Listing Minimal Cores in Large Real-World Graphs

Yukai Sun[†], Kaiqiang Yu[‡], Shengxin Liu[†*], Cheng Long[$],
Raymond Chi-Wing Wong[§], Xun Zhou[†], Min Zhang[†]

[†]*Harbin Institute of Technology, Shenzhen*, [‡]*State Key Laboratory of Novel Software Technology, Nanjing University*,
[$]*Nanyang Technological University*, [§]*The Hong Kong University of Science and Technology*

{220810130@stu., sxliu@, zhouxun2023@, zhangmin2021@}hit.edu.cn,
kaiqiang.yu@nju.edu.cn, c.long@ntu.edu.sg, raywong@cse.ust.hk

***Abstract*—Cohesive subgraph mining is a fundamental task in graph data analytics. We re-visit the problem of listing all minimal k-cores, where a k-core is a subgraph in which every vertex has degree at least k, and minimality requires that no proper subset remains a k-core. Existing methods are computationally prohibitive due to explosive branching and costly branch state update, leading to the trivial worst-case bound $O^*(2^n)$ for the basic branch-and-bound baseline `WH`, where O^* suppresses polynomial factors and n is the number of vertices. In this paper, we present an improved method `IMinC` based on three key ideas: (i) a principled branching state with linear-time update; (ii) a pivot strategy that guides branching toward promising vertices; and (iii) a divide-and-conquer framework that initializes each subproblem to enable our pivot strategy throughout and reduce recursion depth. We further introduce three reduction rules that aggressively prune infeasible branches. Together, these components yield the worst-case time complexity of $O^*(\alpha_\ell^n)$, where α_ℓ is a positive number strictly smaller than 2. We also extend `IMinC` to list minimal k-cores under a size bound, addressing practical needs such as size-bounded community search. Extensive experiments on 12 real-world graphs demonstrate that `IMinC` outperforms the baselines by up to 2 order of magnitude, delivering substantial gains in efficiency.**

***Index Terms*—Cohesive subgraph mining, minimal k-cores**

I. Introduction

Graphs are a natural abstraction for relationships among entities in diverse applications, including social networks [56], [69], transaction networks [35], [33], and biological networks [2], [4]. A fundamental task in graph analysis is cohesive subgraph mining, which extracts densely connected subgraphs from large-scale graphs. This task has attracted substantial attention due to its broad applicability. For example, in social networks, cohesive subgraphs often correspond to tightly knit communities, revealing clustering patterns and group behaviors [28], [31], [43]. These structures are also informative for understanding network evolution and information diffusion, and many platforms leverage cohesive-subgraph-based techniques to discover and recommend communities. Similarly, in biological information networks, cohesive subgraphs often align with functionally related modules or complex pathways, e.g., protein complexes in protein–protein interaction networks and interdependent reaction pathways in metabolic networks [4], [8], [13]. Consequently, cohesive subgraphs are widely used for protein function prediction, disease module identification, and the study of cellular processes.

A variety of cohesive subgraph models have been studied, including cliques [12], [16], [57], [73], k-cores [6], [10], [17], and k-plexes [7], [20], [24]. Among them, k-core-based structures are particularly popular in applications such as community search [22], [5], [46], [61], [37]. Beyond their direct use, core-based structures have also accelerated other graph mining tasks, such as listing k-cliques [58] and finding the densest subgraph [29], [70]. We focus on the k-core model, where a k-core is formally defined as the maximal subgraph in which every vertex is adjacent to at least k vertices.

Existing studies and limitations. A central property of core structures is *nesting*: *every $(k+1)$-core is contained in the k-core*. Core decomposition therefore organizes a graph G into a hierarchy of nested cores. Most prior work focuses on the core decomposition problem, i.e., mining (maximal) k-cores [6], [10], [11], [17], [39], [44], [72], [66], [59], with advances in algorithmic efficiency (including parallel and distributed settings) and extensions to diverse graph types [10], [11], [17], [39], [72], [66], [59]. However, the nesting property often yields structures that are *large but only weakly cohesive* in practice. For example, in a k-core with n vertices, a vertex may still have up to $n-k$ non-neighbors including itself, and removing certain vertices can produce a denser subgraph that continues to satisfy the degree constraint. In these scenarios, k-core structures may offer limited insights due to their low cohesiveness in some applications.

Motivated by strengthening cohesiveness, we study *minimal* k-cores, which were first defined in [60]. A minimal k-core is the *minimal* subgraph where each vertex is adjacent to at least k vertices, and the removal of any vertex violates this property, i.e., at least one vertex in the remaining graph has fewer than k neighbors. We note that the minimum k-core represents only one specific instance (often the most constrained one), which inevitably fails to capture other equally important structures. To address this limitation, we propose to study the problem of enumerating all minimal k-cores, which has many real applications. For instance, in collaboration networks, a researcher often participates in multiple distinct communities (e.g., DB and AI), yet a minimum k-core search typically identifies only the smallest group, overlooking other significant collaborations, as shown in our case study (Section VI-C).

* Shengxin Liu is the corresponding author.

Similarly, in biological networks, a protein may participate in multiple functionally distinct complexes. Finding only the smallest complex fails to reveal biologically relevant modules and the protein's full functional spectrum, as shown in our case study (Section VI-D). For clarity, we use *maximal k-core* for the classic notion, and *k-core* more generally for any subgraph meeting the degree constraint; thus, a k-core may be minimal or maximal.

In this work, we revisit the problem of enumerating all minimal k-cores in a graph [60]. While minimal k-cores have been linked to neurobiological models of memory formation and recall [60], known enumeration methods scale poorly due to heavy branching and expensive minimality checks, which limits applicability on large graphs. Related work on the minimum k-core [37], [71] addresses a different objective, which finds a k-core of minimum cardinality. Although this solution is minimal, it captures only a narrow portion of the landscape and overlooks many informative k-cores (see our protein complex prediction case study in Section VI-D). Moreover, algorithms for the minimum k-core do not directly extend to solve our enumeration problem.

To establish a baseline, we first review the state-of-the-art branch-and-bound method of Wood and Hicks [60], denoted `WH`, in Section III. At a high level, `WH` recursively partitions the search space of vertex subsets into sub-spaces via branching. Each branch is represented as a triple (S, C, N), where S is the set of vertices retained in this branch that belong to a k-core, C is the set of vertices whose removal may lead to a minimal k-core, and N is the set of vertices that had already been processed and cannot be removed from S. To solve a branch, `WH` selects a minimum-degree vertex u from C as the branching vertex and creates two sub-branches by either discarding u from C or moving u from C to N. This branching process is repeated until C is empty. When initialized on G with an appropriate (S, C, N), this procedure exhaustively enumerates all minimal k-cores in G. However, `WH` suffers from efficiency bottlenecks. First, its worst-case time complexity is the trivial $O^*(2^n)$, where O^* suppresses the polynomials. Further, several design issues hinder scalability in practice: (i) the minimum-degree branching rule lacks theoretical justification for search-space reduction; (ii) maintaining the candidate set C requires $O(n^2)$ time per branch, incurring significant overhead; and (iii) when G contains a large maximal k-core, the initial sets S and C can be large, leading to deep recursion and excessive stack overhead.

Our improved method. We ask whether more principled branching can reduce the number of recursive calls and thereby improve both theoretical and empirical performance. We answer in the affirmative. We begin by redefining the branch state as a pair (S, C), where S is the partial set of vertices that must be included, and C is the candidate set of vertices that may be added to S to form larger sets. Solving (S, C) corresponds to enumerating all minimal k-core contained in $S \cup C$ that includes all vertices in S. Under this representation, maintaining the sets S and C can be done in $O(n)$ rather than $O(n^2)$ in `WH`. Then, we have the key observation that if removing certain branching vertices from C causes some vertex $v \in S$ to have fewer than k neighbors in $G[S \cup C]$, then the branch can be pruned, as every minimal k-cores found in this branch must include v. This motivates a targeted branching strategy that preferentially selects branching vertices among the neighbors of v within C to reduce the degree of v and trigger early pruning. We refer to this strategy of selecting branching vertices as the *pivot techniques*. Because the pivot technique relies on a single vertex in S, it does not directly apply to branches when S is empty. To address this, we introduce a divide-and-conquer framework that partitions the input into subproblems on smaller induced subgraphs. Each subproblem is solved by invoking branch-and-bound from a branch with a non-empty partial set S, thereby enabling systematic use of the pivot technique throughout the search. This decomposition also mitigates memory pressure by reducing recursion depth on instances with large maximal k-cores. Collectively, these components yield our improved algorithm `IMinC` (described in Section IV), which achieves the worst-case time complexity of $O^*(\alpha_\ell^n)$ with $\alpha_\ell < 2$ for different values of ℓ, improving upon the $O^*(2^n)$ bound of `WH`.

We further extend our `IMinC` to support size constraints by listing all minimal k-cores with at most θ vertices (Section V). This extension addresses practical needs in applications such as size-bounded community search [55], [22], [5], [46], [61], [40], [67], where resource limitations often impose size constraints on cohesive subgraphs. For instance, event planning on social networks may require cohesive groups of up to 10 attendees due to capacity limits. To boost performance, we further develop three reduction rules, namely **SCR 1-3**, that aggressively prune the search space. We also include a case study on protein complex prediction in Section VI-D to demonstrate the value of size constraints.

Finally, in Section VI, extensive experiments on 12 real-world datasets are conducted to show that our `IMinC` is up to 2 orders of magnitude faster than the baselines.

II. Preliminaries

In this paper, we consider an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, with $|V| = n$ and $|E| = m$. For a vertex $v \in V$, we denote by $N_G(v)$ the set of v's neighbors in G and denote by $d_G(v)$ the degree of v, i.e., $d_G(v) = |N_G(v)|$. Given any $S \subseteq V$, we use $G[S]$ to denote a subgraph of G induced by S, i.e., $G[S]$ includes the vertex set S and the edge set $\{(u, v) \in E \mid u, v \in S\}$. We note that all subgraphs considered in this paper are *vertex-induced*. For a subgraph g of G, we let $V(g)$ and $E(g)$ denote the sets of vertices and edges in g, respectively. We summarize frequently used notations in Table I.

We first introduce the concept of the maximal k-core, which has been widely studied in the literature [54], [10], [17].

Definition 1. *(Maximal k-core) A subgraph g is said to be a (maximal) k-core of G if and only if*

1) ***(Cohesiveness)** every vertex v has at least k neighbors in g, i.e., $d_g(v) \geq k$, and*

TABLE I: Notations frequently used in the paper

Symbol	Description
$G = (V, E)$	Graph with vertices V and edges E.
n, m	Number of vertices ($\|V\|$) and edges ($\|E\|$).
$N_G(v), d_G(v)$	Neighbor set and degree of v.
$G[S]$	Subgraph induced by $S \subseteq V$.
g	Subgraph with vertex/edge sets $V(g), E(g)$.
k	Minimum degree constraint.
δ, Δ	Degeneracy and max degree of G.

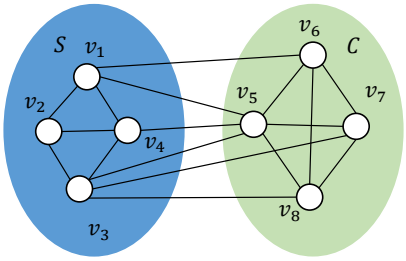


Fig. 1: The example graph used throughout the paper

2) ***(Maximality)*** *there is no subgraph g' containing g that also satisfies (1).*

The maximal k-core can be computed in $O(m)$ by iteratively peeling vertices with the smallest degree [6]. However, a (maximal) k-core g may contain a large number of vertices, where some vertices can have up to $|V(g)| - k$ non-neighbors due to the maximality requirement. Thus, the maximal k-core is less cohesive than other cohesive subgraph models in real applications [55], [5], [37], [47]. To address this, we consider the *minimal k-core*, which is defined in [60].

Definition 2. *(Minimal k-core) A subgraph g is said to be a minimal k-core of G if and only if*

1) ***(Cohesiveness)*** *every vertex v has at least k neighbors in g, i.e., $d_g(v) \geq k$, and*
2) ***(Minimality)*** *there is no subgraph $g' \subset g$ that satisfies (1).*

Clearly, a minimal k-core tends to include fewer vertices and is thus more cohesive than a k-core. Note that several studies [37], [47], [71] define the *minimum k-core* as the k-core with the fewest vertices in the graph, which is a minimal k-core. We also note that a graph has *at most one* maximal k-core (as multiple k-cores could be combined into a larger one by definition) while the number of minimal k-cores could be *exponential* [51]. To illustrate, in Figure 1, the entire graph is the maximal 3-core which includes at least two minimal 3-cores, i.e., $G[\{v_1, v_2, v_3, v_4, v_5\}]$ and $G[\{v_5, v_6, v_7, v_8\}]$.

Remark. In the literature, the maximal k-core is usually abbreviated to the k-core. To ease the presentation, we refer the k-core to a subgraph that satisfies the cohesiveness constraint only, i.e., $d_g(v) \geq k$. We will use *maximal* and *minimal* k-cores to distinguish two concepts throughout the paper.

We formulate the problem studied in this paper as below.

Problem 1. *(Minimal k-Core Enumeration [60]) Given a graph G and a positive integer k, the minimal k-core enumeration problem aims to list all minimal k-cores in G.*

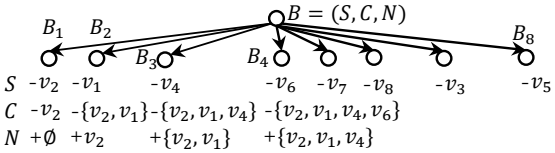


Fig. 2: Illustrating the branching process of WH

Hardness. The minimal k-core enumeration problem has been proven to be #P-hard and NP-hard [60]. This makes the enumeration of minimal k-cores significantly more challenging than finding (maximal) k-cores, a.k.a., core decomposition.

III. The State-of-the-Art Baseline: WH

In this section, we present the state-of-the-art branch-and-bound method proposed by Wood and Hicks [60], referred to as WH, which serves as our baseline algorithm.

Specifically, WH first prunes unnecessary vertices from the input graph G by maintaining the maximal k-core of G, since every k-core of G is a subgraph of the maximal k-core of G. Then, WH recursively partitions the search space (i.e., the set of all possible vertex subsets) into several subspaces via *branching*. In this recursive phase, each subspace is represented by a branch (S, C, N), where the *remaining set* S consists of vertices in the current branch that are part of a k-core, the *candidate set* C is the set of vertices that can be deleted to obtain a minimal k-core, and the *not-reduced set* N includes vertices that have already been processed and cannot be removed from S. A branch (S, C, N) holds those vertex sets (and their induced subgraphs) that 1) include $S \setminus C$ and 2) are subsets of S; thus, solving a branch (S, C, N) lists all minimal k-cores within (S, C, N). Thus, solving $(V(g_k), \{v \mid v \in V(g_k)$ and $G[V(g_k) \setminus \{v\}]$ is a k-core$\}, \emptyset)$, where g_k is the maximal k-core of G, solves the problem.

Branching. To solve a branch $B = (S, C, N)$, WH iteratively conducts the following operations until the candidate set C is empty: (1) It selects a vertex u from C having the minimum degree as the branching vertex. (2) Create a sub-branch by removing u from S. (3) Remove u from C and add it to N. Clearly, solving the formed sub-branches fully solves the branch (S, C). We call this branching strategy as *binary branching*. To illustrate, consider an example of finding minimal 3-cores in Figure 2 running on the graph in Figure 1, where $B_0 = (\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}, \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}, \emptyset)$.

Summary and analysis. We summarize WH in Algorithm 1. Specifically, the algorithm first obtains the maximal k-core of G via core decomposition in linear time $O(m)$ at Line 1. Then, we find the candidate set C at Line 2. It then recursively solves the problem by starting from the branch $(V(g_k), C, \emptyset)$ at Line 3. In particular, to solve a branch (S, C, N), WH_Rec terminates the branch when C becomes empty (Lines 5-8). To determine whether $G[S]$ is a minimal k-core, we remove each vertex $v \in N$ and verify whether the remaining graph

Algorithm 1: A basic method: WH [60]

Input: A graph G and a positive integer k
Output: All minimal k-cores in G

1 $g_k \leftarrow$ the maximal k-core of G;
2 $C \leftarrow \{v \mid v \in V(g_k)$ and $G[V(g_k) \setminus \{v\}]$ induces a k-core$\}$;
3 WH_Rec$(g_k, V(g_k), C, \emptyset, k)$;
4 **Procedure** WH_Rec(G, S, C, N, k)
/* Termination */
5 **if** $C = \emptyset$ **then**
6 **if** $G[S \setminus \{v_i\}]$ *does not induce a* k*-core* $\forall v_i \in N$ **then**
7 **Output** S;
8 **return**
/* Branching */
9 **while** $C \neq \emptyset$ **do**
10 Select a vertex u from C of smallest degree;
11 $C' \leftarrow \{v \mid v \in V(g_k) \setminus \{u\}$ and $G[V(g_k) \setminus \{u\} \setminus \{v\}]$ induces a k-core$\}$;
12 WH_Rec$(G, S \setminus \{u\}, C', N, k)$;
13 $C = C \setminus \{u\}$;
14 $N = N \cup \{u\}$;

still forms a k-core (Line 6). There is no need to examine vertices in $S \setminus N$, as they have already been confirmed not to form a k-core in Lines 2 and 11. We finally create sub-branches via binary branching (Lines 9-15). WH would explore $O(2^n)$ branches in the worst case. As a result, the worst-case time complexity of WH is $O^*(2^n)$, where $O^*(\cdot)$ suppresses the polynomials. We summarize the above analysis as follows.

Theorem 1. *Given a graph G and a positive integer k, WH lists all minimal k-cores in $O^*(2^n)$.*

Limitations. The state-of-the-art method WH has the following limitations which prevents its scalability to large graphs. First, the strategy of selecting a branching vertex with the minimum degree lacks rigorous theoretical support for its effectiveness in reducing the search space, thereby weakening the justification for this design choice in WH. Second, maintaining the candidate set C requires $O(n^2)$ time per branch, leading to considerable computational overhead. Third, WH is subject to memory constraints when the input graph contains a large maximal k-core, as the initial sets S and C can be very large, resulting in excessive stack space consumption during the recursion.

IV. Our Improved Method: IMinC

In this section, we introduce our improved algorithm called IMinC. Specifically, we first develop new pivot techniques for selecting the branching vertex within a branch that contains a non-empty partial set (Section IV-A). With the pivot techniques, fewer sub-branches are formed within each branch, as is theoretically stated in Theorem 2. To handle branches with an empty partial set (where the pivot techniques cannot be applied) and to reduce excessive recursion depth, we further propose a divide-and-conquer framework (Section IV-B), which divides the problem into multiple smaller sub-problems and each can be solved by invoking the branch-and-bound algorithm starting from a branch with a non-empty partial set. Finally, we summarize our IMinC and analyze its time complexity in Section IV-C. Specifically, IMinC has the time complexity of $O(n^2 \cdot m \cdot \alpha_\ell^n)$ with $\alpha_\ell < 2$, which achieves a better asymptotic performance than the state-of-the-art algorithm WH [60] (Section III).

A. IMinC: New Pivot Techniques

We first observe that WH selects the branching vertex of minimum degree without a clearly justified rationale for using binary branching. Naturally, this raises one question: *can we select a more specific branching vertex to improve theoretical and empirical performance by forming fewer branches?* Following previous studies on branch-and-bound algorithms [25], [15], [63], [12], we refer to the resulting technique as the *pivot technique*. We remark that pivot selection, i.e., selecting a vertex for branching based on a certain strategy to reduce the number of branches, is a well-established algorithmic framework widely adopted in the literature [25], [15], [63], [12]. However, the efficacy of pivot selection depends heavily on the problem structure; existing strategies designed for other problems fail to effectively reduce the search space when applied to our problem. Consequently, we propose a novel pivot strategy specifically tailored to efficiently list minimal k-cores. We elaborate on the details in the following.

Before introducing the pivot technique, we first reconstruct the set representation for each branch to eliminate the substantial time overhead associated with maintaining the vertices in C. Specifically, each subspace is now represented by a branch (S, C) rather than (S, C, N), where the *partial set* S denotes the set of vertices that must be included, and the *candidate set* C comprises vertices that may be added to S to form larger vertex sets. Solving a branch (S, C) lists all minimal k-cores contained within $S \cup C$ that include S. With this modification, the time complexity for maintaining the sets S and C is reduced to $O(n)$, as demonstrated in Algorithm 2. We first introduce our core reduction techniques as follows.

Core reduction. Consider a branch (S, C). We observe that all k-cores (of G) covered by (S, C) are also subgraphs of the maximal k-core of $G[S \cup C]$. This can be easily verified. Let g'_k be the maximal k-core of $G[S \cup C]$. Based on this observation, we can derive the following two reductions.

- **CR 1**. The current branch can be terminated if S contains vertices not in the maximal k-core g'_k, i.e., $S \subsetneq V(g'_k)$, or if no vertices in C belong to g'_k, i.e., $C \cap V(g'_k) = \emptyset$.
- **CR 2**. The candidate set C can be pruned by removing those vertices that are not in g'_k.

Now, consider the branching procedure starting at a branch $B = (S, C)$ with a non-empty partial set S, i.e., $S \neq \emptyset$. We can easily observe the following two facts about S.

- **Fact 1**. There exists a vertex v in S such that v is adjacent to less than k vertices in S, i.e., $\exists v \in S, d_{G[S]}(v) < k$.

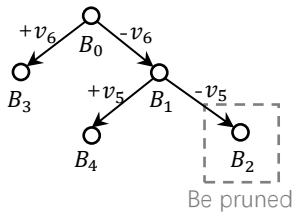


Fig. 3: Illustrating the pivot techniques

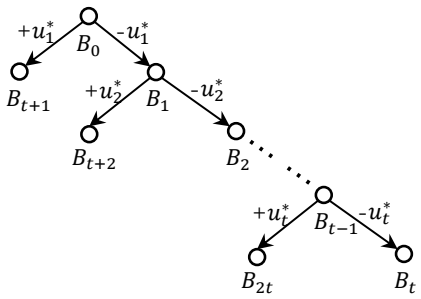


Fig. 4: The right-deep traversal starting from B_0

- **Fact 2**. Every vertex v in S is adjacent to at least k vertices in $S \cup C$, i.e., $\forall v \in S, d_{G[S\cup C]}(v) \geq k$.

Note that violating **Fact 1** implies that S induces a k-core, so the branch can be terminated in Line 4 of Algorithm 1. Besides, violating **Fact 2** implies that S contains a vertex v not in the maximal k-core of $G[S \cup C]$, allowing the branch to be pruned by **CR 1**.

We define $\Psi(B)$ as the subset of S where each vertex is adjacent to less than k vertices in S. Formally, we have

$$\Psi(B) = \{v \in S \mid d_{G[S]}(v) < k\}. \tag{1}$$

We note that $\Psi(B) \neq \emptyset$ according to **Fact 1**. Besides, we define $\mu(B)$ as the minimum degree of a vertex v in $\Psi(B)$ within $G[S \cup C]$:

$$\mu(B) = \min_{v\in\Psi(B)} d_{G[S\cup C]}(v). \tag{2}$$

Note that $\mu(B)$ is *well-defined* since $\Psi(B) \neq \emptyset$ as discussed above. Based on **Fact 2**, we can derive the following lemma.

Lemma 1. *Let B be a branch. If $\mu(B) < k$, branch B will be pruned by **CR 1** without any further branching or output.*

Consider the branching procedure at branch B and the sub-branch $B' = (S, C \setminus \{u\})$ formed by removing the branching vertex u. We have the following observation.

$$\mu(B) \geq \mu(B') \text{ where } B' = (S, C \setminus \{u\}). \tag{3}$$

This is because (1) $G[S\cup C\setminus\{u\}]$ is a subgraph of $G[S\cup C]$, and (2) B' shares the same partial set S as B, meaning $\Psi(B')$ is a subset of $\Psi(B)$. Motivated by the observation, our idea is to *select the branching vertex from C such that $\mu(B')$ is strictly smaller than $\mu(B)$, thereby increasing the likelihood that sub-branch B' would be pruned by Lemma 1*. To achieve this, we can easily derive the following strategy based on the definition of $\mu(\cdot)$.

Pivot selection strategy. Let $B = (S, C)$ be a branch with $S \neq \emptyset$ and v^* be the vertex with the smallest degree (within $G[S \cup C]$) among those vertices in $\Psi(B)$, i.e., $v^* = \arg\min_{v\in\Psi(B)} d_{G[S\cup C]}(v)$. We select from C one of the neighbors of v^* as the branching vertex u^*, called *pivot*. Formally, u^* is an arbitrary vertex in $C \cap N_{G[S\cup C]}(v^*)$.

For illustration, consider the example of listing minimal 3-cores in Figure 3, which is based on the example graph in Figure 1 with $B_0 = (\{v_1, v_2, v_3, v_4\}, \{v_5, v_6, v_7, v_8\})$. Here, v_6 is selected as the pivot since it is a neighbor of v_1, which has the smallest degree $d_{G[S_0]}(v_1) = 2$ in S_0. This results in $\mu(B_1) = 3 < \mu(B_0) = 4$. Similarly, v_5 is selected as the pivot, leading to B_2 being pruned since $\mu(B_2) = 2 \leq 3$. Compared to the example in Figure 2, the binary branching with our pivot techniques would form fewer branches.

Discussion and analysis. We note that the pivot u^* always exists for a branch with a non-empty partial set. We will discuss how to solve those branches with an empty partial set later in the following section. Besides, for the branching process at branch B where u^* is selected as the pivot and the sub-branch $B'' = (S, C \setminus \{u^*\})$ formed by removing the pivot u^*. We can easily derive that:

$$\mu(B) \geq \mu(B'') - 1 \text{ where } B'' = (S, C \setminus \{u^*\}). \tag{4}$$

Thus, the newly formed sub-branch can be pruned by Lemma 1 if $\mu(B'') < k$. Further, with our proposed pivot techniques, we form *theoretically fewer* sub-branches in the worst case, as shown in the following theorem (note that WH creates $O(2^{|C|})$ branches within B in the worst case).

Theorem 2. *Let $B = (S, C)$ be a branch with $S \neq \emptyset$. The binary branching with the proposed pivot selection strategy creates $O(\alpha_\ell^{|C|})$ branches within B in the worst case, where α_ℓ is the largest positive real root of $x^{\ell+2} - 2x^{\ell+1} + 1 = 0$ and $\ell = \mu(B) - k$. For example, when $\ell = 1$, 2, and 3, $\alpha_\ell = 1.6181$, 1.8393, and 1.9276, respectively.*

Proof. Consider the branching process starting at branch $B_0 = B$, as in Figure 4. Specifically, it corresponds to the right-deep traversal (starting from B_0 and ending at B_t for $t \geq 1$) in the recursion tree. Note that the right sub-branch of B_t formed by removing u_t^* is either pruned or terminated. Besides, vertex u_i^* $(1 \leq i \leq t)$ is the pivot selected by our proposed strategy at B_{i-1} (note that all branches within B_0 have non-empty partial sets). Based on Equation (4), we can derive

$$\mu(B_0) \geq \mu(B_1) - 1 \geq \mu(B_2) - 1 \geq ... \geq \mu(B_t) - 1. \tag{5}$$

Therefore, we have

$$\mu(B_i) \leq \mu(B) - i, 1 \leq i \leq t. \tag{6}$$

According to Lemma 1, we can prune B_i once $\mu(B_i) < k$. Hence, we have $\mu(B_i) \geq k$ for $1 \leq i \leq t$ (since otherwise B_t cannot be reached out in the traversal), and thus we derive

$$t \leq \mu(B) - k. \tag{7}$$

Let $T(|C|)$ be the number of branches within a branch B. Based on the above equation, we thus have

$$\begin{aligned} T(|C|) &= T(|C_{t+1}|) + T(|C_{t+2}|) + ... + T(|C_{2t+1}|) \\ &\leq T(|C|-1) + T(|C|-2) + ... + T(|C|-t-1) \\ &\leq T(|C|-1) + ... + T(|C|-\mu(B)+k-1). \quad (8) \end{aligned}$$

By solving the above linear recursion, $T(|C|)$ is bounded by $O(\alpha_\ell^{|C|})$ where α_ℓ is the largest positive real root of $x^{\ell+2} - 2x^{\ell+1} + 1 = 0$ and ℓ is $\mu(B) - k$. For example, when $\ell = 1$, 2, and 3, $\alpha_\ell = 1.6181$, 1.8393, and 1.9276, respectively. □

B. IMinC: Divide-and-Conquer Techniques

We note that the proposed pivot techniques are applicable only to branches with non-empty partial sets. However, the initial branch $(\emptyset, V(g_k))$ has an empty partial set, leading to many sub-branches with empty partial sets. Since the recursion depth is intrinsically bounded by the size of the initial maximal k-core g_k of G, deep recursion on large cores may incur excessive stack overhead. To solve these issues, we further incorporate a *divide-and-conquer* framework.

The idea is to divide the problem of listing minimal k-cores from the input graph G into n sub-problems, and each of them runs on a smaller subgraph $G_i = G[V_i]$ for $1 \leq i \leq n$. Formally, given an ordering $\langle v_1, v_2, \ldots, v_n \rangle$ of vertices in V, we have $V_i = \{v_i, ..., v_n\}$ for $1 \leq i \leq n$. On each subgraph $G[V_i]$, we run the proposed branch-and-bound method by starting with the branch (S, C) with $S = \{v_i\}$ and $C = V(g_{k,i}) \setminus \{v_i\}$ where $g_{k,i}$ is the maximal k-core of $G[V_i]$. Clearly, all minimal k-cores found in $G[V_i]$ must include $\{v_i\}$ and exclude $\{v_1, \ldots, v_{i-1}\}$. Thus, solving all of n sub-problems finds all minimal k-cores in G. We omit the formal proof of the divide-and-conquer framework, as it can be easily verified as above.

With the proposed techniques, all branches formed in the branch-and-bound method have the partial set non-empty since the initial branch for the i-th subproblem includes v_i in the partial set. Thus, the proposed pivot techniques can be applied to all branches. Moreover, we note that the divide-and-conquer framework can further improve efficiency and scalability when a size constraint is imposed on the found minimal k-cores (the details will be discussed in the following section).

Remark. We note that our divide-and-conquer strategy follows the principles of degeneracy-based vertex ordering, a technique widely used in existing literature [25], [15], [63]. Our method differs from previous ones in the strategy of refining each subgraph $G[V_i]$. Specifically, we propose to further refine each subgraph $G[V_i]$ to a smaller one, i.e., g_{k_i} if there is no size constraint, or G_i' otherwise (details are in Appendix C).

C. IMinC: Summary and Analysis

We summarize IMinC with the newly proposed techniques in Algorithm 2, which differs with the state-of-the-art algorithm WH [60] mainly in four aspects.

Algorithm 2: Our improved method: IMinC

Input: A graph $G = (V, E)$ and a positive integer k
Output: All minimal k-cores in G

```
   /* Divide-and-conquer techniques                    */
1  g_k ← the maximal k-core of G;
2  G ← g_k;
3  foreach vertex v_i in {v_1, v_2, ..., v_n} do
4      G_i ← G[{v_i, ..., v_n}];
5      g_{k,i} ← the maximal k-core of G_i;
6      IMinC_Rec(g_{k,i}, {v_i}, V(g_{k,i}) \ {v_i}, k);

7  Procedure IMinC_Rec(G, S, C, k)
       /* Termination and reductions                   */
8      if G[S] is a k-core then
9          if G[S \ {v_i}] does not induce a k-core
             ∀v_i ∈ S then
10             Output S;
11         return
12     if C = ∅ then return;
       /* Core Reductions by CR 1-2                    */
13     g'_k ← the maximal k-core of G[S ∪ C];
14     if S ⊊ V(g'_k) or C ∩ V(g'_k) = ∅ then
15         return;
16     C ← C ∩ V(g'_k);
       /* Branching with the pivot                     */
17     v* ← arg min_{v∈Ψ(B)} d_{G[S∪C](v)};
18     u* ← a vertex selected from N_{G[S∪C]}(v*) ∩ C;
19     IMinC_Rec(G, S ∪ {u*}, C \ {u*}, k);
20     IMinC_Rec(G, S, C \ {u*}, k);
```

First, as previously mentioned, each subspace is now represented by a branch (S, C) rather than (S, C, N) as in WH. As a result of omitting the *not-reduced set* N, we explicitly verify the minimality of the resulting k-core for each vertex in Line 9. Consequently, the time complexity for maintaining the sets S and C is reduced to $O(n)$, which significantly improves the processing time within each branch.

Second, IMinC incorporates the divide-and-conquer techniques (Lines 3-6). Specifically, IMinC performs n iterations (Lines 3-6). In the i-th ($1 \leq i \leq n$) iteration, we construct a smaller graph G_i, shrink G_i to the maximal k-core $g_{k,i}$, and finally run the branch-and-bound method IMinC_Rec by starting with the branch $(\{v_i\}, V(g_{k,i}) \setminus \{v_i\})$.

Third, to solve a branch, the branch-and-bound method IMinC_Rec adopts the pivot techniques for branching (Lines 9-12). Specifically, it selects the pivot u^* from C based on the proposed strategy (Lines 17-18) and uses u^* as the branching vertex for the binary branching (Lines 19-20).

Fourth, we utilize the core reduction technique by **CR 1** and **CR 2** to further narrow the search space (Lines 13–16). This approach ensures that the subgraph $G[S \cup C]$ forms a k-core, thereby eliminating irrelevant vertices.

Time complexity. We note that the worst-case time complexity of IMinC is bounded by invoking IMinC_Rec n times and the size of G_i is bounded by n and m. Besides, each

recursion of IMinC_Rec runs in polynomial time $O(nm)$. This time cost is dominated by checking the minimality of a k-core. Besides, we note that conducting **CR 1** and **CR 2** runs in $O(E(G[S \cup C]))$ which is bounded by $O(m)$, and selecting the pivot needs to obtain the degree information, i.e., $d_{G[S\cup C]}(\cdot)$ and $d_{G[S]}(\cdot)$, for each vertex in $S \cup C$, which runs in $O(\sum_{v\in S\cup C} d_{G[C\cup S]}(v))$ bounded by $O(m)$. Finally, we let Δ be the maximum degree of a vertex in G. We note that $\mu(B)$ is bounded by Δ for all branches based on the definition. According to Theorem 2, IMinC would form $O(\alpha_\ell^n)$ branches for each subproblem where $\ell = \Delta - k$. Moreover, as evidenced by Table III, the value of ℓ in most real-world graphs is less than 10, which is significantly smaller than $\Delta - k$. Thus, the time complexity of IMinC is $O(n^2 m \alpha_\ell^n)$, which is *smaller* than that of WH. We summarize the above analysis as follows.

Theorem 3. *Given a graph G and a positive integer k. IMinC finds all minimal k-cores within G in $O(n^2 m \alpha_\ell^n)$, where α_ℓ is the largest positive real root of $x^{\ell+2} - 2x^{\ell+1} + 1 = 0$ and ℓ is $\Delta - k$. For example, when $l = 1$, 2, and 3, $\alpha_\ell = 1.6181$, 1.8393, and 1.9276, respectively.*

Space complexity. We note that IMinC maintains two disjoint sets S and C during the recursion. Thus, the space complexity is $O(n+m)$, which is *linear* in the input size. We remark that the baseline WH has the same space complexity as IMinC.

Remark. We note that the delay (i.e., the time cost between two consecutive outputs) of IMinC can be exponential, similar to the baselines, as they all adopt the branch-and-bound framework to prioritize practical efficiency. Thus, deriving a tight output-sensitive complexity bound is theoretically challenging. This design choice stands in contrast to reverse-search-based methods [64], [19] that achieve polynomial delay but suffer from high overhead (see Appendix B of [1] for details).

V. Finding Minimal k-cores with Size Constraint

We observe that larger minimal k-cores tend to be less cohesive than other smaller ones. For example, one vertex in a minimal k-core g would have up to $V(g)-k-1$ non-neighbors excluding itself. Further, in some real-world scenarios, e.g., the size-bounded community search [55], [22], [5], [46], [61], [40], [67], one may naturally require the number of vertices within the found cohesive subgraphs, e.g., minimal k-cores, does not exceed a given threshold due to resource limitations. For example, in event planning on social networks, when organizing a party with up to 10 attendees due to space constraints, it is crucial to decide whom to invite to ensure that the group of attendees is as close as possible. To address it, we can find a minimal k-core with the size at least 10 from the social network. The corresponding user group within the found k-core would have high cohesiveness due to the properties of minimal k-cores and satisfy the space limitation. Motivated by this, we study an important variant of enumerating all minimal k-cores, as formulated below.

Problem 2 (Size Constraint Minimal k-Cores Enumeration)**.** *Given a graph G and two positive integers k and θ with $\theta \leq 2k+1$, the size constraint minimal k-cores enumeration problem aims to find all minimal k-cores containing at most θ vertices.*

We remark that the threshold θ is required to be at most $2k+1$ since a k-core with size no larger than $2k+1$ has the diameter at most 2. This ensures the cohesiveness of the k-cores and facilitates the development of efficient algorithms. Formally, we have the following lemma, where the proof is provided in Appendix A of [1].

Lemma 2. *The diameter of a k-core g with $|V(g)| \leq 2k+1$ is at most 2.*

We note that our proposed algorithms (WH and IMinC) can solve the problem by filtering out those minimal k-cores with the size larger than the threshold θ. To further boost the performance of IMinC for this variant, we develop new size-constraint-based reductions, namely **SCR 1-3**, which are integrated into IMinC.

SCR 1: diameter reduction with degeneracy ordering. Consider the i-th ($1 \leq i \leq n$) iteration of the divide-and-conquer process in IMinC, i.e., Lines 3-6 of Algorithm 2. We note that G_i at Line 4 can be further refined to a smaller subgraph G_i' as below.

$$G_i' = G[V_i'] \text{ where } V_i' = N^2_{G_i}(v_i), \tag{9}$$

where $N^2_{G_i}(v_i)$ is the set of 2-hop neighbors of v_i (including v_i) in G_i. This is because all k-cores found in G_i must include v_i and thus have the vertices within the 2-hop neighbors of v_i by Lemma 2.

Besides, we observe that the divide-and-conquer relies on the vertex ordering in G. Following existing studies [62], [15], we adopt the degeneracy ordering, which can be obtained by core decomposition in linear time $O(m)$ [6]. Thus, the size of G_i is bounded by the degeneracy $\delta\Delta$ of G, i.e., $|V_i| \leq \delta\Delta$. We thus call the resulting reduction *diameter-based reduction*.

SCR 2: 2-hop-based reduction. We first observe the following lemma based on the threshold θ, where the proof is provided in Appendix A of [1].

Lemma 3. *Given a graph G and two positive integers k and θ, two vertices u and v in G cannot appear in a k-core g with at most θ vertices if either of the following two conditions is satisfied.*

- $(u, v) \in E$ *and* $|N_G(u) \cap N_G(v)| < 2k - \theta$.
- $(u, v) \notin E$ *and* $|N_G(u) \cap N_G(v)| < 2k - \theta + 2$.

Based on Lemma 3, we can further refine G_i' in the i-th ($1 \leq i \leq n$) iteration of the divide-and-conquer technique with **SCR 1**. Specifically, we can remove from G_i any vertex v that satisfies either of the following conditions: (1) $(v_i, v) \in E$ and $|N_{G_i'}(v_i) \cap N_{G_i'}(v)| < 2k - \theta$, or (2) $(v_i, v) \notin E$ and $|N_{G_i'}(v_i) \cap N_{G_i'}(v)| < 2k - \theta + 2$. This is because all minimal k-cores found in G_i must include v_i, and, thus, cannot include vertices that violate the above conditions. We refer to the resulting reduction as the *2-hop-based reduction*.

SCR 3: lower-bound-based reduction. Consider a branch $B = (S, C)$. We can derive a lower bound, denoted as **LB**, of the size of a minimal k-core found in B. This means that all minimal k-cores found in B have a size of at least **LB**. Formally, we have:

Lemma 4. *Let $B = (S, C)$ be a branch. The lower bound **LB** of the size of a minimal k-core found in B is $k - \min_{v \in S} d_{G[S]}(v) + |S|$.*

The proof is provided in Appendix A of [1]. Clearly, if a branch has the lower bound LB larger than the threshold θ, the branch can be pruned since all minimal k-cores found in the branch are larger than θ. We call the resulting reduction *lower-bound-based reduction*. More details of the implementation of **SCR 1-3** are put in Appendix C of [1].

VI. Experimental Evaluations

Datasets. We use 12 real-world datasets downloaded from the SNAP website (http://snap.stanford.edu) and the Network Repository (https://networkrepository.com/index.php). The dataset statistics are summarized in Table II, where the density of a graph $G = (V, E)$ is defined as $2|E|/(|V| \cdot (|V| - 1))$, δ is the degeneracy, Δ is the maximum vertex degree, and k_d is the default value of k. In Table II, we also provide the statistics for avg-Density, which denotes the average density of the found minimal k_d-cores, and max-Density, which refers to the density of the maximal k_d-core. From the results, we observe that the maximal k_d-core is less cohesive compared to the minimal k_d-cores.

Algorithms. We compare our proposed `IMinC` (Algorithm 2) with the state-of-the-art baseline `WH` (Algorithm 1). We also compare against two adapted state-of-the-art baselines, namely `IBB` [71] and `Plex-Enum` [18]. `IBB` builds upon the minimum k-core algorithm in [71]. We adapt it by disabling the original size-based pruning rules to ensure the enumeration of all minimal k-cores rather than just the smallest one. Regarding `Plex-Enum`, which is based on the maximal s-plex enumeration algorithm [18], we leverage the property that a k-core of size θ is equivalent to a $(\theta - k)$-plex. Consequently, we adapt the search strategy to enumerate relevant candidates (forming a superset of k-cores) and subsequently filter out non-minimal ones. For ablation studies, we also compare different variants of our `IMinC`.

Setup. All algorithms are coded in C++ (g++ -O3). All experiments run on a Linux server equipped with an Intel CPU and 256GB memory. We set the time limit as 3,600 seconds and use **INF** to indicate timeouts. For ablation studies, we use 4 datasets, namely G2, G5, G7, and G8 as default ones (varying in graph scale). Our code is available at [1].

A. Comparison among Algorithms

All datasets (default settings). We compare our `IMinC` with three baselines, i.e., `WH`, `IBB`, and `Plex-Enum`, by using the settings k_d shown in Table II as the default value for k and setting $\theta = 2k_d + 1$. First, we report the running time in Figures 5 and 6 for listing all minimal k_d-cores without or with the size constraint $\theta_d = 2k_d + 1$, respectively, and report the number of minimal k_d-cores for each graph in Table II. We have the following observations. First, both algorithms run faster in Figure 6 when enumerating minimal k-cores with the size constraint $\theta = 2k_d + 1$. This demonstrates the effectiveness of the proposed reductions, i.e., **SCR 1-3**, based on the size constraint θ. Second, `IMinC` demonstrates superior performance across all datasets. It achieves speedups of up to five orders of magnitude over `WH` and `Plex-Enum`, and up to 554× over `IBB`. Furthermore, `IMinC` is the only algorithm that completes all datasets within the time limit. While baselines like `Plex-Enum` and `IBB` improve upon the solvability of `WH` (which handles only a few small graphs within one hour, i.e., G2 in the setting without the size constraint, and G2 and G4 in the setting with the size constraint), they remain significantly inferior to `IMinC` in computational efficiency. This indicates the superiority of the proposed pivot and divide-and-conquer techniques. Besides, the running time results also align with our theoretical analysis in Sections III and IV. Second, we evaluate the memory consumption across all datasets and observe that the memory usage of `IMinC` is comparable to that of the baselines (see Appendix D of [1] for details). Third, we report the empirical value of ℓ (measured by the maximum value of $\mu(B) - k$ during the recursion) and the gap between α_ℓ and 2 in Table II. Although the gap is small (implying marginal theoretical improvement), `IMinC` achieves significant practical speedups. This discrepancy is likely due to (1) The time complexity in Theorem 3 reflects the worst-case scenario, whereas real-world graphs (typically sparse) are not worst-case instances, allowing the pivot technique to prune branches effectively. (2) Other techniques, such as the divide-and-conquer framework with **SCR 1-3**, substantially reduce the search space in practice, even though they do not improve the worst-case theoretical bound. Fourth, we also report the statistical distribution of solution sizes in Appendix D of [1], which reveals distinct patterns for different graphs.

Varying k on default datasets. We compare `IMinC` with the baselines, i.e., `WH`, `IBB`, and `Plex-Enum`, when k varies, i.e., the value of k spanned from $k_d - 3$ to k_d, and maintain $\theta = 2k_d - 5$ to satisfy the constraint of all values of k. We report the running time in Figure 7. We have the following observations. First, `IMinC` significantly outperforms the baselines, achieving speedups of up to 2 orders of magnitude. This indicates the effectiveness of the proposed techniques across various settings of k. Second, while the runtime of all algorithms decreases as k increases, the decline in `IMinC` is more gradual. This is because the number of minimal k-cores decreases exponentially when k increases, yet `IMinC` remains computationally efficient for all tested values of k. Third, the achieved speedups increase when k increases, which indicates that `IMinC` performs better for larger k's. Some possible reasons include (1) the parameter ℓ which equals to $\Delta - k$ (in Theorem 2) decreases when k increases, resulting in fewer branches for larger k; and (2) the proposed **SCR 2** and **SCR 3** rely on k and the larger the value of k, the more vertices and branches can be pruned.

TABLE II: Statistics of Datasets

ID	Dataset	$\|V\|$	$\|E\|$	δ	Δ	Density	k_d	# Min. Cores	ℓ	avg-Density	max-Density	$2-\alpha_\ell$
G1	email-Eu-core	1,005	16,064	34	345	0.031841	33	2,430	39	0.666916	0.553846	9.1e-13
G2	lastfm	7,624	27,806	20	216	0.000957	19	53	27	0.708645	0.586939	3.7e-9
G3	ca-CondMat	23,133	93,439	25	279	0.000349	22	2,601	3	1.000000	0.491497	7.2e-2
G4	as-caida	26,475	53,381	22	2,628	0.000152	22	615	22	0.654860	0.530754	1.2e-7
G5	deezer	28,281	92,752	12	172	0.000232	11	467	13	0.719421	0.149791	6.1e-5
G6	soc-Slashdot	82,168	504,230	55	2,552	0.000149	55	111	26	0.711438	0.605544	7.5e-9
G7	yahoo-msg	100,058	739,815	25	9,374	0.000148	24	164	13	0.838894	0.406316	6.1e-5
G8	com-amazon	334,863	925,872	6	549	0.000017	6	87	3	0.875010	0.013655	7.2e-2
G9	as-skitter	1,696,415	11,095,298	112	35,455	0.000008	111	2,801	79	0.686403	0.678815	8.3e-25
G10	roadNet-CA	1,965,206	2,766,607	3	12	0.000001	3	403	1	0.792012	0.000726	3.8e-1
G11	soc-LiveJournal	4,847,571	42,851,237	373	20,333	0.000004	371	1,351	30	0.995090	0.990955	4.7e-10
G12	FullUSA	23,947,347	28,854,312	3	9	0.000000	3	427	1	0.816639	0.000859	3.8e-1

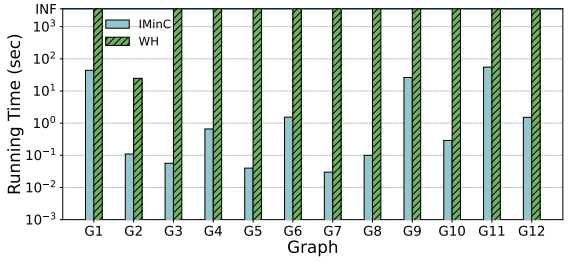


Fig. 5: Comparison on all datasets using k_d

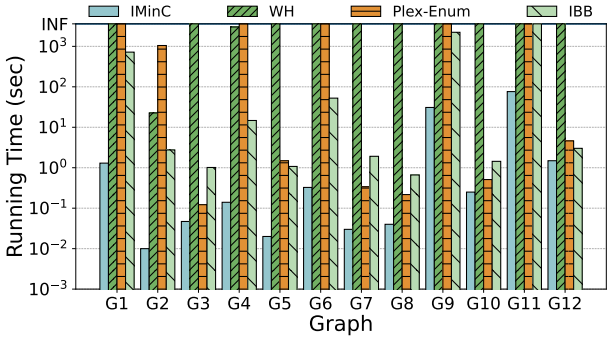
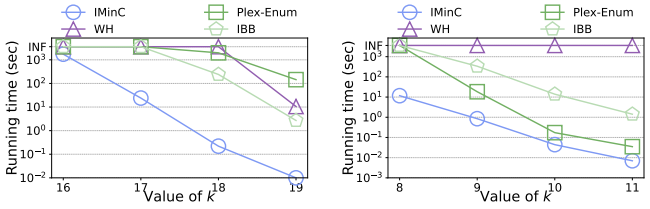
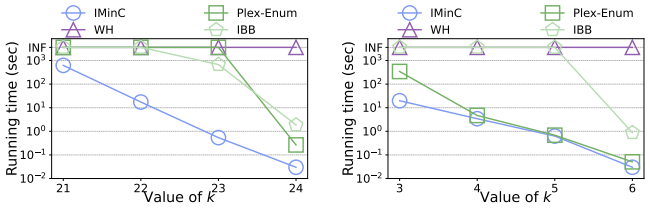


Fig. 6: Comparison on all datasets using k_d and $\theta_d = 2k_d + 1$



(a) Varying k (lastfm, G2) (b) Varying k (deezer, G5)



(c) Varying k (yahoo-msg, G7) (d) Varying k (com-amazon, G8)

Fig. 7: Comparison by varying k

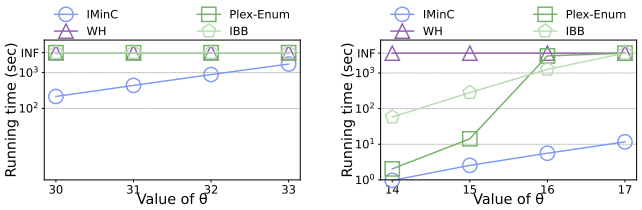
Varying θ on default datasets. We compare IMinC with the baselines, i.e., WH, IBB, and Plex-Enum, when θ varies. Specifically, we set $k = k_d - 3$ and vary θ over the range from $2k - 2$ to $2k + 1$. We report the running time in Figure 8 and have the following observations. WH fails to process all graphs across different values of θ, whereas IMinC significantly outperforms the baselines, achieving speedups of up to 307×, which indicates the efficiency of IMinC among various settings of θ. Second, the runtime of all algorithms increases as θ grows; however, the increase in IMinC is significantly more gradual compared to the baselines. This is mainly because the number of minimal k-cores satisfying the size constraint increases with θ, yet IMinC processes k-cores of varying sizes more efficiently. Third, the achieved speedups decrease when θ grows. One possible reason is that the proposed **SCR 2** and **SCR 3** rely on θ and prune fewer vertices or branches when θ increases.

B. Ablation Study

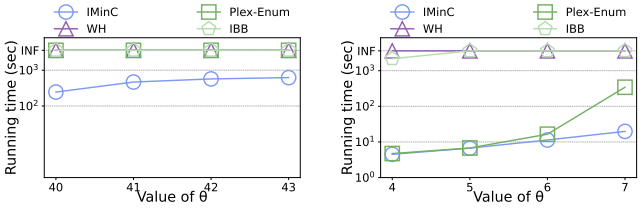
We next conduct the ablation study to investigate the effect of the proposed techniques. We also perform the scalability test in Appendix D of [1].

Effect of the pivot techniques. We compare IMinC with its variant IMinC\Piv: the full version of IMinC without the pivot techniques, which randomly selects the branching vertex for branching. We report the running time in Figure 9. We observe that IMinC runs faster than IMinC\Piv on settings for small values of k and large values of θ, achieving, on average, approximately 5× speedups. This demonstrates the superiority of the proposed pivot techniques. Besides, both algorithms have the running time increases when k decreases and/or θ grows. The achieved speedups narrow when k grows and θ increases. This probably because of the effect of the adopted reductions in this case.

Effect of the divide-and-conquer techniques and the reductions. We compare IMinC with its variant called IMinC\DAC: the full version of IMinC without the divide-and-conquer techniques and **SCR 1-2**. We report the running time in Figure 10. We observe that IMinC significantly outperforms IMinC\DAC for all settings of θ and k by

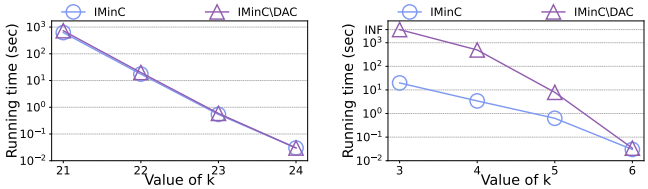


(a) Varying θ (lastfm, G2) (b) Varying θ (deezer, G5)

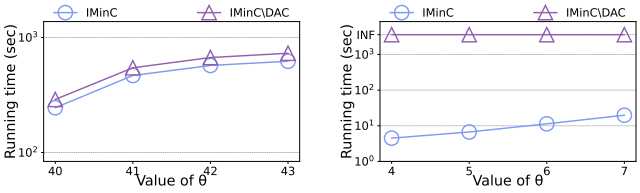


(c) Varying θ (yahoo-msg, G7) (d) Varying θ (com-amazon, G8)

Fig. 8: Comparison by varying θ

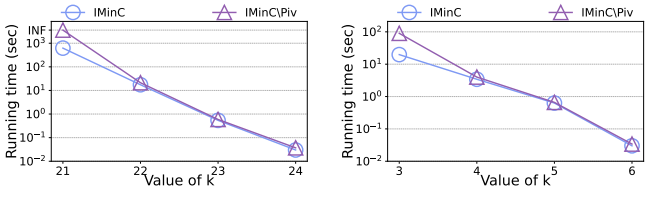


(a) Varying k (yahoo-msg, G7) (b) Varying k (com-amazon, G8)

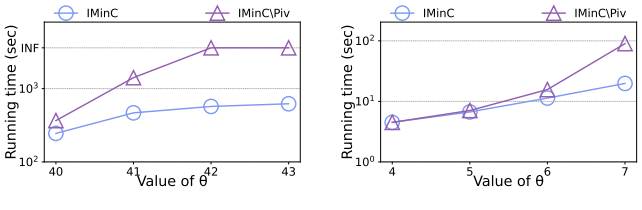


(c) Varying θ (yahoo-msg, G7) (d) Varying θ (com-amazon, G8)

Fig. 10: Comparison of divide-and-conquer frameworks



(a) Varying k (yahoo-msg, G7) (b) Varying k (com-amazon, G8)



(c) Varying θ (yahoo-msg, G7) (d) Varying θ (com-amazon, G8)

Fig. 9: Comparison of branching vertex selection strategies

achieving around 10× speedups on average. This verifies the effectiveness of the proposed divide-and-conquer techniques with **SCR 1-2**. Similarly, both algorithms have the running time increase when k decreases and θ grows.

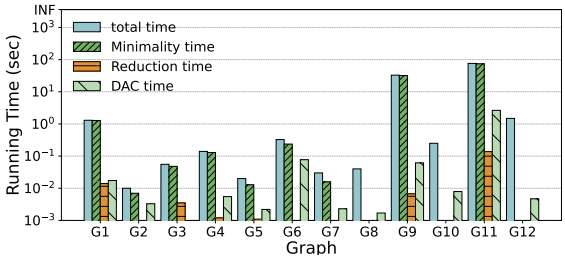
Time costs of conducting various techniques. We study the time costs of applying the proposed techniques during the execution of our proposed `IMinC`. Specifically, we measure the running time of the following procedures.

- **Minimality time**: the time costs of conducting the procedure of checking the minimality of a k-core during the recursion.
- **Reduction time**: the time costs of conducting the core reduction and `SCR 3` during the recursion.
- **DAC time**: the time costs of conducting the divide-and-conquer techniques with **SCR 1-2**.

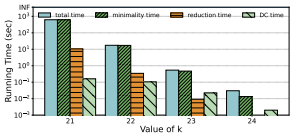
We report the running times in Figure 11(a), Figure 11(b,c), and Figure 11(d,e), by using the default setting on all datasets, varying k on two default datasets, and varying θ on two default datasets, respectively. We observe the followings. First, the time cost of checking the minimality dominates that of other procedures in most settings and datasets. It well-aligns with our theoretical analysis, i.e., the complexity of checking the minimality is $O(nm)$. Hence, the improvement of this procedure would advance the solution in both theory and practice, for which we leave it as a potential research direction in the future. Second, the time cost of the reduction uses a larger proportion when k decreases and/or θ grows on com-amazon. This is because the number of minimal k-cores exponentially increases when k decreases and/or θ grows.

C. Case Study: Community search

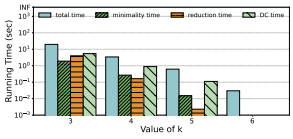
We first present a case study of community (i.e., research group) search on the AMiner Academic Citation Dataset (https://www.kaggle.com/datasets/kmader/aminer-academic-citation-dataset) by mining the maximal k-core and minimal k-cores. Specifically, the dataset consists of around 4.2 million cooperation relationships between 1.7 million authors. It can be modeled as a graph by representing each author by a vertex and adding an edge between two authors if they collaborate at least 3 times. Consequently, the resulting graph includes around 0.24 million vertices and around 0.44 million edges. Given a researcher as the query, the task aims to find the research group that contains the query researcher. To solve the task, we select Prof. Raghu Ramakrishnan as the query, and then find the maximal 8-core and three minimal 8-cores containing the query as the found research group. In particular, the found maximal 8-core is reported in Figure 12(a), which consists of 808 vertices and 5,111 edges with the density of 0.0157. The found three minimal 8-cores (which are subgraphs of the found maximal 8-core) are reported in Figures 12(b), 12(c), and 12(d), which include 15, 15, 13 vertices and 81, 79, 67 edges with the density of 0.7714, 0.7524, 0.8590, respectively. We observe that



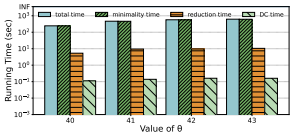
(a) All datasets (default k and θ)



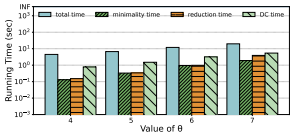
(b) Varying k (yahoo-msg, G7)



(c) Varying k (com-amazon, G8)

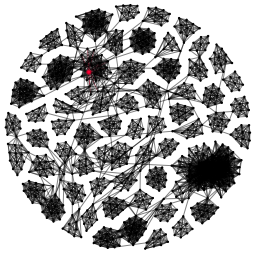


(d) Varying θ (yahoo-msg, G7)

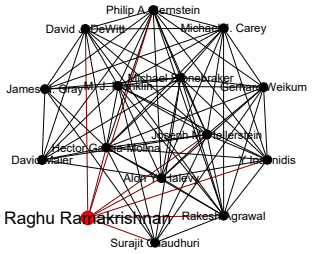


(e) Varying θ (com-amazon, G8)

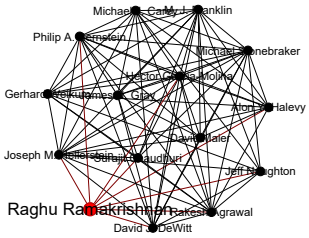
Fig. 11: Comparison of time costs across techniques



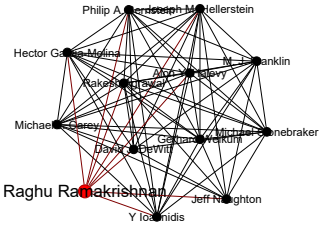
(a) The maximal 8-core



(b) A minimal 8-core g_1



(c) A minimal 8-core g_2



(d) A minimal 8-core g_3

Fig. 12: Case study: research groups found by mining the maximal 8-core and minimal 8-cores

the found minimal 8-cores enjoy high density compared to the maximal 8-cores, and thus the corresponding research groups are more cohesiveness, e.g., every researchers have collaborated with more than half of other researchers within the group. Besides, based on the visualization of the maximal 8-core, it naturally consists of serval local dense regions each of which corresponds to one minimal 8-core. In summary, finding minimal k-cores tends to perform better than finding the maximal k-core on the task of searching research group (since the former would find denser subgraphs).

D. Case Study: Protein Complex Prediction

We present a case study on protein complex prediction in Protein-Protein Interaction (PPI) networks by minimal k-cores (with size constraints) using `IMinC`. Following the existing study [52], we construct a PPI network by merging two PPI datasets on *Escherichia coli* from [3], [53], resulting in a network with 2,119 vertices and 3,778 edges. For the ground truth, we obtain known *E. coli* protein complexes from ECOCYC[1]. To focus on biologically meaningful complexes, we restrict the analysis to those complexes containing at least 4 proteins, yielding 41 complexes in the ground truth dataset for evaluation. We evaluate the performance of `IMinC` using *Geometric Accuracy [49] (or Accuracy)*, a widely used metric in protein complex prediction. Let C_i denote the i-th ground truth complex, P_j the j-th predicted complex (i.e., minimal k-cores identified by `IMinC`), p the number of ground truth complexes, and q the number of predicted complexes. Geometric Accuracy is defined as the geometric mean of *Sensitivity (SNS)* and *Positive Predictive Value (PPV)*, where

$$SNS = \frac{\sum_{i=1}^{p} \max_{j=1}^{q}(|C_i \cap P_j|)}{\sum_{i=1}^{p} |C_i|}, \tag{10}$$

$$PPV = \frac{\sum_{j=1}^{q} \max_{i=1}^{p}(|C_i \cap P_j|)}{\sum_{j=1}^{q} \sum_{i=1}^{p} |C_i \cap P_j|}. \tag{11}$$

Result. We extract all minimum and minimal k-cores (with $\theta = 7$) in our case study. Table IV shows that as k increases, **SNS** decreases and **PPV** increases. This is because a higher k leads to denser but fewer k-cores. In this study, the result for minimal 2-cores yields the highest **Accuracy**. Additionally, compared to minimum k-cores, minimal k-cores are able to cover more complexes, as reflected in the increase in **SNS**, while the decrease in **PPV** remains negligible. The **Accuracy** of minimal k-cores consistently exceeds that of minimum k-cores. We also show in Table III that certain ground truth complexes that are missed by minimum k-cores are successfully matched by minimal k-cores. In terms of efficiency, the average time per solution for minimal k-cores is lower, confirming their superior scalability. These results further demonstrate the limitations of minimum k-cores and the advantages of minimal k-cores for protein complex prediction. Moreover, we fix $k = 2$ and evaluate the impact of varying θ in Table V. We know that when $k = 2$, **SNS** increases when θ grows, while **PPV** initially rises and then declines. We note that, for $\theta \in \{5, 6, 7, 8\}$, **Accuracy** consistently reaches a high value of 0.56. It indicates that increasing the number of predicted k-cores improves **SNS** by producing more matches with the ground truth complexes. However, beyond a certain threshold, the additional k-cores introduce irrelevant

[1]https://ecocyc.org/group?id=:ALL-PROTEINS-2&orgid=ECOLI

TABLE III: Protein complexes that match at least 3 members with minimal 2-cores but not in minimum 2-cores with $\theta = 7$

ground truth complex	**complex members,** C	$\|C\|$	**predicted complex members,** P	$\|P\|$	$\|C \cap P\|$
holo-EntB dimer	entB, entD, entE, entF	4	ydiR, ygiT, entB, entE, entF, rpmE	6	3
primosome	dnaB, dnaG, dnaT, priA, priB, priC	6	dnaB, dnaG, priC, eutC, uvrA	5	3
Ubi complex	ubiE, ubiF, ubiG, ubiH, ubiI, ubiJ, ubiK	7	insO, proQ, ubiF, ubiG, ubiH, ybjM, ykgN	7	3
flagellar export apparatus	flhA, flhB, fliH, fliI, fliJ, fliO, fliP, fliQ, fliR	9	fliH, fliI, fliJ, fliT, glcC, recJ, topA	7	3

TABLE IV: Case study: protein complex prediction with $\theta = 7$

model	**SNS**	**PPV**	**Accuracy**	**# k-cores**	**Avg. Time per Solution**
minimum 2-cores	0.37788	0.615497	0.48227	643	6.33e-4
minimum 3-cores	0.262673	0.747541	0.443124	222	2.24e-3
minimum 4-cores	0.032258	**1**	0.179605	67	5.97e-3
minimal 2-cores	**0.506912**	0.634537	**0.567146**	255168	4.27e-4
minimal 3-cores	0.327189	0.637196	0.4566	2279	1.59e-5
minimal 4-cores	0.0553	0.936652	0.227588	165	3.03e-5

TABLE V: Case study: protein complex prediction with $k = 2$

θ	**SNS**	**PPV**	**Accuracy**	**# k-cores**
3	0.37788	0.615497	0.48227	643
4	0.437788	0.64878	0.532943	3,797
5	0.470046	**0.681006**	0.565778	10,268
6	0.497696	0.643324	0.565844	57,686
7	0.506912	0.634537	**0.567146**	255,168
8	0.529954	0.596698	0.562337	1,432,365
9	**0.534562**	0.576742	0.555252	7,220,780

information, leading to a decline in **PPV**. Thus, it is crucial to impose a size constraint on θ to balance **Accuracy** and complexity by limiting the number of k-cores generated.

VII. Related Work

Core mining. The concept of (maximal) k-core was first introduced by Seidman in 1983 [54]. One important problem, called *core decomposition*, seeks to decompose a graph into nested cores [6], [10], [11], [17], [39], [44], [72], [66], [59]. It can be solved efficiently in linear time $O(m+n)$ [6] and thus has been widely used as a building block when solving other graph mining problems, e.g., finding the densest subgraph [29] and k-clique listing [58]. Recent studies on core decomposition aims to develop efficient parallel and distributed algorithms or to extend core to different types of graphs [10], [11], [17], [39], [72], [66], [59]. Other related problems include the *collapse k-core problem* [45], [68], the *k-core problems with size constraints* [55], [22], [5], [46], [61], [37], the *connected k-core* problem [55], [65], and the *anchored k-core* problems [9], [42]. Recent studies have considered the minimum k-core search problem, which aims to find the k-core with the smallest number of vertices. Specifically, [47], [9] address the problem via integer programming, while [71], [37] focus on branch-and-bound methods. While the minimum k-core is necessarily minimal, it captures only a limited subset of the graph's information, ignoring other significant k-cores (as verified in Section VI-D). These limitations highlight the need for more efficient algorithms and a deeper understanding of the k-core model in memory-related applications. For instance, prior work explored the use of minimal k-core mining for neuro-biological modeling of memory formation and recall [60]. However, their approach `Wh` is computationally expensive and does not scale efficiently to real-world datasets, which is demonstrated in our experiments (Section VI).

Cohesive subgraph enumeration. One important problem in cohesive subgraph mining is to enumerate cohesive subgraphs. Notable problems include *maximal cliques enumeration* [23], [12], [16], [21], [48], [57], *maximal k-plexes enumeration* [7], [20], [38], [24], *maximal s-defective cliques enumeration* [50], and *maximal γ-quasi-cliques enumeration* [41], [34], [62]. We note that some of our proposed techniques are influenced by prior studies in cohesive subgraph mining (particularly k-plex) [25], [15], [63], [12], [30]. However, these techniques have been significantly adapted and extended to address the unique challenges of the new problem introduced in our work.

For an overview on cohesive subgraphs, see the excellent books and survey, e.g., [14], [26], [27], [32], [36].

VIII. Conclusion

In this paper, we revisited the problem of enumerating all minimal k-cores. To solve the problem, we proposed an improved branch-and-bound algorithm, `IMinC`, which incorporates pivot and divide-and-conquer techniques. `IMinC` achieves an improved worst-case time complexity of $O^*(\alpha_\ell^n)$, where α_ℓ is a positive number strictly smaller than 2. We further extended `IMinC` with reduction techniques to handle the size constraint. Extensive experiments on 12 real-world datasets demonstrated the superiority of `IMinC`. In future work, we aim to adapt `IMinC` to other graph types.

Acknowledgments

We thank reviewers for their valuable comments and effort to improve this manuscript. Shengxin Liu is supported by the Guangdong Basic and Applied Basic Research Foundation (2025A1515060015). Raymond Chi-Wing Wong is supported by fund PRP/004/25FX. Xun Zhou is supported by the National Natural Science Foundation of China (62472125), Guangdong Basic and Applied Basic Research Foundation (2025A1515011258), Key Technologies R&D Program of Guangdong Province (2026B0909060001) and Shenzhen Science and Technology Programs (GXWD20231128102922001, ZDCY20250901111705007, ZDSYS20230626091203008). Cheng Long is supported in part by the Ministry of Education, Singapore, under its Academic Research Fund (Tier 2 Award MOE-T2EP20224-0011 and Tier 1 Award (RG20/24)). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

IX. AI-Generated Content Acknowledgement

The authors declare that no AI-generated content was used in the creation of this work.

REFERENCES

[1] https://github.com/sykhhhh/minimal-kcore/, Technical Report and Source Code.

[2] M. Altaf-Ul-Amine, K. Nishikata, T. Korna, T. Miyasato, Y. Shinbo, M. Arifuzzaman, C. Wada, M. Maeda, T. Oshima, H. Mori *et al.*, "Prediction of protein functions based on k-cores of protein-protein interaction networks and amino acid sequences," *Genome Informatics*, vol. 14, pp. 498–499, 2003.

[3] M. Arifuzzaman, M. Maeda, A. Itoh, K. Nishikata, C. Takita, R. Saito, T. Ara, K. Nakahigashi, H.-C. Huang, A. Hirai *et al.*, "Large-scale identification of protein-protein interaction of escherichia coli k-12," *Genome Research*, vol. 16, no. 5, pp. 686–691, 2006.

[4] G. D. Bader and C. W. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 4, no. 1, pp. 1–27, 2003.

[5] N. Barbieri, F. Bonchi, E. Galimberti, and F. Gullo, "Efficient and effective community search," *Data Mining and Knowledge Discovery*, vol. 29, no. 5, pp. 1406–1433, 2015.

[6] V. Batagelj and M. Zaveršnik, "An $O(m)$ algorithm for cores decomposition of networks," *CoRR*, vol. cs.DS/0310049, 2003.

[7] D. Berlowitz, S. Cohen, and B. Kimelfeld, "Efficient enumeration of maximal k-plexes," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2015, pp. 431–444.

[8] M. Bhattacharyya and S. Bandyopadhyay, "Mining the largest quasi-clique in human protein interactome," in *Proceedings of the International Conference on Adaptive and Intelligent Systems*, 2009, pp. 194–199.

[9] K. Bhawalkar, J. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma, "Preventing unraveling in social networks: The anchored k-core problem," *SIAM Journal on Discrete Mathematics*, vol. 29, no. 3, pp. 1452–1475, 2015.

[10] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich, "Core decomposition of uncertain graphs," in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2014, pp. 1316–1325.

[11] F. Bonchi, A. Khan, and L. Severini, "Distance-generalized core decomposition," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2019, pp. 1006–1023.

[12] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[13] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang *et al.*, "Topological structure analysis of the protein–protein interaction network in budding yeast," *Nucleic Acids Research*, vol. 31, no. 9, pp. 2443–2450, 2003.

[14] L. Chang and L. Qin, *Cohesive Subgraph Computation over Large Sparse Graphs*. Springer, 2018.

[15] L. Chang and K. Yao, "Maximum k-plex computation: Theory and practice," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 2, no. 1, pp. 1–26, 2024.

[16] L. Chang, J. X. Yu, and L. Qin, "Fast maximal cliques enumeration in sparse graphs," *Algorithmica*, vol. 66, pp. 173–186, 2013.

[17] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2011, pp. 51–62.

[18] Q. Cheng, D. Yan, T. Wu, L. Yuan, J. Cheng, Z. Huang, and Y. Zhou, "Efficient enumeration of large maximal k-plexes," in *Proceedings of the International Conference on Extending Database Technology (EDBT)*, 2025.

[19] S. Cohen, B. Kimelfeld, and Y. Sagiv, "Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties," *Journal of Computer and System Sciences*, vol. 74, no. 7, pp. 1147–1159, 2008.

[20] A. Conte, T. De Matteis, D. De Sensi, R. Grossi, A. Marino, and L. Versari, "D2K: Scalable community detection in massive networks via small-diameter k-plexes," in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2018, pp. 1272–1281.

[21] A. Conte, R. Grossi, A. Marino, and L. Versari, "Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques," in *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016, pp. 1–148.

[22] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2014, pp. 991–1002.

[23] Q. Dai, R.-H. Li, M. Liao, and G. Wang, "Maximal defective clique enumeration," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 1, no. 1, pp. 1–26, 2023.

[24] Q. Dai, R.-H. Li, H. Qin, M. Liao, and G. Wang, "Scaling up maximal k-plex enumeration," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2022, pp. 345–354.

[25] Q. Dai, R.-H. Li, X. Ye, M. Liao, W. Zhang, and G. Wang, "Hereditary cohesive subgraphs enumeration on bipartite graphs: The power of pivot-based approaches," *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, vol. 1, no. 2, pp. 1–26, 2023.

[26] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, "A survey of community search over big graphs," *The VLDB Journal*, vol. 29, pp. 353–392, 2020.

[27] Y. Fang, K. Wang, X. Lin, and W. Zhang, "Cohesive subgraph search over big heterogeneous information networks: Applications, challenges, and solutions," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2021, pp. 2829–2838.

[28] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 6, pp. 854–867, 2020.

[29] Y. Fang, K. Yu, R. Cheng, L. V. S. Lakshmanan, and X. Lin, "Efficient algorithms for densest subgraph discovery," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1719–1732, 2019.

[30] S. Gao, K. Yu, S. Liu, and C. Long, "Maximum k-plex search: An alternated reduction-and-bound method," *Proceedings of the VLDB Endowment*, vol. 18, no. 2, pp. 363–376, 2025.

[31] B. Ghosh, M. E. Ali, F. M. Choudhury, S. H. Apon, T. Sellis, and J. Li, "The flexible socio spatial group queries," *Proceedings of the VLDB Endowment*, vol. 12, no. 2, pp. 99–111, 2018.

[32] X. Huang, L. V. S. Lakshmanan, and J. Xu, *Community Search over Big Graphs*. Morgan & Claypool Publishers, 2019.

[33] W. Jack, A. Ray, and T. Suri, "Transaction networks: Evidence from mobile money in Kenya," *American Economic Review*, vol. 103, no. 3, pp. 356–361, 2013.

[34] J. Khalil, D. Yan, G. Guo, and L. Yuan, "Parallel mining of large maximal quasi-cliques," *The VLDB Journal*, vol. 31, no. 4, pp. 649–674, 2022.

[35] F. Kyriakopoulos, S. Thurner, C. Puhr, and S. W. Schmitz, "Network and eigenvalue analysis of financial transaction networks," *The European Physical Journal B*, vol. 71, pp. 523–531, 2009.

[36] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal, "A survey of algorithms for dense subgraph discovery," in *Managing and Mining Graph Data*. Springer, 2010, pp. 303–336.

[37] C. Li, F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Efficient progressive minimum k-core search," *Proceedings of the VLDB Endowment*, vol. 13, no. 3, pp. 362–375, 2020.

[38] D. R. Lick and A. T. White, "k-degenerate graphs," *Canadian Journal of Mathematics*, vol. 22, no. 5, pp. 1082–1096, 1970.

[39] B. Liu, F. Zhang, C. Zhang, W. Zhang, and X. Lin, "Corecube: Core decomposition in multilayer graphs," in *Proceedings of International Web Information Systems Engineering Conference (WISE)*, 2019, pp. 694–710.

[40] B. Liu, F. Zhang, W. Zhang, X. Lin, and Y. Zhang, "Efficient community search with size constraint," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2021, pp. 97–108.

[41] G. Liu and L. Wong, "Effective pruning techniques for mining quasi-cliques," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2008, pp. 33–49.

[42] K. Liu, S. Wang, Y. Zhang, and C. Xing, "An efficient algorithm for the anchored k-core budget minimization problem," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2021, pp. 1356–1367.

[43] Q. Liu, M. Zhao, X. Huang, J. Xu, and Y. Gao, "Truss-based community search over large directed graphs," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2020, pp. 2183–2197.

[44] Y. Liu, X. Dong, Y. Gu, and Y. Sun, "Parallel k-core decomposition: Theory and practice," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 3, no. 3, pp. 1–27, 2025.

[45] J. Luo, H. Molter, and O. Suchý, "A parameterized complexity view on collapsing k-cores," in *Proceedings of the International Symposium on Parameterized and Exact Computation (IPEC)*, 2018, pp. 7:1–7:14.

[46] Y.-L. Ma, Y. Yuan, F.-D. Zhu, G.-R. Wang, J. Xiao, and J.-Z. Wang, "Who should be invited to my party: A size-constrained k-core problem in social networks," *Journal of Computer Science and Technology*, vol. 34, pp. 170–184, 2019.

[47] D. Mikesell and I. V. Hicks, "Minimum k-cores and the k-core polytope," *Networks*, vol. 80, no. 1, pp. 93–108, 2022.

[48] K. A. Naudé, "Refined pivot selection for maximal clique enumeration in graphs," *Theoretical Computer Science*, vol. 613, pp. 28–37, 2016.

[49] T. Nepusz, H. Yu, and A. Paccanaro, "Detecting overlapping protein complexes in protein-protein interaction networks," *Nature Methods*, vol. 9, no. 5, pp. 471–472, 2012.

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

[51] D. Peleg, I. Sau, and M. Shalom, "On approximating the d-girth of a graph," *Discrete Applied Mathematics*, vol. 161, no. 16-17, pp. 2587–2596, 2013.

[52] A. Rahman, K. Roy, R. Maliha, and T. F. Chowdhury, "A fast exact algorithm to enumerate maximal pseudo-cliques in large sparse graphs," in *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2024, pp. 2479–2490.

[53] S. V. Rajagopala, P. Sikorski, A. Kumar, R. Mosca, J. Vlasblom, R. Arnold, J. Franca-Koh, S. B. Pakala, S. Phanse, A. Ceol *et al.*, "The binary protein-protein interaction landscape of escherichia coli," *Nature Biotechnology*, vol. 32, no. 3, pp. 285–290, 2014.

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

[55] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2010, pp. 939–948.

[56] P. Symeonidis, E. Tiakas, and Y. Manolopoulos, "Product recommendation and rating prediction based on multi-modal social networks," in *Proceedings of the ACM Conference on Recommender systems (RecSys)*, 2011, pp. 61–68.

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

[58] K. Wang, K. Yu, and C. Long, "Efficient k-clique listing: An edge-oriented branching strategy," *Proc. ACM Manag. Data (SIGMOD)*, vol. 2, no. 1, pp. 1–26, 2024.

[59] Z. Wang, M. Zhong, Y. Zhu, T. Qian, M. Liu, and J. X. Yu, "On more efficiently and versatilely querying historical k-cores," *Proceedings of the VLDB Endowment*, vol. 18, no. 5, pp. 1335–1347, 2025.

[60] C. I. Wood and I. V. Hicks, "The minimal k-core problem for modeling k-assemblies," *The Journal of Mathematical Neuroscience*, vol. 5, no. 1, pp. 1–19, 2015.

[61] K. Yao and L. Chang, "Efficient size-bounded community search over large networks," *Proceedings of the VLDB Endowment*, vol. 14, no. 8, pp. 1441–1453, 2021.

[62] K. Yu and C. Long, "Fast maximal quasi-clique enumeration: A pruning and branching co-design approach," *Proc. ACM Manag. Data (SIGMOD)*, vol. 1, no. 3, pp. 1–26, 2023.

[63] ——, "Maximum k-biplex search on bipartite graphs: A symmetric-BK branching approach," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 1, no. 1, pp. 1–26, 2023.

[64] K. Yu, C. Long, S. Liu, and D. Yan, "Efficient algorithms for maximal k-biplex enumeration," in *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2022, pp. 860–873.

[65] Y. Yu, D. Wen, M. Yu, L. Qin, Y. Zhang, W. Zhang, and X. Lin, "Querying historical k-cores in large temporal graphs," *The VLDB Journal*, vol. 34, no. 2, p. 26, 2025.

[66] C. Zhang, Y. Zhu, and L. Chang, "A local search approach to efficient (k, p)-core maintenance," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 3, no. 1, pp. 1–26, 2025.

[67] F. Zhang, H. Guo, D. Ouyang, S. Yang, X. Lin, and Z. Tian, "Size-constrained community search on large networks: An effective and efficient solution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, pp. 356–371, 2024.

[68] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Finding critical users for social network engagement: The collapsed k-core problem," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017, pp. 245–251.

[69] H. Zhang, Z. Wang, S. Chen, and C. Guo, "Product recommendation in online social networking communities: An empirical study of antecedents and a mediator," *Information & Management*, vol. 56, no. 2, pp. 185–195, 2019.

[70] Q. Zhang, Y. Zhang, R.-H. Li, and G. Wang, "Approximate anchored densest subgraph search on large static and dynamic graphs," *Proceedings of the VLDB Endowment*, vol. 18, no. 3, pp. 623–636, 2024.

[71] Q. Zhang and S. Liu, "Efficient exact minimum k-core search in real-world graphs," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2023, pp. 3391–3401.

[72] W. Zhang, Z. Yang, D. Wen, W. Li, W. Zhang, and X. Lin, "Accelerating core decomposition in billion-scale hypergraphs," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 3, no. 1, pp. 1–27, 2025.

[73] Y. Zhou, Q. Guo, and Y. Fang, "Efficient k-clique densest subgraph discovery: Towards bridging practice and theory," *Proceedings of the VLDB Endowment*, vol. 18, no. 10, pp. 3490–3503, 2025.

APPENDIX

A. Omitted Proofs

1) Proof of Lemma 2:

Proof. This can be easily verified by contradiction. Suppose that a k-core g with $V(g) \leq 2k+1$ has the diameter lager than 2. Hence, there exists a pair of non-adjacent vertices u and v such that they have no common neighbors, i.e., $(u, v) \notin E(g)$ and $N_g(u) \cap N_g(v) = \emptyset$ (since otherwise the diameter is at most 2). We thus have $|V(g)| \geq |\{u, v\}| + |N_g(u)| + |N_g(v)| \geq 2k + 2$, which contradicts to the fact that $V(g) \leq 2k + 1$. □

2) Proof of Lemma 3:

Proof. This can be easily proved by contradiction and the inclusion-exclusion principle. In general, there are two cases. First, when (u, v) is an edge in G, we have

$$\begin{aligned} |V(g)| &\geq |N_g(u) \cup N_g(v)| \\ &= |N_g(u)| + |N_g(v)| - |N_g(u) \cap N_g(v)| \\ &\geq |N_g(u)| + |N_g(v)| - |N_G(u) \cap N_G(v)| \\ &\geq k + k - (2k - \theta - 1) \geq \theta + 1, \end{aligned} \tag{12}$$

which contradicts to the fact that $V(g) \leq \theta$. Second, when $(u, v) \notin E$, we have

$$\begin{aligned} |V(g)| &\geq |\{u, v\}| + |N_g(u) \cup N_g(v)| \\ &= 2 + |N_g(u)| + |N_g(v)| - |N_g(u) \cap N_g(v)| \\ &\geq 2 + |N_g(u)| + |N_g(v)| - |N_G(u) \cap N_G(v)| \\ &\geq 2 + k + k - (2k - \theta + 1) \geq \theta + 1, \end{aligned} \tag{13}$$

which contradicts to the fact that $V(g) \leq \theta$. □

3) Proof of Lemma 4:

Proof. This can be easily proved by contradiction. Suppose that a minimal k-core g found in branch B has the size smaller than $k - \min_{v \in S} d_{G[S]}(v) + |S|$, i.e., $|V(g)| < k - \min_{v \in S} d_{G[S]}(v) + |S|$. Clearly, we have $S \subseteq V(g)$. Let v' in S be the vertex with the smallest degree within S, i.e., $v' = \arg\min_{v \in S} d_{G[S]}(v)$. We have

$$d_g(v') = d_{G[S]}(v') + |N_g(v') \backslash S| < k, \tag{14}$$

which contradicts to the fact that g is a minimal k-core. □

B. Discussion on Output-sensitive Complexity

An enumeration algorithm is typically considered output-sensitive if its running time is bounded by a polynomial function of the input size and the output size. While IMinC achieves the worst-case time complexity of $O^*(\alpha_\ell)$, it does not guarantee a tight output-sensitive bound. Specifically, the total runtime is determined by the product of the number of solutions and the maximum delay between consecutive outputs. Since IMinC adopts the branch-and-bound framework to prioritize practical pruning efficiency, the worst-case delay can be exponential (and thus not polynomially bounded). Consequently, in scenarios where the solution set is small but the search space is large (e.g., dense graphs with high degeneracy), the algorithm may explore substantial portions of the search tree without producing outputs. This design is a deliberate trade-off compared to reverse-search-based methods [64], [19], which achieve polynomial delay theoretically but often suffer from significant overhead in practice.

Algorithm 3: Adapting IMinC-θ for finding size constraint minimal k-cores

Input: A graph $G = (V, E)$ and two positive integers k, θ with $\theta < 2k + 1$
Output: All minimal k-cores with the size at most a threshold θ in G

```
/* Divide-and-conquer with SCR 1-2 */
1  g_k ← the maximal k-core of G;
2  G ← g_k;
3  for each vertex v_i in {v_1, v_2, ..., v_n} do
4      G_i ← G[{v_i, ..., v_n}];
5      G'_i ← G_i[N^2_{G_i}(v_i)];
6      Refine G'_i by applying SCR 2;
7      g_{k,i} ← the maximal k-core of G_i;
8      IMinC_Rec-θ(g_{k,i}, {v_i}, V(g_{k,i}) \ {v_i}, k);

9  Procedure IMinC_Rec-θ(G, S, C, k)
       /* Termination and reductions */
10     if |S| > θ then return;
11     if G[S] is a k-core then
12         if G[S \ {v_i}] does not induce a k-core
             ∀v_i ∈ S then
13             Output S;
14         return;
15     if C = ∅ then return;
       /* Core Reductions */
16     g'_k ← the maximal k-core of G[S ∪ C];
17     if S ⊊ V(g'_k) or C ∩ V(g'_k) = ∅ then
18         return;
19     C ← C ∩ V(g'_k);
       /* Lower bound based reduction */
20     LB ← k − min_{v∈S} d_{G[S]}(v) + |S|;
21     if LB > θ then return;
       /* Branching with the pivot */
22     v* ← arg min_{v∈Ψ(B)} d_{G[S∪C](v)};
23     u* ← a vertex selected from N_{G[S∪C]}(v*) ∩ C;
24     IMinC_Rec(G, S ∪ {u*}, C \ {u*}, k);
25     IMinC_Rec(G, S, C \ {u*}, k);
```

C. The Algorithm IMinC-θ and Its Analysis

Summary. We summarize the adaptation of our IMinC with the newly developed reductions in Algorithm 2. We remark that **SCR 1** and **SCR 2** are used to improve the divide-and-conquer technique by refining the input graph for each subproblem (Lines 5-6), and **SCR 3** is a new reduction to improve the branch-and-bound process by conducting it after the core reductions (Lines 20-21). Besides, when the size of

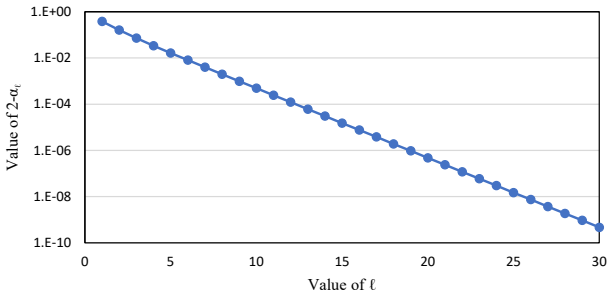
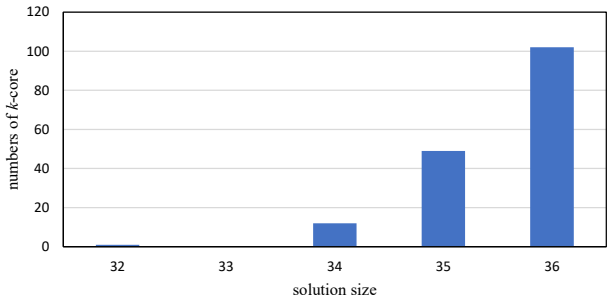
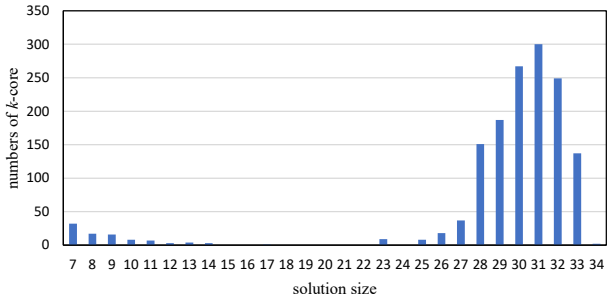


Fig. 13: The value of $2 - \alpha_\ell$ with respect to ℓ



(a) yahoo-msg, G7



(b) com-amazon, G8

Fig. 14: Distribution of minimal k-core sizes on G7 and G8.

the partial set S is larger than the threshold θ, we can terminate the branch since it is no longer possible to find a minimal k-core with at most θ vertices in that branch.

Time complexity analysis. First, based on SCR 1, the size of G_i' can be bounded by $\delta\Delta$. Second, although additional reductions are deployed in `IMinC`, the worst-case time complexity is still dominated by the part of checking the minimality of a k-core which runs in $O(nm)$. Therefore, the worst-case time complexity is $O(n^2 m \alpha_\ell^{\delta\Delta})$.

D. Additional Experiments

We provide additional experiments in this section.

The value of α_ℓ. Figure 13 illustrates the gap between α_ℓ and 2 as a function of ℓ. Since α_ℓ approaches 2 as ℓ increases, we use $2 - \alpha_\ell$ as a more illustrative measure to visualize its asymptotic behavior.

Distribution of solution sizes. We analyze the size distribution of minimal k-cores using two representative datasets, G7 and G8. As shown in Figure 14, the results reveal distinct patterns:

TABLE VI: Memory consumption on all datasets (KB). Symbol "-" denotes timeouts/failures.

ID	IMinC	WH	Plex-Enum	IBB
G1	3,072	-	-	4,096
G2	5,120	4,096	19,456	5,120
G3	10,240	-	37,888	9,216
G4	10,240	5,120	-	8,192
G5	12,288	-	37,888	9,216
G6	35,660	-	-	32,768
G7	46,620	-	29,696	45,056
G8	110,040	-	41,984	69,632
G9	685,640	-	-	637,952
G10	599,192	-	129,024	277,504
G11	2,273,388	-	-	-
G12	7,289,380	-	1,425,408	2,982,912

TABLE VII: Impact of the divide-and-conquer strategy on maximum recursion depth

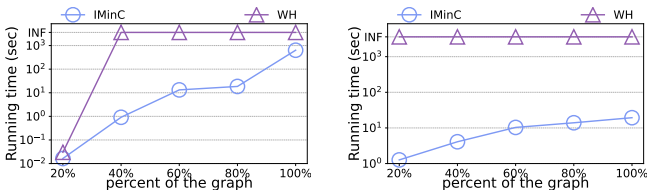
ID	IMinC	IMinC\DAC
G1	69	72
G2	33	35
G3	24	26
G4	19	28
G5	26	44
G6	16	22
G7	36	41
G8	3	79
G9	42	43
G10	1	597
G11	393	400
G12	1	557

in G7, the sizes of minimal k-cores are concentrated around the parameter value $k_d = 24$. In contrast, G8 exhibits a heavy-tailed distribution containing numerous large-sized minimal k-cores. Notably, only the unconstrained version of `IMinC`, i.e., without the size constraint, is capable of successfully enumerating these large-scale structures in G8, demonstrating its robustness.

Memory consumption. We further evaluate the memory consumption across all datasets. As presented in Table VI, memory usage shows a linear correlation with the scale of the input graphs. Furthermore, all algorithms demonstrate comparable memory usage. This observation is consistent with their theoretical space complexity of $O(n + m)$.

Impact of divide-and-conquer on recursion depth. As shown in Table VII, the divide-and-conquer framework drastically reduces the maximum recursion depth. This effect is more obvious in sparse graphs like G10 and G12, where the depth is reduced by orders of magnitude. By limiting the recursion depth, our method prevents excessive stack overhead and potential stack overflow, thus ensuring better scalability.

Scalability test. For the scalability test, we utilize G7 and G8 with $k = k_d - 3$ and $\theta = 2k + 1$. In the experiment, we randomly sample between 20% and 100% of the vertices and assess the performance of `IMinC` and `WH`. As shown in Figure 15, we observe the following key points. (1) The running time decreases significantly as the graph size is reduced, primarily because the graph becomes sparser, resulting in a rapid reduction in the number of remaining minimal k-



(a) yahoo-msg, G7 (b) com-amazon, G8

Fig. 15: Scalability test

cores. (2) In all cases, `IMinC` exhibits shorter running times compared to `WH`, and the rate of increase in runtime for `IMinC` is slower than for `WH`. For instance, in Figure 15(b), when the ratio increases from 0.2 to 0.4, the running time of `IMinC` increases from 0.02 seconds to 0.91 seconds, while `WH` rises from 0.03 seconds to exceeding the time limit of 3,600 seconds. These results confirm the scalability of `IMinC`.