$ calculated as $UB_{G,k}(S, v) = |S \cup \{v\}| + |N_{G[C]}(v)| + res_{G,k}(S, v)$, which actually considers C as a clique and adding a vertex from $C\backslash N_{G[C\cup\{v\}]}[v]$ to S only leads to one more missing edge.

Similarly, KDBB defines function $rem_{G,k}(S, u, v) = k - |E(\overline{G}[S \cup \{u, v\}])|$ as the remaining number of allowed missing edges of $S \cup \{u, v\}$ to form a feasible k-defective clique, and function $res_{G,k}(S, u, v) = \min\{rem_{G,k}(S, u, v), |C\backslash N_{G[C\cup\{u,v\}]}[u, v]|\}$ as the maximum number of extra vertices in C non-adjacent to u or v that are allowed to be added to $S \cup \{u, v\}$. And KDBB proposes another upper bound of $\omega_{G,k}(S \cup \{u, v\})$ calculated as $UB_{G,k}(S, u, v) = |S \cup \{u, v\}| + |N_{G[C]}(u, v)| + res_{G,k}(S, u, v)$, regarding C as a clique also.

Given a lower bound LB of $\omega_{G,k}(\emptyset)$, KDBB proposes the following two reduction rules based on the above bounds.

Rule 1. Remove vertex v from G if it satisfies $UB_{G,k}(\emptyset, v) \leq LB$.

Rule 2. Remove edge (u, v) from G if it satisfies $UB_{G,k}(\emptyset, u, v) \leq LB$.

Coloring-based Upper Bound

This section introduces our proposed upper bound, CLUB. We first introduce the main idea and definition of CLUB, then present an example for illustration. In the end, we introduce our ColorBound algorithm to calculate the CLUB.

Main Idea

Suppose $\{C_0, \cdots, C_k\}$ is a partition of C *w.r.t.* S, where C_i is the set of vertices with i non-adjacent vertices in S, *i.e.*, $C_i = \{v|v \in C \wedge |N_G(v) \cap S| = |S| - i\}$. In other words, adding each vertex $v \in C_i$ to S leads to i missing edges between v and vertices in S. Then, we can summarize the following important lemma which implies our main idea.

Lemma 1. *Suppose C_i can be partitioned into r_i independent sets $\{I_{i,1}, \cdots, I_{i,r_i}\}$, adding any $1 \leq t \leq |C_i|$ vertices in C_i to S leads to at least $\mathcal{N}(t) = c \times \frac{d(d+1)}{2} + (r_i -$*

$c) \times \frac{d(d-1)}{2}$ *more missing edges between the* t *added vertices, where* $d = \lfloor t/r_i \rfloor$ *and* $c = t - r_i d$.

Proof. Suppose the t vertices contain d_j vertices in $I_{i,j}$, then the number of missing edges between the t vertices is at least $\sum_{j=1}^{r_i} \frac{d_j(d_j-1)}{2} = \frac{1}{2}\left(\sum_{j=1}^{r_i} d_j^2 - t\right)$. Obviously, to make the above lower bound as small as possible, every d_j is expected to be t/r_i. Since d_j must be an integer, every d_j should be either $\lceil t/r_i \rceil$ or $\lfloor t/r_i \rfloor$ to minimize the lower bound. In this case, there are $t\%r_i$ independent sets containing $\lceil t/r_i \rceil$ vertices in t and the rest of the $r_i - t\%r_i$ independent sets contain $\lfloor t/r_i \rfloor$ vertices in t, which results in the lower bound in Lemma 1. □

Lemma 1 provides a lower bound, *i.e.*, $\mathcal{N}(t)$, of the number of increased missing edges, with which we can calculate an upper bound of the number of vertices that C can provide for S, resulting in the proposed CLUB.

Lemma 1 also indicates that the smaller the value of r_i, the more missing edges by adding vertices in C_i to S. KDBB actually regards $r_i = |C_i|$ since it regards C as a clique for the bound calculation, while our CLUB uses graph coloring methods to partition C_i and determine the value of r_i. So CLUB is strictly no worse than the upper bounds in KDBB.

Definition of CLUB

For each set C_i that can be partitioned into r_i independent sets, we randomly re-partition C_i into $m_i = \lceil |C_i|/r_i \rceil$ disjoint subsets $\{I'_{i,1}, \cdots, I'_{i,m_i}\}$. If r_i divides $|C_i|$, each of the m_i sets contains r_i vertices in C_i. Otherwise, I'_{i,m_i} contains $|C_i|\%r_i$ vertices in C_i and each of the remaining $m_i - 1$ sets contains r_i vertices in C_i. We assume that vertices in $I'_{i,j}$ $(1 < j \leq m_i)$ can be added to S only when all vertices in $\cup_{l=1}^{j-1} I'_{i,l}$ have been added to S.

According to Lemma 1, when we add r_i vertices in $I'_{i,1}$ to S, we have $\mathcal{N}(1) = \cdots = \mathcal{N}(r_i) = 0$, which may only occur when the r_i vertices belong to different independent sets. When we further add r_i vertices in $I'_{i,2}$ to S, we have $\mathcal{N}(r_i + 1) - \mathcal{N}(r_i) = \mathcal{N}(r_i + 2) - \mathcal{N}(r_i + 1) = \cdots = \mathcal{N}(2r_i) - \mathcal{N}(2r_i - 1) = 1$. Similarly, we have $\mathcal{N}(jr_i + 1) - \mathcal{N}(jr_i) = \mathcal{N}(jr_i + 2) - \mathcal{N}(jr_i + 1) = \cdots = \mathcal{N}((j + 1)r_i) - \mathcal{N}((j+1)r_i - 1) = j$. In summary, adding any vertex in $I'_{i,j+1}$ to S leads to at least $i + j$ more missing edges.

Suppose $P_l = \cup_{j-1+i=l} I'_{i,j}$, then adding each vertex $v \in P_l$ to S leads to l more missing edges. We define function $Sub(v) = l | v \in P_l$ as the subscript of the subset that $v \in C$ belongs to, and define an ordered set of C as $ord(C) = \{v_1, \cdots, v_{|C|}\}$ such that for any pair of v_i and v_j, we have $Sub(v_i) \leq Sub(v_j)$ if $i < j$. With such an ordered set, we can calculate a lower bound on the number of increased missing edges caused by adding any t vertices of C to S, which is defined as $LB_{inc}(ord(C), t) = \sum_{i=1}^{t} Sub(v_i)$.

Finally, we define our CLUB of $\omega_{G,k}(S)$ as follows.

$$CLUB_{G,k}(S) = |S| + \max\{i | 0 \leq i \leq |C| \wedge LB_{inc}(ord(C), i) \leq k - |E(\overline{G}[S])|\}. \quad (1)$$

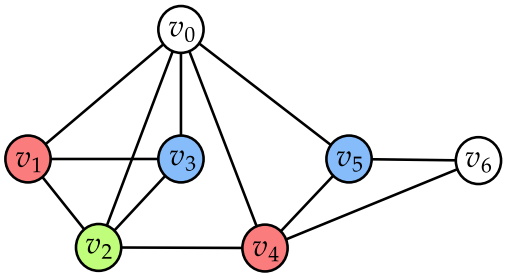


Figure 1: An example for the upper bound calculation.

An Example for Illustration

In this subsection, we provide an example to show how the upper bound in KDBB and CLUB are calculated. Figure 1 illustrates a growing partial 1-defective clique $S = \{v_0\}$ and its candidate set $C = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, which can be partitioned into $C_0 = \{v_1, v_2, v_3, v_4, v_5\}$ and $C_1 = \{v_6\}$. Suppose the current lower bound LB of $\omega_{G,1}(\emptyset)$ is 6.

KDBB regards C_0 and C_1 as cliques, *i.e.*, adding all vertices in C_0 to S does not increase any missing edges, and adding each vertex in C_1 to S leads to one more missing edge. Thus, KDBB considers the upper bound of $\omega_{G,1}(S) = |S| + 6 = 7 > LB$, and the current branch cannot be pruned, *i.e.*, v_0 will not be removed from the graph.

In CLUB, C_0 and C_1 are partitioned into 3 and 1 independent sets, respectively (*e.g.*, $I_{0,1} = \{v_1, v_4\}, I_{0,2} = \{v_2\}, I_{0,3} = \{v_3, v_5\}$, and $I_{1,1} = \{v_6\}$). Thus, we can re-partition C_0 into $I'_{0,1} = \{v_1, v_2, v_3\}, I'_{0,2} = \{v_4, v_5\}$, and C_1 into $I'_{1,1} = \{v_6\}$. And we have $P_0 = I'_{0,1} = \{v_1, v_2, v_3\}, P_1 = I'_{0,2} \cup I'_{1,1} = \{v_4, v_5, v_6\}$. According to Eq. 1, $CLUB_{G,1}(S) = |S| + 4 = 5 < LB$, and the current branch can be pruned, *i.e.*, v_0 will be removed.

The ColorBound Algorithm

We propose the ColorBound algorithm for practically calculating the proposed CLUB, as summarized in Algorithm 1. The algorithm first uses $|S|$ to initialize the upper bound UB (line 1), then uses the Extract function to extract subsets $\{P_0, \cdots, P_k\}$ from the candidate set C (line 3), such that adding each vertex in P_l to S leads to at least l more missing edges when adding the vertices sequentially according to $ord(C)$. Note that subsets P_l with $l > k$ are ignored since there are at most k missing edges in a k-defective clique. After that, CLUB is calculated by simulating the process of adding vertices sequentially until the number of allowed missing edges cannot afford one more vertex or all the candidate vertices have been added (lines 4-9).

Function Extract is summarized in Algorithm 2, which first partitions C into subsets $\{C_0, \cdots, C_k\}$ according to the number of missing edges between each vertex and the vertices in S and initializes each set P_i to $\emptyset$ (lines 1-3). Then, for each subset $C_i \neq \emptyset$, the algorithm sequentially colors each vertex in C_i with the minimum feasible index of color, satisfying that adjacent vertices cannot be in the same color, and obtains the value of r_i (lines 4-6). Finally, the algorithm

Algorithm 1: ColorBound(G, k, S, C)

Input: a graph G, an integer k, the current k-defective clique S, the candidate set C
Output: $CLUB_{G,k}(S)$

1 initialize the upper bound $UB \leftarrow |S|$;
2 initialize the number of allowed missing edges $\kappa \leftarrow k - |E(\overline{G}[S])|$;
3 $\{P_0, \cdots, P_k\} \leftarrow$Extract$(G, k, S, C)$;
4 $UB \leftarrow UB + |P_0|$;
5 **for** $i \leftarrow 1 : k$ **do**
6 **if** $i \times |P_i| \leq \kappa$ **then**
7 $UB \leftarrow UB + |P_i|$, $\kappa \leftarrow \kappa - i \times |P_i|$;
8 **else**
9 $UB \leftarrow UB + \lfloor \kappa / i \rfloor$, **break**;
10 **return** UB;

Algorithm 2: Extract(G, k, S, C)

Input: a graph G, an integer k, the current k-defective clique S, the candidate set C
Output: subsets $\{P_0, \cdots, P_k\}$ of C

1 **for** $i \leftarrow 0 : k$ **do**
2 $C_i \leftarrow \{v | v \in C \wedge |N_G(v) \cap S| = |S| - i\}$;
3 initialize $P_i \leftarrow \emptyset$;
4 **for** $i \leftarrow 0 : k$ **do**
5 **if** $C_i \neq \emptyset$ **then**
6 partition C_i into r_i independent sets by sequentially coloring the vertices;
7 $j \leftarrow 1$;
8 **while** $|C_i| \geq r_i \wedge j - 1 + i \leq k$ **do**
9 $I'_{i,j} \leftarrow$ set of r_i vertices in C_i;
10 $C_i \leftarrow C_i \backslash I'_{i,j}$, $P_{j-1+i} \leftarrow P_{j-1+i} \cup I'_{i,j}$;
11 $j \leftarrow j + 1$;
12 **if** $j - 1 + i \leq k$ **then** $P_{j-1+i} \leftarrow P_{j-1+i} \cup C_i$;
13 **return** $\{P_0, \cdots, P_k\}$;

iteratively moves r_i vertices, *i.e.*, set $I'_{i,j}$, from C_i to P_{j-1+i} until C_i is empty or $j - 1 + i > k$ (lines 8-12), which means adding any vertex in $I'_{i,j}$ to S according to $ord(C)$ leads to more than k missing edges.

The time complexities of both Algorithms 1 and 2 are dominated by the graph coloring process, which needs to traverse each candidate vertex and its neighbors. Thus, their time complexities are both $O(D|C|^2)$, where D is the maximum degree of vertices in G.

Branch and Bound Algorithm

We propose a new BnB algorithm for MDCP, called KD-Club, where a new preprocessing method based on CLUB is introduced to reduce the graph, and the CLUB is also used to prune branches during the BnB process. We first present the main framework of KD-Club, and then introduce the preprocessing method and the BnB process, respectively.

Algorithm 3: KD-Club (G, k)

Input: a graph G, an integer k
Output: $\omega_{G,k}(\emptyset)$

1 initialize LB by a heuristic method *FastLB*;
2 $G \leftarrow$ Preprocessing(G, k);
3 $LB \leftarrow$ BnB$(G, k, \emptyset, null, LB)$;
4 **return** LB;

General Framework

The framework of KD-Club is summarized in Algorithm 3. KD-Club maintains a lower bound LB of the size of the maximum k-defective clique in the input graph G, which is initialized by a method called *FastLB* (line 1), which is also used in the MADEC$^+$ (Chen et al. 2021) and KDBB (Gao et al. 2022) algorithms for calculating an initial lower bound. After calculating LB, the Preprocessing() function is called to reduce the graph (line 2), and the reduced graph is sent to the BnB() function to find the maximum k-defective clique (line 3). During the BnB process, LB will be updated once a larger k-defective clique is found. After the BnB() function traverses the entire search tree, we have $LB = \omega_{G,k}(\emptyset)$.

Preprocessing Method

Preprocessing plays an essential role in solving massive sparse instances. Given a lower bound LB of $\omega_{G,k}(\emptyset)$, we propose two new rules to reduce the graph based on CLUB.

Rule 3. Remove vertex v from G if it satisfies $CLUB_{G,k}(\{v\}) \leq LB$.

Rule 4. Remove edge (u, v) from G if it satisfies $CLUB_{G,k}(\{u, v\}) \leq LB$.

Algorithm 4 shows the detailed procedure of the Preprocessing method, where four functions with Rules 1-4 are used to reduce the input graph G. The functions $check\_vertex\_with\_Rule1$ and $check\_edge\_with\_Rule2$ are derived from KDBB, which uses Rules 1 and 2 to remove vertices and edges from G, respectively. Similarly, the functions $check\_vertex\_with\_Rule3$ and $check\_edge\_with\_Rule4$ use our proposed Rules 3 and 4 to remove vertices and edges from G.

Since Rules 1 and 2 ignore the connectivity between vertices in the candidate set C (we have $C = V \backslash \{v\}$ when trying to remove vertex v), they are computationally efficient for removing vertices with small degrees or edges whose endpoints have small degrees. Hence, our Preprocessing method uses Rules 1 and 2 to quickly remove vertices and edges that are *easy* to remove before using our Rules 3 and 4 to further reduce the graph until no more vertices and edges can be removed. Similarly, before calculating our CLUB to reduce the graph, we also try to use Rules 1 and 2 to achieve the current reduction and save computation time (lines 14 and 22). In other words, CLUB is used only when the vertex or edge cannot be removed by Rules 1 and 2.

Branch and Bound Process

The BnB process is depicted in Algorithm 5. The algorithm first calls function $reduction()$ used in KDBB (Gao et al.

Algorithm 4: Preprocessing(G, k)

Input: a graph G, an integer k
Output: the reduced graph G

1 $G \leftarrow check\_vertex\_with\_Rule1(G, k, LB)$;
2 $G \leftarrow check\_vertex\_with\_Rule3(G, k, LB)$;
3 $G \leftarrow check\_edge\_with\_Rule2(G, k, LB)$;
4 **while** *true* **do**
5 $G' \leftarrow check\_vertex\_with\_Rule3(G, k, LB)$;
6 $G' \leftarrow check\_edge\_with\_Rule4(G', k, LB)$;
7 **if** G' *and* G *is the same* **then break**;
8 **else** $G \leftarrow G'$;
9 **return** G;
10 Function $check\_vertex\_with\_Rule3(G, k, LB)$
11 $Q \leftarrow \emptyset$, add all vertices in V to Q;
12 **while** Q *is not empty* **do**
13 $v \leftarrow pop(Q)$, $N_v \leftarrow N_G(v)$;
14 apply Rules 1 and 3 on v with LB and k;
15 **if** v *is removed* **then**
16 add all vertices in N_v to Q;
17 **return** G;
18 Function $check\_edge\_with\_Rule4(G, k, LB)$
19 $Q \leftarrow \emptyset$; add all edges in E to Q;
20 **while** Q *is not empty* **do**
21 $(u, v) \leftarrow pop(Q)$;
22 apply Rules 2 and 4 on (u, v) with LB and k;
23 **if** (u, v) *is removed* **then**
24 add all edges adjacent to u or v in E to Q;
25 **return** G;

2022) to reduce the input graph G according to the current solution S (line 1), which actually removes each vertex $v \in V \backslash S$ (resp. edge $(u, v) \in E$) if $S \cup \{v\}$ (resp. $S \cup \{u, v\}$) is not a feasible solution. After which, we have the candidate set $C = V \backslash S$ (line 3). Then, the algorithm calculates CLUB and checks whether it is larger than the current lower bound (lines 4-5). If so, the algorithm will continue to search the subtree. Otherwise, the branch will be pruned.

For the selection of the branching vertex, we select the vertex in C with the minimum degree (ties are broken randomly) to find larger LB quickly and reduce the tree size for the subsequent searching. After selecting a branching vertex u, the algorithm uses a binary branching strategy, that is, either adding u to S or removing it from G (lines 7-11).

Empirical Evaluation

In this section, we first introduce the benchmarks and algorithms (also called solvers) used in the experiments, then present and analyze the experimental results. All the algorithms were implemented in C++ and run on a server using an AMD EPYC 7H12 CPU, running Ubuntu 18.04 Linux operation system. Since our machine is about 5-10 times faster than the machine in (Gao et al. 2022), which sets the cut-off time to 10,800 seconds for each instance, we set the cut-off time to 1,800 seconds in our experiments.

Algorithm 5: BnB(G, k, S, v, LB)

Input: a graph G, an integer k, the current k-defective clique S, the branching vertex v, the lower bound LB
Output: $\omega_{G,k}(S)$

1 **if** $v \neq null$ **then** $G \leftarrow reduction(G, k, S, v, LB)$;
2 **if** $|V| \leq LB$ **then return** LB;
3 $C \leftarrow V \backslash S$;
4 $ub \leftarrow$ ColorBound(G, k, S, C);
5 **if** $ub > LB$ **then**
6 select a vertex u in C with the minimum degree;
7 $size \leftarrow$ BnB$(G, k, S \cup \{u\}, u, LB)$;
8 **if** $size > LB$ **then** $LB \leftarrow size$;
9 remove u from G;
10 $size \leftarrow$ BnB(G, k, S, u, LB);
11 **if** $size > LB$ **then** $LB \leftarrow size$;
12 **return** LB;

Benchmark Datasets

We evaluated the algorithms on four public datasets that are widely used in existing works.

- **Facebook**[1]: This dataset contains 114 massive sparse graphs derived from Facebook social networks, which is used in KDBB (Gao et al. 2022).
- **Realword**[2]: This dataset contains 139 massive sparse graphs from the Network Data Repository (Rossi and Ahmed 2015), which is frequently used in studies related to relaxation clique models, including the k-defective clique and k-plex.
- **SNAP**[3] **and DIMACS10**[4]: This dataset contains 39 graphs with up to 1.05×10^6 vertices from the Stanford large network dataset collection (SNAP) and the 10th DIMACS implementation challenge, which are used in both KDBB and MADEC$^+$ (Chen et al. 2021).
- **DIMACS2**[5]: This dataset contains 49 dense graphs with up to 1,500 vertices from the 2nd DIMACS implementation challenge, which is used in MADEC$^+$.

For each graph, we generated six MDCP instances with $k = 1, 3, 5, 10, 15, 20$ as KDBB did. Therefore, there are a total of $6 \times (114 + 139 + 39 + 49) = 2046$ MDCP instances.

Solvers

To evaluate the performance of our proposed KD-Club algorithm, we select the state-of-the-art BnB MDCP algorithms, MADEC$^+$ (Chen et al. 2021) and KDBB (Gao et al. 2022) as the baseline algorithms. To evaluate the effectiveness of CLUB in different stages of KD-Club, we generate

[1] https://networkrepository.com/socfb.php
[2] http://lcs.ios.ac.cn/%7Ecaisw/Resource/realworld%20graphs.tar.gz
[3] http://snap.stanford.edu/data/
[4] https://www.cc.gatech.edu/dimacs10/downloads.shtml
[5] http://archive.dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique/

k	Facebook			Realworld			SNAP and DIMACS10			DIMACS2		
	KD-Club	KDBB	MADEC$^+$	KD-Club	KDBB	MADEC$^+$	KD-Club	KDBB	MADEC$^+$	KD-Club	KDBB	MADEC$^+$
1	**112**	110	12	**130**	124	81	**39**	**39**	24	31	17	**32**
3	**112**	110	0	**125**	116	62	**38**	**38**	23	**27**	12	14
5	**111**	109	0	**121**	110	52	37	**38**	23	**25**	12	12
10	**109**	108	0	**106**	94	28	**34**	**34**	11	**19**	11	6
15	**108**	104	0	**94**	70	22	**30**	28	9	**13**	11	2
20	**104**	86	0	**85**	59	16	**27**	24	6	**12**	11	1

Table 1: Summarized results of KD-Club, KDBB, and MADEC$^+$ on four benchmarks. The best results appear in bold.

two variant algorithms of KD-Club, called KD-Club$^-_{\text{Pre}}$ and KD-Club$^-_{\text{BnB}}$. Details are as follows.

- **MADEC$^+$**: A BnB MDCP algorithm with a rough coloring-based upper bound that increases sharply with the increment of k. It shows good performance on instances based on dense DIMACS2 graphs with small k values. The source code is available at[6].
- **KDBB**: The best-performing BnB MDCP algorithm for instances based on massive sparse graphs and instances with large k values. Since its code has not been open source, we use our implemented version in the experiments, which shows better performance than the results reported in the literature.
- **KD-Club**: An implementation of our algorithm[7].
- **KD-Club$^-_{\text{Pre}}$**: A variant of KD-Club without the CLUB during preprocessing, *i.e.*, replacing the preprocessing in KD-Club with that in KDBB; Also a variant of KDBB with the BnB searching process in KD-Club.
- **KD-Club$^-_{\text{BnB}}$**: A variant of KD-Club without the CLUB during BnB searching, *i.e.*, replacing the BnB searching process in KD-Club with that in KDBB; Also a variant of KDBB with the preprocessing method in KD-Club.

Performance Comparison

We first compare KD-Club, KDBB, and MADEC$^+$ on all four benchmarks to evaluate the overall performance of these solvers. The results are summarized in Table 1, which presents the number of instances that can be solved within the cut-off time by each algorithm in solving instances of each benchmark grouped according to the k values.

From the results, one can see that KD-Club solves considerably more instances than the baselines on most groups of benchmark instances, especially on instances with larger k values. Since the upper bound in MADEC$^+$ increases significantly with the increment of k, it fails to solve most instances with $k > 5$. With the increment of k, the reduction rules in KDBB can hardly reduce the graph. Thus, its performance also declines significantly. With the benefit of our CLUB that considers the missing edges more thoroughly, the preprocessing and BnB stages in KD-Club are both very efficient, and thus KD-Club shows excellent performance on various benchmarks, even with large k values. Moreover, although the number of solved instances of KD-Club is not much larger than the baselines in solving instances with small k values, the running time of KD-Club to solve the instances is usually much shorter than that of the baselines, as indicated by the follow-up experiments.

We further present the detailed results of KD-Club and KDBB in solving 20 representative instances from four benchmarks with $k = 3$ and $k = 10$, as shown in Table 2. The results include the number of vertices (column $|V|$) and edges (column $|E|$) of each original graph, the number of vertices (column $|V'|$) and edges (column $|E'|$) of each graph after reducing by the preprocessing method of each algorithm, the running time in seconds (column *Time*) and the sizes of their entire search trees in 10^4 (column *Tree*) to solve the instances. The symbol 'NA' means the algorithm cannot solve the instance within the cut-off time.

The results show that when solving massive sparse graphs, such as the Facebook instances started with 'socfb', the preprocessing method based on CLUB can help KD-Club reduce the graph size to an order of magnitude smaller than the graph reduced by the preprocessing in KDBB, indicating a significant reduction. When solving dense graphs, such as the DIMACS2 instances C125-9, johnson8-4-4, and san200-0-7-1, the preprocessing cannot reduce any vertex or edge, while the BnB process based on CLUB can still help KD-Club solve these instances with much shorter running time as compared to the cut-off time, but KDBB cannot even solve them within the cut-off time. As a result, both the search tree sizes and running time of KD-Club are several orders of magnitude smaller than those of KDBB in solving these instances with either small or large k values.

Ablation Study

We then compare KD-Club with its two variants, KD-Club$^-_{\text{Pre}}$ and KD-Club$^-_{\text{BnB}}$, as well as KDBB to evaluate the effectiveness of CLUB in preprocessing and BnB searching stages. The results are shown in Figure 2, where we present the variation of the number of solved instances with $k = 3$ and $k = 10$ for each algorithm over all the 341 graphs during the running time (in seconds). To present more clearly, we omit the results of easy instances solved within 10 seconds for each algorithm.

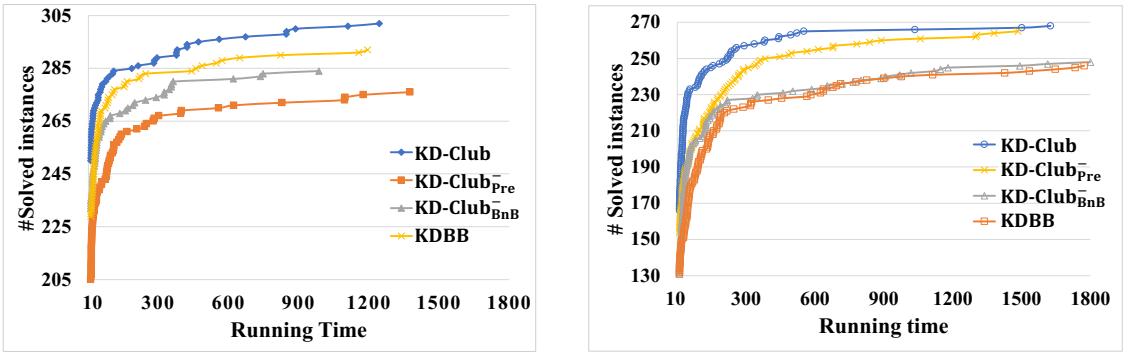
In general, KD-Club yields better performance than the two variants, and the two variants perform better than KDBB, indicating that using CLUB in both preprocessing and BnB stages can improve the performance of the BnB al-

[6]https://github.com/chenxiaoyu233/k-defective

[7]The source codes of KD-Club are available at https://github.com/JHL-HUST/KD-Club

Instance	$\|V\|$	$\|E\|$	$k=3$								$k=10$							
			KD-Club				KDBB				KD-Club				KDBB			
			$\|V'\|$	$\|E'\|$	Tree	Time	$\|V'\|$	$\|E'\|$	Tree	Time	$\|V'\|$	$\|E'\|$	Tree	Time	$\|V'\|$	$\|E'\|$	Tree	Time
C125-9	125	6963	125	6963	**35.8**	**23.8**	125	6963	NA	NA	125	6963	**536**	**137**	125	6963	NA	NA
gen200-p0-9-55	200	17910	200	17910	**45.7**	**102**	200	17910	NA	NA	200	17910	**325**	**275**	200	17910	NA	NA
ia-wiki-Talk	92117	360767	179	2244	**0.04**	**2.95**	1119	46194	NA	NA	2492	68463	**334**	**1420**	9424	155317	NA	NA
johnson8-4-4	70	1855	70	1855	**3.72**	**0.96**	70	1855	184	33.6	70	1855	**3436**	**339**	70	1855	NA	NA
MANN-a27	378	70551	378	70551	**102**	**460**	378	70551	NA	NA	378	70551	**100**	**469**	378	70551	NA	NA
san200-0-7-1	200	13930	200	13930	**0.87**	**2.79**	200	13930	NA	NA	200	13930	**36.9**	**34.2**	200	13930	NA	NA
Slashdot0811	77360	469180	138	5411	**0.14**	**0.65**	246	9888	65.7	80.9	210	7787	**15.3**	**10.0**	503	16319	428	75.8
soc-digg	770799	5907132	199	13551	**0.61**	**70.0**	5580	875610	NA	NA	519	37458	**29.3**	**214**	7349	1141673	NA	NA
socfb-Cornell5	18660	790777	211	6552	**0.06**	**4.45**	1292	52540	22.6	89.2	541	15848	**0.83**	**22.6**	2174	84381	44.7	107
socfb-Indiana	29732	1305757	136	4394	**0.04**	**7.92**	1596	57741	2.84	22.2	844	29290	**1.07**	**17.5**	2884	104681	16.8	152
socfb-OR	63392	816886	152	2901	**0.08**	**2.09**	1440	35123	5.21	12.0	621	14999	**2.65**	**23.5**	4910	117554	42.2	511
socfb-Penn94	41536	1362220	71	2228	**0.04**	**4.69**	875	23454	1.07	4.86	237	6564	**0.42**	**7.03**	2087	58201	2.86	34.3
socfb-Rice31	4087	184828	83	1601	**0.03**	**2.36**	1170	36409	10.9	16.9	921	25731	**2.34**	**35.2**	2161	87191	80.3	142
socfb-Texas84	36364	1590651	129	6606	**0.15**	**8.99**	927	41536	85.4	101	461	18985	**17.6**	**40.7**	1494	66131	601	399
socfb-UF	35111	1465654	299	10983	**0.06**	**15.2**	1612	75851	54.7	82.7	975	42128	**5.51**	**111**	2088	98361	237	159
socfb-UGA50	24389	1174057	125	6457	**0.10**	**7.87**	1276	55310	80.3	103	457	18956	**11.4**	**39.8**	2093	89760	472	286
socfb-Uillinois	30795	1264421	104	4828	**0.06**	**6.58**	586	25096	14.0	13.1	232	11219	**1.20**	**8.37**	1028	43222	76.2	27.8
socfb-Vassar85	3068	119161	80	1220	**0.02**	**0.65**	940	22547	0.55	3.36	496	9340	**1.40**	**18.3**	2591	99363	7.66	221
socfb-Yale4	8578	405450	140	3145	**0.03**	**2.62**	1033	28691	4.35	7.87	809	20248	**1.19**	**31.4**	2883	95160	31.4	161
Wiki-Vote	7115	100762	85	1602	**0.10**	**1.39**	902	34228	182	431	1378	40535	**40.4**	**171**	2233	72679	NA	NA

Table 2: Detailed results of KD-Club and KDBB on 20 representative MDCP instances from four benchmarks with $k = 3$ and $k = 10$. The search tree size is in 10^4, and the time is in seconds. The better results appear in bold.



(a) On instances with $k = 3$

(b) On instances with $k = 10$

Figure 2: Ablation study on MDCP instances over all the 341 graphs with different k values.

gorithm. Moreover, when solving instances with $k = 10$, the performance of KD-Club and KD-Club$^-_{Pre}$ is similar, and the performance of KD-Club$^-_{BnB}$ and KDBB is similar. This is because the larger the value of k, the smaller the size of the graph that can be reduced by preprocessing, and the worse the effectiveness of preprocessing. On the other hand, when solving instances with large k values, the BnB searching process based on CLUB also shows excellent performance, helping KD-Club significantly outperform KD-Club$^-_{BnB}$, and KD-Club$^-_{Pre}$ significantly outperform KDBB.

Conclusion

For the NP-hard Maximum k-Defective Clique Problem (MDCP), we proposed a novel CoLoring-based Upper Bound (CLUB) and then a new BnB algorithm called KD-Club. CLUB considers the missing edges that are ignored by previous methods, and uses graph coloring techniques in an ingenious way to calculate lower bounds on the number of missing edges, so as to obtain a tighter upper bound. By using CLUB on graph reduction and branch pruning, KD-Club significantly outperforms state-of-the-art BnB MDCP algorithms and exhibits excellent performance and robustness on instances based on either massive sparse or dense graphs, with either small or large k values.

In future work, we plan to apply our approach to improve the upper bounds used in BnB algorithms for other combinatorial optimization problems, such as other relaxation clique problems. In fact, existing BnB algorithms might ignore part of the relationship between elements in the problem to be solved, and could be greatly improved via a similar relationship detection method.

Acknowledgments

This work is supported by National Natural Science Foundation (U22B2017).

References

Balasundaram, B.; Butenko, S.; and Hicks, I. V. 2011. Clique Relaxations in Social Network Analysis: The Maximum k-Plex Problem. *Operations Research*, 59(1): 133–142.

Brunato, M.; Hoos, H. H.; and Battiti, R. 2007. On Effectively Finding Maximal Quasi-cliques in Graphs. In *Proceedings of the 2nd International Conference on Learning and Intelligent Optimization*, volume 5313, 41–55.

Chang, L.; Xu, M.; and Strash, D. 2022. Efficient Maximum k-Plex Computation over Large Sparse Graphs. *Proceedings of the VLDB Endowment*, 16(2): 127–139.

Chen, X.; Zhou, Y.; Hao, J.; and Xiao, M. 2021. Computing maximum k-defective cliques in massive graphs. *Computers & Operations Research*, 127: 105131.

Conte, A.; Matteis, T. D.; Sensi, D. D.; Grossi, R.; Marino, A.; and Versari, L. 2018. D2K: Scalable Community Detection in Massive Networks via Small-Diameter k-Plexes. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1272–1281.

Dai, Q.; Li, R.; Liao, M.; and Wang, G. 2023. Maximal Defective Clique Enumeration. *Proceedings of the ACM on Management of Data*, 1(1): 77:1–77:26.

Gao, J.; Xu, Z.; Li, R.; and Yin, M. 2022. An Exact Algorithm with New Upper Bounds for the Maximum k-Defective Clique Problem in Massive Sparse Graphs. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence, the 34th Conference on Innovative Applications of Artificial Intelligence, and the 12th Symposium on Educational Advances in Artificial Intelligence*, 10174–10183.

Gschwind, T.; Irnich, S.; Furini, F.; and Calvo, R. W. 2021. A Branch-and-Price Framework for Decomposing Graphs into Relaxed Cliques. *INFORMS Journal on Computing*, 33(3): 1070–1090.

Gschwind, T.; Irnich, S.; and Podlinski, I. 2018. Maximum weight relaxed cliques and Russian Doll Search revisited. *Discrete Applied Mathematics*, 234: 131–138.

Jain, S.; and Seshadhri, C. 2020. Provably and Efficiently Approximating Near-cliques using the Turán Shadow: PEANUTS. In *Proceedings of the Web Conference 2020*, 1966–1976.

Jiang, H.; Li, C.; Liu, Y.; and Manyà, F. 2018. A Two-Stage MaxSAT Reasoning Approach for the Maximum Weight Clique Problem. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, the 30th innovative Applications of Artificial Intelligence, and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, 1338–1346.

Jiang, H.; Xu, F.; Zheng, Z.; Wang, B.; and Zhou, W. 2023. A Refined Upper Bound and Inprocessing for the Maximum k-plex Problem. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*.

McCreesh, C.; Prosser, P.; and Trimble, J. 2017. A Partitioning Algorithm for Maximum Common Subgraph Problems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 712–719.

Rossi, R. A.; and Ahmed, N. K. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 4292–4293.

Sherali, H. D.; and Smith, J. C. 2006. A polyhedral study of the generalized vertex packing problem. *Mathematical Programming*, 107(3): 367–390.

Verfaillie, G.; Lemaître, M.; and Schiex, T. 1996. Russian Doll Search for Solving Constraint Optimization Problems. In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, 181–187.

Wang, Z.; Zhou, Y.; Luo, C.; and Xiao, M. 2023. A Fast Maximum k-Plex Algorithm Parameterized by the Degeneracy Gap. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*.

Wang, Z.; Zhou, Y.; Xiao, M.; and Khoussainov, B. 2022. Listing Maximal k-Plexes in Large Real-World Graphs. In *Proceedings of the 22th ACM Web Conference*, 1517–1527.

Yang, Y.; Shi, P.; Wang, Y.; and He, K. 2022. Quadratic Optimization based Clique Expansion for overlapping community detection. *Knowledge-Based Systems*, 247: 108760.

Yin, H.; Benson, A. R.; Leskovec, J.; and Gleich, D. F. 2017. Local Higher-Order Graph Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 555–564.

Yu, H.; Paccanaro, A.; Trifonov, V.; and Gerstein, M. 2006. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics*, 22(7): 823–829.

Zheng, J.; Jin, M.; Jin, Y.; and He, K. 2023. Relaxed Graph Color Bound for the Maximum k-plex Problem. *arXiv preprint arXiv:2301.07300*.