

Efficient Querying of Maximum Connectivity-Based Quasi-Cliques in Large Graphs

YANG LIU, Harbin Institute of Technology, Shenzhen, China

HEJIAO HUANG*, Harbin Institute of Technology, Shenzhen, China

KAIQIANG YU, State Key Laboratory of Novel Software Technology, Nanjing University, China

SHENGXIN LIU*, Harbin Institute of Technology, Shenzhen, China

ZHAOQUAN GU, Harbin Institute of Technology, Shenzhen, China and PengCheng Laboratory, China

Structural graph queries involving connectivity constraints are fundamental primitives in modern data management and analytics. Compared with degree- and edge-based variants, connectivity-based γ -quasi-cliques (γ -CQCs) impose stronger structural constraints and fault tolerance, where robustness is measured by vertex connectivity, i.e., the minimum number of vertices whose removal disconnects the subgraph. However, processing such queries on large graphs remains computationally challenging. In this paper, we study the efficient execution of the *Maximum Connectivity-based γ -Quasi-Clique* (MCQC) query, which seeks the γ -CQC with the largest number of vertices. The problem is NP-hard, and the non-hereditary nature of vertex connectivity invalidates many traditional pruning techniques, posing significant challenges for exact query processing. Existing solutions rely on generic optimization formulations and exhibit poor scalability in practice. To address these challenges, we propose MCQC-TD, a specialized exact branch-and-bound framework for the MCQC query. MCQC-TD adopts a top-down search strategy and exploits a novel structural characterization of γ -CQCs to enable effective pruning and guided branching. These techniques substantially reduce the theoretical worst-case complexity compared to existing approaches. We further integrate practical query optimization techniques, including an optimized initialization strategy and a query decomposition framework, to enhance scalability on large graphs. Extensive experiments demonstrate that MCQC-TD consistently outperforms the state-of-the-art solution, achieving speedups of up to five orders of magnitude in query time.

CCS Concepts: • **Theory of computation** → **Graph algorithms analysis**.

Additional Key Words and Phrases: maximum connectivity-based quasi-clique, vertex connectivity, branch-and-bound, exact algorithm

ACM Reference Format:

Yang Liu, Hejiao Huang, Kaiqiang Yu, Shengxin Liu, and Zhaoquan Gu. 2026. Efficient Querying of Maximum Connectivity-Based Quasi-Cliques in Large Graphs. *Proc. ACM Manag. Data* 4, 4 (SIGMOD), Article 272 (September 2026), 27 pages. <https://doi.org/10.1145/3837110>

*Corresponding authors.

Authors' Contact Information: Yang Liu, Shenzhen Key Laboratory of Internet Information Collaboration, School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen, China, 23b951003@stu.hit.edu.cn; Hejiao Huang, Shenzhen Key Laboratory of Internet Information Collaboration, School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen, China, huanghejiao@hit.edu.cn; Kaiqiang Yu, State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China, kaiqiang.yu@nju.edu.cn; Shengxin Liu, Shenzhen Key Laboratory of Internet Information Collaboration, School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen, China, sxliu@hit.edu.cn; Zhaoquan Gu, Harbin Institute of Technology, Shenzhen, Shenzhen, China and PengCheng Laboratory, Shenzhen, China, guzhaoquan@hit.edu.cn.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2836-6573/2026/9-ART272

<https://doi.org/10.1145/3837110>

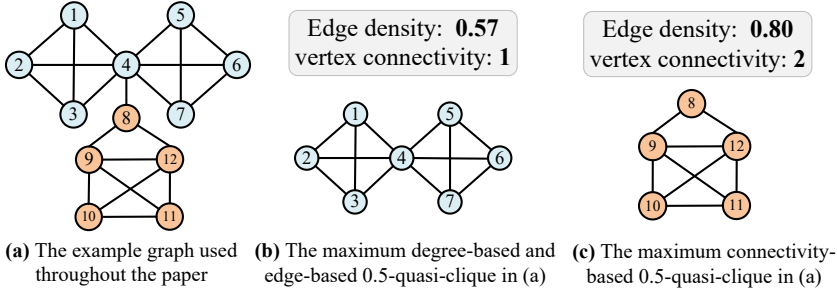


Fig. 1. Illustration of different γ -quasi-clique models.

1 Introduction

Graph data management systems have become indispensable in domains such as social network analysis [6, 22, 31], bioinformatics [43, 61, 62], and recommendation systems [23, 29]. To support these data-intensive applications, modern graph engines are increasingly tasked with processing complex analytical queries that involve rigorous structural constraints [20, 21]. A critical requirement is the efficient execution of exact structural queries that identify cohesive subgraphs [7, 9–11, 16, 26]. Although the clique model is a canonical notion of structural cohesion, its strict requirement of full pairwise connectivity makes it unsuitable for noisy real-world graphs [28, 36, 39, 45], motivating a variety of relaxed clique models, such as k -core [49], k -plex [14, 15, 50], and γ -quasi-clique [2, 54, 59]. Among these, the γ -quasi-clique offers a flexible framework by relaxing the all-to-all connectivity with a parameter $0 \leq \gamma \leq 1$, facilitating applications like protein complex identification [8, 38] and anomaly detection [4, 63].

Existing graph query engines typically support three variants of γ -quasi-cliques: degree-based (γ -DQC) [28, 59, 64], edge-based (γ -EQC) [46, 51, 52], and connectivity-based (γ -CQC) [54]. These variants enforce constraints on *vertex degree*, *edge density*, and *vertex connectivity*, respectively. While γ -DQCs and γ -EQCs have been extensively studied and efficiently supported, query processing for γ -CQCs remains largely underexplored. The connectivity-based formulation endows γ -CQCs with stronger structural guarantees in terms of both cohesiveness and robustness. Specifically, we show that every γ -CQC is also a γ -DQC and a γ -EQC, whereas the converse does not hold (Lemma 2.3). More importantly, γ -CQC is the only quasi-clique model that explicitly enforces structural resilience: a γ -CQC must remain connected after the removal of fewer than $\lceil \gamma(n-1) \rceil$ vertices. This vertex-connectivity guarantee yields more stable and fault-tolerant query results (see our case study in Section 6.3), which is highly desirable in practical graph analytics.

Example 1. Consider the graph in Figure 1(a) and a query that retrieves the maximum 0.5-quasi-clique. The results returned by degree-based (0.5-DQC) and edge-based (0.5-EQC) predicates (Figure 1(b)) satisfy the required degree and density constraints but have vertex connectivity of 1. In contrast, the result returned by the 0.5-CQC predicate (Figure 1(c)) achieves higher edge density and vertex connectivity of 2, indicating a more robust structure.

Furthermore, compared with cohesive subgraph models governed by an absolute integer threshold s (e.g., s -bundle [39]), the γ -CQC model is defined by a proportional constraint γ . This semantic distinction is practically important. In real-world graph analytics, where ground-truth communities may vary substantially in size, a fixed absolute threshold s can be overly restrictive for large communities yet too permissive for small ones. By contrast, the γ -CQC model naturally adapts its

connectivity requirement to the size of the retrieved community, making it appealing for robust structure discovery.

Motivated by this, we study the *Maximum Connectivity-based γ -Quasi-Clique* (MCQC) problem from a data management perspective. Intuitively, vertex connectivity measures structural robustness and fault tolerance, namely, the minimum number of vertices whose removal disconnects a community. Accordingly, the MCQC query aims to efficiently retrieve the valid community with the largest number of vertices under the γ -CQC constraint. However, evaluating MCQC queries is highly challenging. The problem is NP-hard [54], and the non-hereditary nature of vertex connectivity invalidates many traditional monotonicity-based pruning techniques. To the best of our knowledge, the only existing exact solution for MCQC is the Exact Iterative Algorithm (EIA) [54], which formulates the problem as a generic Mixed Integer Programming (MIP) model. Because it relies on general-purpose operations research (OR) solvers and does not exploit the topological locality and structural properties of graph data, EIA suffers from severe scalability limitations in practice.

Our Solutions. To address these data management challenges, we propose MCQC-TD, a specialized exact branch-and-bound framework. As illustrated in Figure 2(a), MCQC-TD adopts a *top-down search strategy*. It first initializes an upper bound (denoted as ub) on the size of the maximum γ -CQC, and then iteratively enforces a strict *target size constraint* (denoted as s), starting from ub and decrementing it systematically. At each iteration, MCQC-TD checks whether a γ -CQC of size s exists. This design guarantees that the first feasible solution identified corresponds to the true maximum γ -CQC (denoted as s^*). Once s^* is found, the search terminates immediately, thereby bypassing the extensive redundant exploration of smaller γ -CQCs that is inherent in traditional bottom-up approaches.

Crucially, the target size s also serves as the basis for our micro-level pruning mechanism. As shown in Figure 2(b), s induces a strict *pruning threshold*. Derived from our structural characterization of γ -CQCs, this pruning threshold is strictly smaller than the trivial size limit s . Within the resulting *feasible search region*, we employ a *hybrid branching strategy* (**HPS**) to guide the search. **HPS** prioritizes branching choices that rapidly increase a *structural indicator value*, a metric defined based on our characterization of γ -CQCs. During the search, whenever a search state violates the pruning threshold, our reduction rule (**RR**) is immediately triggered to prune the branch at a shallow depth i (where $i < s$), preventing unnecessary exploration toward deeper levels. The tight integration of the top-down strategy, **HPS**, and **RR** reduces the worst-case time complexity from the trivial $O^*(2^n)$ to $O^*(\lambda_t^n)$, where $\lambda_t < 2$.

To enhance practical scalability, we further integrate advanced optimization techniques, including an optimized initialization strategy and a query decomposition method, enabling efficient processing on massive real-world graphs.

Contributions. The main contributions of our paper are as follows.

- We propose MCQC-TD, the first exact branch-and-bound algorithm for the MCQC query, utilizing a top-down search framework. Theoretically, MCQC-TD reduces the trivial worst-case time complexity of $O^*(2^n)$ to $O^*(\lambda_t^n)$ (where $\lambda_t < 2$) by leveraging a new reduction rule and a hybrid branching strategy. (Section 4)
- We design an optimized initialization strategy to construct high-quality initial solutions for accelerated pruning, along with a query decomposition framework that enables scalable processing on massive graphs. (Section 5)
- Extensive evaluations on 664 graphs from three collections demonstrate that: (1) MCQC-TD significantly outperforms the state-of-the-art method, EIA [54], in terms of the number of solved query instances. Particularly, in the 10th DIMACS and real-world collections, MCQC-TD processes more queries within 1 second than EIA does within 3 hours; (2) on 40 representative graphs, MCQC-TD

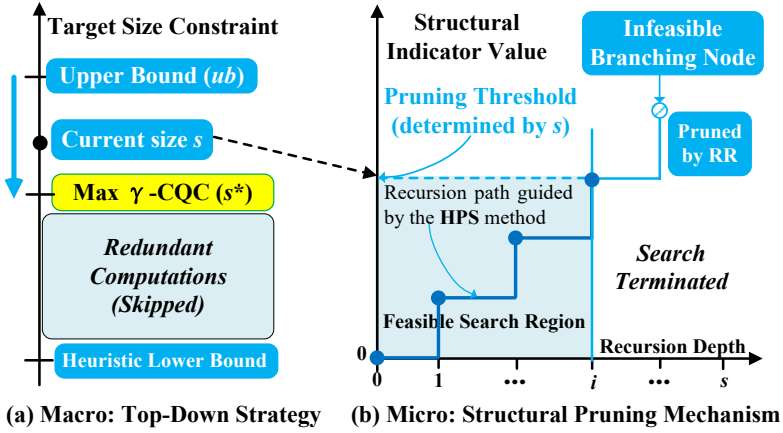


Fig. 2. Illustration of our MCQC-TD framework.

achieves up to five orders of magnitude speedup in query time; (3) each optimization technique contributes positively to the overall performance. (Sections 6.1–6.2)

- A case study on a real-world dataset with ground-truth labels validates the effectiveness of the γ -CQC predicate. The results show that the maximum γ -CQC yields query results that better match the ground truth compared to degree- and edge-based variants, confirming the necessity of vertex-connectivity constraints for high-quality analytical queries. (Section 6.3)

2 Preliminaries

In this paper, we consider an undirected graph $G = (V, E)$, where V and E denote the vertex and edge sets, respectively, with $|V| = n$ and $|E| = m$. For any subset $S \subseteq V$, let $G[S]$ denote the subgraph of G induced by S , i.e., $G[S] = (S, \{(u, v) \in E \mid u, v \in S\})$. Given an arbitrary vertex $v \in V$ and a vertex set S , we denote by $N(v, S) = \{u \in S \mid (u, v) \in E\}$ the set of neighbors of v in S , and by $d(v, S) = |N(v, S)|$ the corresponding degree. All subgraphs in this paper refer to *vertex-induced* subgraphs. For a subgraph g of G , we use $V(g)$ and $E(g)$ to denote its vertex and edge sets. The *diameter* of a graph is the maximum distance between any pair of vertices. A *degeneracy ordering* of G is a sequence of vertices $v_1, v_2, \dots, v_n$ such that each vertex v_i has the minimum degree in $G[v_i, v_{i+1}, \dots, v_n]$ [5]. The degeneracy of G , denoted by δ , is the maximum degree in a degeneracy ordering, i.e., $\delta = \max_{i \in [1, n]} \{d(v_i, G[\{v_i, \dots, v_n\}])\}$.

Given a graph G , its vertex cut, denoted by Γ_G , is a vertex subset of $V(G)$ whose removal disconnects the graph or reduces it to a single vertex. Note that the vertex cut of a disconnected graph is the empty set. The *vertex connectivity* (VC) of a graph G , denoted by $\kappa(G)$, is defined as the cardinality of a minimum vertex cut of G , denoted by $\Gamma_G^{\min}$. In other words, $\kappa(G)$ represents the *smallest* number of vertices whose deletion disconnects G or makes it trivial. In this paper, we focus on the following dense subgraph model.

Definition 2.1 (Connectivity-based γ -Quasi-Clique (γ -CQC) [54]). Given a graph G and a real number $0 \leq \gamma \leq 1$, an induced subgraph g is a *connectivity-based γ -quasi-clique* iff $\kappa(g) \geq \gamma \cdot (|V(g)| - 1)$.

Example 2. Consider the graph in Figure 1(b) as G . Removing the set $\{v_1, v_3, v_4\}$ splits G into two components: $\{v_2\}$ and $\{v_5, v_6, v_7\}$, making it a valid vertex cut. However, the minimum vertex cut is

$\{v_4\}$, so $\kappa(G) = 1$. For $\gamma = 0.5$, G is not a 0.5-CQC since its VC is below the required $0.5 \cdot (7 - 1) = 3$. In contrast, the graph G' in Figure 1(c) is a 0.5-CQC because $\kappa(G') = 2 \geq 0.5 \cdot (5 - 1)$.

A larger value of γ implies a more cohesive γ -CQC. When $\gamma = 1$, the definition reduces to a clique. γ -CQCs exhibit two important properties. First, γ -CQCs are *non-hereditary*. That is, an induced subgraph of a γ -CQC does not necessarily remain a γ -CQC. For instance, the 0.5-CQC in Figure 1(c) contains an induced subgraph on $\{v_8, v_{10}, v_{11}\}$ that fails to satisfy the 0.5-CQC condition. Second, γ -CQCs with $\gamma \geq 0.5$ have bounded diameter, as stated in Lemma 2.2.

LEMMA 2.2. *For $\gamma \geq 0.5$, the diameter of a γ -CQC is at most 2.*

Due to space constraints, the detailed proofs in this paper are deferred to the Appendix of the technical report [1].

Motivated by Lemma 2.2, this paper focuses on γ -CQCs with $\gamma \geq 0.5$. The rationale for this restriction is threefold: (1) when $\gamma < 0.5$, the resulting γ -CQCs often exhibit insufficient cohesiveness and hold limited practical significance; (2) for $\gamma \geq 0.5$, γ -CQCs possess both structural cohesiveness and bounded diameters, which facilitates the design of efficient query algorithms; (3) this restriction aligns with established conventions, as most prior work on *degree-based γ -quasi-cliques*¹ (γ -DQC) and *edge-based γ -quasi-cliques*² (γ -EQC) have also adopted $\gamma \geq 0.5$ as a standard setting [17, 35, 46, 48, 59, 64]. We formulate the problem as follows.

PROBLEM DEFINITION (MCQC QUERY PROCESSING). *Given a graph G and a parameter $0.5 \leq \gamma \leq 1$, the maximum γ -CQC (MCQC) query returns a γ -CQC in G with the maximum number of vertices.*

As proven in [54], MCQC is NP-hard for any $\gamma \in [0, 1]$. Additionally, we observe that the γ -CQC model imposes a strictly stronger structural constraint than both γ -DQC and γ -EQC.

LEMMA 2.3. *Any γ -CQC is also a γ -DQC and a γ -EQC; however, the converse does not necessarily hold.*

Lemma 2.3 also implies that existing query algorithms for maximum γ -DQC or γ -EQC cannot be adapted to answer MCQC queries, necessitating a dedicated exact solution.

3 Baseline Query Methods

In this section, we analyze two baseline strategies for processing the MCQC query. We first briefly review the state-of-the-art solution based on *mixed integer programming* (MIP). We then formalize a canonical *Branch-and-Bound* (BB) method, which serves as a baseline for our algorithmic optimizations. Finally, we analyze their efficiency bottlenecks to motivate our advanced approach.

MIP-based baseline. To the best of our knowledge, the only existing solution for the MCQC query is the *Exact Iterative Algorithm* (EIA) [54]. EIA builds upon a series of MIP formulations³ and cannot handle large-scale graphs; e.g., the largest graph used in [54] contains only 2,112 vertices and 2,203 edges. Since MIP techniques are not the focus of this paper, we defer the detailed description of EIA to the related work section (Section 7).

BB-based baseline. Most state-of-the-art exact methods for querying the maximum γ -DQC and γ -EQC rely on the BB (a.k.a. backtracking) framework [46, 47, 59, 64]. Compared to MIP solvers, BB techniques allow for problem-specific pruning, making them more suitable for graph query processing. Consequently, *all methods proposed in this paper are developed within the BB framework*.

¹Given a graph G and $0 \leq \gamma \leq 1$, an induced subgraph g of G is a *degree-based γ -quasi-clique* (γ -DQC) if $\forall v \in V(g)$, $d(v, V(g)) \geq \gamma \cdot (|V(g)| - 1)$.

²Given a graph G and $0 \leq \gamma \leq 1$, an induced subgraph g of G is an *edge-based γ -quasi-clique* (γ -EQC) if $|E(g)| \geq \gamma \cdot (|V(g)|(|V(g)| - 1)/2)$.

³For completeness, MIP formulations are provided in Appendix B of the technical report [1].

Algorithm 1: BB-based baseline method: MCQC-Enum

Input: A graph $G = (V, E)$ and a real number $0.5 \leq \gamma \leq 1$
Output: A maximum connectivity-based γ -quasi-clique g^*

- 1 $g^* \leftarrow$ compute a degeneracy-based clique in G ;
- 2 Iteratively remove vertices from G based on Lemma 3.1;
- 3 Enum-Rec($\emptyset, V$); **return** g^* ;

Procedure: Enum-Rec(S, C)

- 4 **if** $C = \emptyset$ **or** $|S| + |C| \leq |V(g^*)|$ **then return**;
- 5 **if** $|S| > |V(g^*)|$ **and** $G[S]$ is a γ -CQC **then** $g^* \leftarrow G[S]$;
- 6 $v^* \leftarrow \arg \max_{v \in C} d(v, S)$; // Ties are broken by global degree
- 7 Enum-Rec($S \cup \{v^*\}, C \setminus \{v^*\}$);
- 8 Enum-Rec($S, C \setminus \{v^*\}$);

Specifically, BB recursively solves a problem instance through a process called *branching*, which partitions the current search space into multiple sub-instances. Each sub-instance, or *branch*, is typically represented by a pair of disjoint vertex sets (S, C) . Here, S is the selected set, which induces a partial solution that must be included in any γ -CQC within the branch; C is the candidate set, which consists of vertices that can potentially extend S to form a larger γ -CQC. Solving a problem instance (or branch) (S, C) means finding the largest γ -CQC in the branch. A γ -CQC g is said to be in a branch (S, C) only if it (1) contains all vertices in S and (2) is a subgraph of $G[S \cup C]$, i.e., $S \subseteq V(g) \subseteq S \cup C$. Obviously, solving the branch $(\emptyset, V)$ finds the maximum γ -CQC in graph G .

A crucial component of BB is its *branching strategy*, which determines how a branch is decomposed into sub-branches. In this paper, all our approaches adopt a natural *Binary Branching with Pivot Selection* (BBPS) strategy, where the *pivot vertex* refers to the key element used during the branching process. Given a branch $B = (S, C)$, BBPS selects a pivot vertex $v^* \in C$ and splits the search space depending on whether v^* is included in the γ -CQCs being explored. This recursive process continues until the maximum γ -CQC within the branch is identified. Since the pivot selection rule governs the size and quality of the resulting search tree, it plays a central role in BB performance. Our BB-based baseline adopts a simple greedy rule: it selects from C the vertex with the *largest number of neighbors in S* . The intuition is that denser partial solutions are more likely to satisfy the VC requirement of γ -CQCs.

Another key component of BB is *reduction rule*, which eliminates candidate vertices from C that cannot be part of the optimal solution. However, due to the non-hereditary of γ -CQCs, determining which candidate vertices are absent from the maximum γ -CQC is highly non-trivial. Thus, our BB-based baseline does not incorporate any effective reduction rule.

Before proceeding, we first introduce a useful lemma for designing the BB-based baseline method.

LEMMA 3.1. *Given a graph G and a vertex $v \in V$, if $d(v, V) < \lceil lb \cdot \gamma \rceil$, where lb is the size of the largest γ -CQC found so far, then v cannot belong to any γ -CQC of size larger than lb .*

We then present the BB-based baseline, MCQC-Enum, in Algorithm 1. First, it applies an intuitive heuristic to identify a clique, which is always a qualified γ -CQC, and then shrinks the input graph according to Lemma 3.1 (Lines 1–2). The *degeneracy-based clique* refers to the longest suffix of the *degeneracy ordering* that forms a clique [10]. Then, MCQC-Enum invokes the procedure Enum-Rec to recursively search the maximum γ -CQC (Line 3). This procedure has termination conditions (Line 4), solution update rules (Line 5), the pivot selection method (Line 6), and branching steps (Lines 7–8).

Example 3. Consider the graph in Figure 1(b) as G , with $\gamma = 0.75$. Following MCQC-Enum, a possible degeneracy ordering of G is $\langle v_1, v_2, v_3, v_4, v_5, v_6, v_7 \rangle$. The longest suffix of this ordering that forms a clique is $\{v_4, v_5, v_6, v_7\}$. Therefore, MCQC-Enum first initializes the optimal solution g^* with the subgraph induced by this suffix. Since no vertex in G has a degree less than $\lceil 4 \cdot 0.75 \rceil = 3$ (based on Lemma 3.1), no vertices can be pruned initially, and MCQC-Enum proceeds to invoke its branch-and-bound procedure Enum-Rec. Since the initial selected set is $S = \emptyset$, the number of neighbors in S for all candidate vertices is 0; therefore, Enum-Rec selects v_4 , which has the maximum global degree, as the pivot vertex v^* . Enum-Rec then generates two sub-branches: one where v_4 is included in the selected set, and another where v_4 is removed from the branch. Through a similar recursive exploration, Enum-Rec will eventually determine that no solution larger than $\{v_4, v_5, v_6, v_7\}$ exists, thus returning $\{v_4, v_5, v_6, v_7\}$ as the maximum 0.75-CQC.

Limitations. Both baselines exhibit fundamental limitations that necessitate a more specialized query method. For EIA, the MIP formulation translates the graph topology into complex linear inequalities. By relying on generic MIP solvers, EIA fails to exploit problem-specific properties (e.g., vertex cut) for efficient traversal, incurring prohibitive computational costs on large graphs. For MCQC-Enum, the core challenge lies in designing effective reduction rules. Due to the *non-hereditary* property of γ -CQCs, a partial solution S that violates the requirement of γ -CQC does not imply that its supersets will also violate it. Consequently, MCQC-Enum lacks effective reduction rules to prune unpromising branches early, forcing an exhaustive enumeration.

Time complexity analysis. Both methods suffer from a trivial worst-case time complexity of $O^*(2^n)$ (analysis omitted for brevity, see Appendix B of the technical report [1]). These scalability bottlenecks motivate the design of our advanced query processing approach, MCQC-TD (detailed in Section 4), which integrates a top-down search strategy with structural reduction rules to break the efficiency barrier.

4 MCQC-TD: Our Advanced Approach

In this section, we present MCQC-TD, a practically effective branch-and-bound framework tailored for the exact MCQC query. The efficiency of MCQC-TD stems from three tightly coupled components, anchored by a **top-down framework** (Section 4.1). Unlike traditional bottom-up enumeration, this framework iteratively searches for the maximum γ -CQC by maintaining a dynamic size upper bound, thereby imposing a strict target size constraint in each iteration. Leveraging this specific size constraint, we derive a novel equivalent definition of γ -CQCs, which enables a **new reduction rule (RR)** (Section 4.2) to safely prune candidate vertices that violate the connectivity requirement of the target size. Critically, to maximize the invocation of **RR**, we design a **hybrid pivot selection (HPS)** strategy (Section 4.3). **HPS** dynamically guides the branching process to expose pruning opportunities at the earliest possible stage, ensuring that the theoretical pruning power of **RR** is fully realized during the search. Finally, in Section 4.4, we integrate these components and formally prove that MCQC-TD achieves a non-trivial worst-case time complexity of $O^*(\lambda_t^n)$, where $\lambda_t < 2$ depends on the graph degeneracy. For instance, when $t = 1, 2$, and 3 , λ_t equals 1.6181, 1.8393, and 1.9276, respectively.

4.1 Overview and Motivations

We note that the baseline method MCQC-Enum identifies the maximum γ -CQC in a *bottom-up* manner, which maintains the largest γ -CQC seen so far (as the lower bound on the size of the maximum γ -CQC) and recursively searches for a larger one. As a result, it will inevitably explore a large number of small γ -CQCs, thereby introducing redundant computations. To solve this, we propose a *top-down* framework called MCQC-TD. Conversely, it maintains an upper bound ub on the size of

Algorithm 2: Our advanced BB method: MCQC-TD

Input: A graph $G = (V, E)$ and a real number $0.5 \leq \gamma \leq 1$
Output: A maximum connectivity-based γ -quasi-clique g^*

```

1  $g^* \leftarrow$  compute a degeneracy-based clique in  $G$ ;
2 Iteratively remove vertices from  $G$  based on Lemma 3.1;
3  $ub \leftarrow 1 + \lceil \delta/\gamma \rceil$ ;
4 while  $|V(g^*)| < ub$  do
5   Refine  $G$  to  $G'$  similar to Lemma 3.1;
6   TD-Rec( $\emptyset, V(G'), ub$ );
7    $ub \leftarrow ub - 1$ ;
8 return  $g^*$ ;

Procedure: TD-Rec( $S, C, s$ )
/* Upper-bound-based reduction rule (Section 4.2) */
9 Refine the candidate set  $C$  via RR;
/* New termination criterion (Section 4.2) */
10 Terminate the branch via Lemma 4.5;
11 if  $|S| = s$  then
12   if  $G[S]$  is a  $\gamma$ -CQC then  $g^* \leftarrow G[S]$ ;
13   return;
14 if  $|S \cup C| < s$  then return;
/* Pivot-based branching (Section 4.3) */
15  $v^* \leftarrow$  select a pivot by a hybrid pivot selection method;
16 TD-Rec( $S \cup \{v^*\}, C \setminus \{v^*\}, s$ );
17 TD-Rec( $S, C \setminus \{v^*\}, s$ );

```

the maximum γ -CQC, then iteratively searches for a γ -CQC of size ub and decreases ub by 1 until such a γ -CQC is found. Clearly, when MCQC-TD terminates, the maximum γ -CQC can be found. Compared to MCQC-Enum, MCQC-TD prioritizes searching for large γ -CQCs and thus reduces the redundant exploration of smaller γ -CQCs. Specifically, when searching for a large γ -CQC of size ub (via a BB procedure similar to Enum-Rec), the newly incorporated upper bound ub will allow us to significantly narrow down the search space by (1) removing those unpromising vertices that cannot appear in any γ -CQCs with size at least ub (Section 4.2) and (2) designing an effective branching strategy for pruning unpromising branches that cannot yield a sufficiently large γ -CQC (Section 4.3). Below, we elaborate on the details.

We summarize MCQC-TD in Algorithm 2, which first initializes the upper bound ub at Line 3 based on the following lemma.

LEMMA 4.1. *Given a graph $G = (V, E)$, the size of the maximum γ -CQC in G is bounded by $1 + \lceil \delta/\gamma \rceil$, where δ is the degeneracy of G .*

Then, MCQC-TD runs in multiple rounds (Lines 4-7), and each round aims to search for a γ -CQC of size equal to ub . Specifically, it refines the input graph G by removing from G all vertices v with a degree smaller than $\lceil (ub - 1) \cdot \gamma \rceil$, formally, $d(v, V) < \lceil (ub - 1) \cdot \gamma \rceil$. This is because those vertices cannot appear in any γ -CQC of size at least ub (which can be proved analogously to Lemma 3.1). Finally, it uses a BB procedure called TD-Rec to find a γ -CQC of size equal to ub from the refined graph (Line 6).

TD-Rec vs. Enum-Rec. TD-Rec also follows a BB framework, and Enum-Rec can be easily adapted to find a γ -CQC of size equal to ub by using the same termination condition (Lines 11-14) as TD-Rec. We remark that TD-Rec distinguishes itself from Enum-Rec in two aspects: First, TD-Rec employs our newly proposed upper-bound-based reduction **RR** to prune unpromising vertices from the candidate set C that cannot appear in any γ -CQC of size ub (details in Section 4.2). It also uses a new termination criterion for stopping the branching of those branches that can be easily solved (Section 4.2). Second, TD-Rec employs a carefully-designed pivot selection strategy for further unleashing the pruning power of **RR** (details in Section 4.3). As a result, TD-Rec significantly outperforms Enum-Rec in both theory and practice. We show that TD-Rec will generate theoretically fewer branches than Enum-Rec and thus achieves lower worst-case time complexity (details in Section 4.4).

4.2 Reduction Rule and Termination Criterion

Consider the BB procedure TD-Rec in MCQC-TD, which aims to find a γ -CQC of size s . Let $B = (S, C, s)$ be a branch in the recursion of TD-Rec. To narrow down the search space, it is essential to refine the candidate set C by removing unpromising vertices that cannot belong to any γ -CQC of size s . However, the non-hereditary property makes it challenging to determine whether a vertex v can be part of such a γ -CQC within the branch B . We observe that the newly incorporated upper bound s will help to derive a necessary condition for a vertex to appear in a γ -CQC within B . Based on this, we propose a new reduction rule **RR** to safely prune those vertices in C that violate the condition. Furthermore, we develop a new termination criterion to terminate the branching early. Below, we provide the details.

Let g be an arbitrary graph. We define its **cut-based vertex partition** by two vertex sets, namely $P(g)$ and $Q(g)$, formally: (1) $P(g) \cup Q(g) = V(g)$, $P(g) \cap Q(g) = \emptyset$ and (2) $P(g)$ is a vertex cut of g (i.e., its removal disconnects the graph g). Clearly, graph g admits multiple cut-based vertex partitions. Thus, we use the notation $P_i(g)$ and $Q_i(g)$ ($i > 0$) to distinguish different cut-based vertex partitions. Among all those cut-based vertex partitions, let $\alpha(g)$ be the maximum size of the $Q(g)$ subset, formally,

$$\alpha(g) := \max\{|Q(g)| \mid V(g) \setminus Q(g) \text{ is a vertex cut of } g\}.$$

Note that $\alpha(g)$ corresponds to the structural indicator value in Figure 2(b). Recall that $\kappa(g)$ (the vertex connectivity of g) refers to the size of the minimum vertex cut of g , which also corresponds to the minimum size of $P(g)$. Therefore, we can easily derive that

$$\alpha(g) = |V(g)| - \kappa(g). \quad (1)$$

Furthermore, consider a graph g_{sup} such that g is a subgraph of g_{sup} , i.e., $V(g_{sup}) \supseteq V(g)$. We have

$$\alpha(g) \leq \alpha(g_{sup}). \quad (2)$$

This can be verified based on the fact that the vertex set $T = P(g) \cup (V(g_{sup}) \setminus V(g))$ is a vertex cut set of g_{sup} (note that the removal of T reduces g_{sup} to $g \setminus P(g)$ which is clearly disconnected) and thus $\kappa(g_{sup}) \leq |T| = \kappa(g) + (|V(g_{sup})| - |V(g)|)$. Together with Equation (1), we can derive Equation (2), i.e., $\alpha(g) - \alpha(g_{sup}) = |V(g)| - |V(g_{sup})| + \kappa(g_{sup}) - \kappa(g) \leq 0$.

We summarize the above observations in the following lemma.

LEMMA 4.2. *Given a subgraph g , we have (1) $\alpha(g) = |V(g)| - \kappa(g)$ and (2) $\alpha(g) \leq \alpha(g_{sup})$ where g is a subgraph of g_{sup} .*

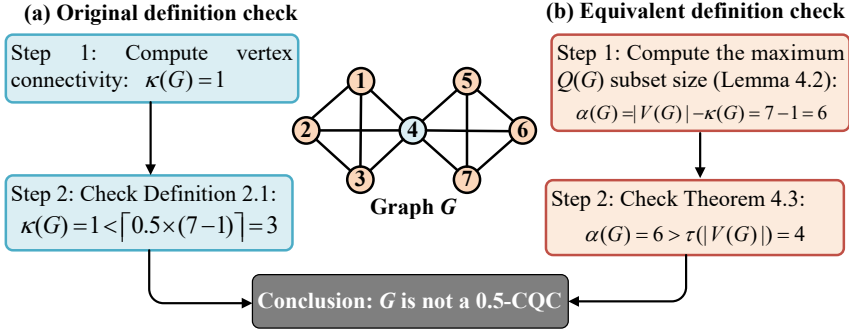


Fig. 3. Comparison of the workflows for determining whether a graph is a 0.5-CQC based on the original definition versus the proposed equivalent definition.

Recall that for a γ -CQC g , its vertex connectivity $\kappa(g)$ is at least $\gamma \cdot (|V(g)| - 1)$. Equivalently, we can derive that $\alpha(g)$ is at most $|V(g)| - \gamma \cdot (|V(g)| - 1)$, formally,

$$\alpha(g) \leq |V(g)| - \gamma \cdot (|V(g)| - 1) = \lfloor (1 - \gamma) \cdot (|V(g)| - 1) \rfloor + 1. \quad (3)$$

To ease the presentation, we define the following function.

$$\tau(x) = \lfloor (1 - \gamma) \cdot (x - 1) \rfloor + 1. \quad (4)$$

Then, we can derive an equivalent definition of γ -CQCs and summarize the above in the following theorem.

THEOREM 4.3 (EQUIVALENT DEFINITION OF γ -CQCs). *A graph g is a γ -CQC if and only if there does not exist a $Q(g)$ subset whose size exceeds $\tau(|V(g)|)$, formally,*

$$\alpha(g) \leq \tau(|V(g)|). \quad (5)$$

Example 4. To illustrate, consider the example in Figure 3, which shows the validation of a 0.5-CQC G . According to Definition 2.1, Figure 3(a) has the vertex connectivity of G as $\kappa(G) = 1$, which is less than the required value $0.5 \cdot \lceil |V(G)| - 1 \rceil = 3$. Therefore, G does not satisfy the original 0.5-CQC definition. According to Theorem 4.3, Figure 3(b) has the size of the largest $Q(G)$ subset (as indicated by those orange vertices) as $\alpha(G) = |V(G)| - \kappa(G) = 6$, which exceeds $\tau(|V(G)|) = 4$. Thus, G is not a 0.5-CQC.

New reduction rule. Now, we are ready to derive the necessary condition. Consider a γ -CQC g within B . Clearly, we have $S \subseteq V(g)$ and $|V(g)| \leq s$ (note that s is the upper bound on the size of the maximum γ -CQC, otherwise MCQC-TD terminates without invoking TD-Rec). Consider a vertex v in g (say, v appears in a γ -CQC within the branch). According to Equation (2) and Lemma 4.2, we have

$$\alpha(G[S \cup \{v\}]) \leq \alpha(g) \leq \tau(|V(g)|) \leq \tau(s). \quad (6)$$

We note that the function $\tau(\cdot)$ is non-decreasing, thus $\tau(|V(g)|) \leq \tau(s)$ holds. Note that $\tau(s)$ corresponds to the pruning threshold in Figure 2(b). Based on the above, we can easily derive the necessary condition for a vertex v to appear in a γ -CQC as follows.

$$\textbf{Condition: } \alpha(G[S \cup \{v\}]) \leq \tau(s). \quad (7)$$

Clearly, for a vertex v in C that violates the above condition, i.e., $\alpha(G[S \cup \{v\}]) > \tau(s)$, we can derive that v cannot appear in any γ -CQC within the branch by contradiction. Therefore, we obtain a new reduction rule as follows.

selection. The second one finds a pivot quickly but is only applicable in certain cases, thereby lacking the theoretical guarantee (on the worst-case time complexity of TD-Rec) in general. To combine the strengths of both methods, we propose a hybrid strategy **HPS** that identifies a pivot efficiently while providing the theoretical guarantee on the worst-case time complexity. Below, we provide the details.

Consider the branching procedure at a branch $B = (S, C, s)$. Let $g_b = G[S \cup C]$ throughout this section for simplicity. We have:

LEMMA 4.6. *Given a branch $B = (S, C, s)$, let $P(g_b)$ and $Q(g_b)$ be a cut-based vertex partition of g_b . If the vertices in S are distributed across multiple connected components of $G[Q(g_b)]$, then $P(g_b) \cap S$ is a vertex cut of $G[S]$.*

Example 6. Consider Figure 1(b). Suppose $S = \{v_3, v_4, v_5\}$, $C = \{v_1, v_2, v_6, v_7\}$, and $P(g_b) = \{v_4\}$. Since (1) $G[S \cup C \setminus P(g_b)]$ consists of two connected components ($\{v_1, v_2, v_3\}$ and $\{v_5, v_6, v_7\}$), and (2) v_3 and v_5 from S are distributed across two distinct components, Lemma 4.6 indicates that $P(g_b) \cap S = \{v_4\}$ is a vertex cut of $G[S]$.

Recall that $g_b = G[S \cup C]$. Given a specific cut-based vertex partition of g_b , then Lemma 4.6 enables us to systematically increase the value of $\alpha(G[S])$ (i.e., the size of the maximum $Q(G[S])$ subset) by selecting vertices from $Q(g_b)$ as pivots. This is because $P(g_b) \cap S$ forms a vertex cut of $G[S]$ (i.e., the $P(G[S])$ subset) and $P(g_b) \cap Q(g_b) = \emptyset$. Once $\alpha(G[S])$ reaches the threshold $\tau(s)$, **RR** can then be invoked to remove the remaining vertices in $Q(g_b)$ that are not in S , i.e., $Q(g_b) \cap C$. However, to ensure the theoretical performance of **RR** (i.e., $Q(g_b) \cap C$ is always non-empty), a crucial prerequisite is that there exists a cut-based vertex partition whose $Q(g_b)$ subset has a size that exceeds $\tau(s)$. Fortunately, our new termination criterion (Lemma 4.5) guarantees the existence of such a partition. Specifically, we define

$$P_1(g_b) := \Gamma_{g_b}^{\min} \quad \text{and} \quad Q_1(g_b) := V(g_b) \setminus P_1(g_b), \quad (8)$$

where $\Gamma_{g_b}^{\min}$ is an arbitrary minimum vertex cut of g_b . According to Lemma 4.5, as long as a target γ -CQC with size s is not directly extracted from the branch, $|Q_1(g_b)| = \alpha(g_b) > \tau(s)$ must hold.

To use Lemma 4.6, our first pivot selection method is as follows.

Cut-Deprioritized Pivot Selection (CDPS) method:

- **Case 1.** If the vertices in S are distributed across zero or multiple components of the subgraph $G[Q_1(g_b)]$, then the pivot is chosen as any vertex from $Q_1(g_b) \cap C$. Note that the set $Q_1(g_b) \cap C$ is not empty; otherwise, it would mean $Q_1(g_b) \subseteq S$. Recall that $|Q_1(g_b)| > \tau(s)$, so $\alpha(G[S]) > \tau(s)$ at this point, and thus we can terminate the branch because it cannot contain any γ -CQC of size s according to Theorem 4.3 and Equation (2).
- **Case 2.** If Case 1 does not hold, i.e., the vertices in S are confined to exactly one component of $G[Q_1(g_b)]$, then the pivot is chosen as any vertex from $Q_1(g_b) \cap C$ that *does not belong to the same component as S in $G[Q_1(g_b)]$* . Such a pivot must exist since there are at least two connected components in $G[Q_1(g_b)]$, and S entirely located within one of them, the remaining components outside the one containing S must belong to $Q_1(g_b) \cap C$.

CDPS guarantees the theoretical performance of **RR** by prioritizing vertices in $Q_1(g_b)$ as pivots. Specifically, in the worst case where the selected set S is either empty or entirely contained in the vertex cut $P_1(g_b)$, CDPS selects pivots from $Q_1(g_b) \cap C$. By Lemma 4.6, a vertex cut of $G[S]$ is given by $P_1(g_b) \cap S$. Consequently, the pivots chosen by CDPS never belong to this cut and therefore can be used to increase the size of the largest $Q(G[S])$ subset. Thus, CDPS requires at most $\tau(s)$ iterations to ensure $\alpha(G[S]) \geq \tau(s)$ for any branch $B = (S, C, s)$. However, CDPS necessitates computing the minimum vertex cut, incurring a time complexity of $O(n^{2.5}m)$ [58]. This cost is

impractical, especially given the exponential number of pivot selection steps in theory. To overcome this dependency on the expensive partitioning method in Equation (8), we seek a more efficient approach that achieves an equivalent partitioning effect.

LEMMA 4.7. *Given a branch $B = (S, C, s)$, if there exists a vertex v whose degree is smaller than $|S \cup C| - \tau(s)$, then we define*

$$P_2(g_b) := N(v, S \cup C) \quad \text{and} \quad Q_2(g_b) := V(g_b) \setminus P_2(g_b),$$

where $Q_2(g_b)$ also has cardinality exceeding $\tau(s)$.

Lemma 4.7 enables the following pivot selection method.

Degree-Deprioritized Pivot Selection (DDPS) method. Given a branch $B = (S, C, s)$, suppose the vertex of minimum degree in B is v . If we have $d(v, S \cup C) < |S \cup C| - \tau(s)$, then:

- **Case 1.** If $v \in S$, then the pivot is any vertex from $Q_2(g_b) \cap C$. Note that $Q_2(g_b) \cap C$ is not empty, as otherwise we could derive that $\alpha(G[S]) > \tau(s)$ and terminate the branch.
- **Case 2.** If $v \in C$, then the pivot is chosen as v itself.

Whenever applicable (i.e., the minimum degree is less than $|S \cup C| - \tau(s)$), DDPS guarantees the same theoretical performance of **RR** as CDPS. As with CDPS, in the worst-case scenario, S is either empty or entirely consists of vertices in $P_2(g_b)$. In both cases, all pivots selected by DDPS belong to $Q_2(g_b) \cap C$ and thus are disjoint from the vertex cut $P_2(g_b) \cap S$ of $G[S]$ (i.e., the neighbor set of v in S). Thus, these pivots can be used to increase $\alpha(G[S])$, and DDPS also requires at most $\tau(s)$ iterations to ensure $\alpha(G[S]) \geq \tau(s)$ and then trigger **RR**. Moreover, the time complexity of DDPS for each pivot selection is only $O(n)$, which is substantially lower than that of CDPS. However, the applicability of DDPS is inherently limited due to the minimum degree constraint. In contrast, although CDPS incurs higher computational overhead, it remains universally applicable. To balance this trade-off, we naturally propose a hybrid pivot selection (**HPS**) method that combines the strengths of both, thereby ensuring theoretical soundness and practical efficiency.

Hybrid Pivot Selection (HPS) method. Assuming the vertex of minimum degree in branch B is v , the **HPS** method is as follows.

- **Case 1.** If $d(v, S \cup C) < |S \cup C| - \tau(s)$, DDPS is applied.
- **Case 2.** If $d(v, S \cup C) \geq |S \cup C| - \tau(s)$, CDPS is applied.

Since both DDPS and CDPS (when applicable) can ensure the theoretical performance of **RR**, it is easy to draw the conclusion that **HPS** can also achieve the corresponding theoretical guarantee.

Time complexity analysis. As discussed previously, in terms of single pivot selection complexity, CDPS requires $O(n^{2.5}m)$, while DDPS requires $O(n)$. Furthermore, it is evident that the worst-case complexity of **HPS** depends on CDPS, meaning it is also $O(n^{2.5}m)$.

4.4 Summary and Analysis

In this subsection, we integrate the techniques proposed in Sections 4.2 and 4.3 into the branching procedure TD-Rec, yielding the complete version of TD-Rec (Algorithm 3). We then provide a comprehensive analysis of MCQC-TD (Algorithm 2) with Algorithm 3.

The complete version of Procedure TD-Rec. As shown in Algorithm 3, TD-Rec first removes candidate vertices that satisfy **RR** and then tries to terminate the recursion (Lines 1-5). If no termination condition is met, TD-Rec proceeds to the **HPS**-based branching stage (Lines 6-16). A vertex v with the minimum degree in the branch is first selected (Line 6), and the pivot vertex v^* is chosen based on the degree of v . That is, if the degree of v satisfies the condition specified in Lemma 4.7, TD-Rec uses the DDPS method (Lines 7-10); otherwise, it uses the CDPS method (Lines 11-15). Two sub-branches are generated and recursively solved (Lines 16-17).

Algorithm 3: The complete version of Procedure TD-Rec

Input: A selected set S , a candidate set C , and a target size s
Output: A γ -CQC of size s , if one exists; otherwise $\emptyset$

```

1 Remove candidate vertices according to RR;
  /* New termination criterion */
2 if  $\alpha(G[S \cup C]) \leq \tau(s)$  then
3   if  $|S \cup C| \geq s$  then  $g^* \leftarrow$  extract a subgraph by Lemma 4.5 ;
4   return;
5 if  $|S| = s$  or  $|S \cup C| < s$  then return;
6  $v \leftarrow \arg \min_{u \in S \cup C} d(u, S \cup C)$ ;  $g_b \leftarrow G[S \cup C]$ ;
7 if  $d(v, S \cup C) < |S \cup C| - \tau(s)$  then
  /* Degree-deprioritized pivot selection method */
  8  $P_2(g_b) \leftarrow N(v, S \cup C)$ ;  $Q_2(g_b) \leftarrow V(g_b) \setminus P_2(g_b)$ ;
  9 if  $v \in S$  then  $v^* \leftarrow$  any vertex from  $Q_2(g_b) \cap C$ ;
  10 else  $v^* \leftarrow v$ ;
11 else
  /* Cut-deprioritized pivot selection method */
  12  $P_1(g_b) \leftarrow \Gamma_{g_b}^{\min}$ ;  $Q_1(g_b) \leftarrow V(g_b) \setminus P_1(g_b)$ ;
  13 if  $S$  is confined to one component of  $G[Q_1(g_b)]$  then
  14    $v^* \leftarrow$  any vertex from  $Q_1(g_b) \cap C$  that does not belong to the connected component
    containing  $S$  in  $G[Q_1(g_b)]$ ;
  15 else  $v^* \leftarrow$  any vertex from  $Q_1(g_b) \cap C$ ;
16 TD-Rec( $S \cup \{v^*\}, C \setminus \{v^*\}, s$ );
17 TD-Rec( $S, C \setminus \{v^*\}, s$ );

```

Example 7. We provide an example of MCQC-TD in Appendix C of the technical report [1].

Correctness proof (sketch). MCQC-TD is an exact method guaranteed to identify the maximum γ -CQC, based on two core properties. First, MCQC-TD utilizes a top-down search strategy that systematically narrows the solution space by commencing from the size upper bound of the maximum γ -CQC and decrementing until the true maximum is located. Second, each iteration executes a complete branch-and-bound procedure. Critically, pruning is only performed via reduction rule **RR**, ensuring that only vertices provably incapable of participating in any target γ -CQCs are removed from the candidate set. The combination of this exhaustive, size-decreasing search and the soundness of the **RR** pruning guarantees that MCQC-TD will discover a maximum γ -CQC, thereby confirming its correctness. A formal proof is provided in Appendix C of the technical report [1].

Time complexity analysis. The analysis is shown in Theorem 4.8.

THEOREM 4.8. *The time complexity of MCQC-TD is $O(n^{3.5} m \lambda_t^n)$, where $\lambda_t < 2$ is the largest real root of the equation $x^{t+2} - 2x^{t+1} + 1 = 0$, $t = \tau(1 + \lceil \delta/\gamma \rceil)$, and δ is the degeneracy of the graph. For example, when $t = 1, 2$, and 3 , λ_t are 1.6181, 1.8393, and 1.9276, respectively.*

PROOF SKETCH. The complexity of MCQC-TD is dominated by its top-down search phase. MCQC-TD performs at most n iterations, and the cost per iteration is determined by the recursive procedure TD-Rec. Intuitively, a naive branch-and-bound algorithm explores a nearly unconstrained binary search tree where each step creates two subproblems that shrink the candidate set by only one vertex, inevitably leading to a trivial $O(2^n)$ branches. In contrast, our **HPS** ensures that within at most $t = \tau(s)$ consecutive branching steps, the structural indicator reaches the pruning threshold.

At this point, **RR** is triggered, safely pruning at least one additional vertex from the candidate set. This breaks the trivial branching pattern, ensuring that at least one branch shrinks by more than one vertex. Mathematically, this yields a constrained recurrence relation, bounding the maximum number of recursive calls to $O(\lambda_t^n)$, where $\lambda_t < 2$ is the largest positive real root of $x^{t+2} - 2x^{t+1} + 1 = 0$. Furthermore, the complexity of each iteration is $O(n^{2.5}m)$. Thus, the overall complexity of MCQC-TD is $O(n^{3.5}m\lambda_t^n)$. $\square$

Remark. In contrast to the baseline methods introduced in Section 3, which exhibit a trivial worst-case complexity of $O^*(2^n)$, MCQC-TD achieves an improved complexity of $O^*(\lambda_t^n)$, where $\lambda_t < 2$. This performance gain stems directly from the synergy among the top-down strategy (Section 4.1), the new reduction rule **RR** (Section 4.2), and the pivot selection method **HPS** (Section 4.3). By effectively exploiting the equivalent definition of γ -CQCs (Theorem 4.3), these techniques allow MCQC-TD to prune candidate vertices that cannot be part of any target γ -CQC, thus theoretically avoiding the exhaustive enumeration required by the baseline approaches. A detailed comparison between the state-of-the-art method EIA and our proposed MCQC-TD is provided in Appendix C of the technical report [1].

5 Query Optimization Techniques

Although MCQC-TD achieves a non-trivial theoretical complexity, minimizing query time on massive graphs requires further optimization. In this section, we introduce two query optimization techniques: (1) an optimized initial solution construction strategy to tighten the lower bound for aggressive pruning (Section 5.1); and (2) a diameter-bounded query decomposition framework to enable scalable processing via problem partitioning (Section 5.2).

5.1 Optimized Initial Solution Construction

A tight lower bound (i.e., a large initial γ -CQC) is critical for query efficiency, as it enables MCQC-TD to maximize the effectiveness of reduction rules (e.g., Lemma 3.1). However, the basic initialization method (Line 1 of Algorithm 2), which relies on a degeneracy-based clique, often yields loose bounds due to the rigid connectivity requirement of cliques. To address this, we propose MCQC-Heu, a tailored optimizer that constructs high-quality initial solutions by exploiting local graph structures.

MCQC-Heu improves the quality of the initial solution through two key mechanisms. First, it employs a *multi-view ordering strategy*. While the basic method relies on a single global degeneracy ordering, we observe that structural cohesiveness varies significantly across different granularities. For instance, in Figure 1(b), the global degeneracy ordering may rank v_1 before v_5 , whereas in the local subgraph induced by $\{v_1, \dots, v_5, v_7\}$, their relative importance might be reversed. This indicates that relying on a single view limits the discovery of dense local substructures. To overcome this, MCQC-Heu adopts a two-phase approach: it first scans the global graph, and then iteratively refines the search within the *ego-network* (i.e., the 1-hop neighborhood) of each vertex. This locality-aware exploration exposes dense regions hidden in the global topology. Second, to avoid expensive connectivity computations during initialization, MCQC-Heu utilizes a *lightweight sufficiency check*:

LEMMA 5.1. *Let $d_{\min}$ denote the minimum degree of a graph g . If $2(d_{\min} + 1) - |V(g)| \geq \gamma \cdot (|V(g)| - 1)$, then g is a γ -CQC.*

This lemma allows MCQC-Heu to quickly identify valid γ -CQCs that are larger than cliques, providing a tighter lower bound than the basic method to initialize the MCQC-TD algorithm.

Details of MCQC-Heu. Algorithm 4 summarizes MCQC-Heu. Due to page limits, we defer its description to Appendix D of the technical report [1].

Example 8. We provide an example of MCQC-Heu in Appendix D of the technical report [1].

Algorithm 4: Optimized initialization method: MCQC-Heu**Input:** A graph $G = (V, E)$ and a real number $0.5 \leq \gamma \leq 1$ **Output:** A large heuristic connectivity-based γ -quasi-clique

```

1  $g \leftarrow \emptyset$ ; //  $g$  is maintained globally
2  $\text{Degen}(G)$ ; // Stage-1: Global View
3 Perform graph reduction on  $G$  based on Lemma 3.1;
4  $\text{Ego-Degen}(G)$ ; // Stage-2: Local View (Ego-networks)
5 return  $g$ ;

Procedure:  $\text{Degen}(G')$ 
6 Let  $(v_1, \dots, v_n)$  be the degeneracy ordering of  $G'$ ;
7 for  $i \leftarrow 1$  to  $n$  do
8   if  $2 \cdot (d(v_i, V(G')) + 1) - (n - i + 1) \geq \gamma \cdot (n - i)$  and  $n - i + 1 > |V(g)|$  then
      $g \leftarrow G'[V(G') \setminus \{v_1, \dots, v_{i-1}\}]$ ;

Procedure:  $\text{Ego-Degen}(G'')$ 
9 Let  $(v_1, \dots, v_n)$  be the degeneracy ordering of  $G''$ ;
10 for  $i \leftarrow 1$  to  $n$  do
11    $g' \leftarrow$  Extract the ego-network of  $v_i$  in  $G''$ , and remove the vertices from it that appear before  $v_i$  in
     the ordering;
12    $\text{Degen}(g')$ ;

```

5.2 Diameter-Bounded Query Decomposition

To ensure scalability on massive graphs, we introduce a query decomposition approach. This strategy is theoretically grounded in Lemma 2.2, which guarantees that any γ -CQC ($\gamma \geq 0.5$) has a diameter of at most 2. This strict locality constraint allows us to decompose the intractable global MCQC query into a series of independent and manageable local sub-queries without loss of correctness.

Local sub-query construction. Based on a given degeneracy ordering $\langle v_1, v_2, \dots, v_n \rangle$ of V , we partition the query processing into n independent tasks. For each vertex v_i , we extract a local subgraph $G_i = G[V_i]$, where V_i contains v_i and its 2-hop successors (vertices appearing after v_i in the ordering with distance ≤ 2). Since any target γ -CQC must be fully contained within the 2-hop neighborhood of its earliest vertex in the ordering, executing the query on all G_i collectively yields the exact global optimum. Crucially, each G_i is significantly smaller than G , enabling efficient processing through improved data locality.

Instance-specific reduction. Creating subgraphs alone is insufficient. We further apply instance-specific reduction rules to prune the materialized subgraph G_i before invoking the expensive enumeration procedure. Let lb be the current global lower bound.

- **Degree-based reduction rule:** Remove any vertex in G_i with degree $< \lceil lb \cdot \gamma \rceil$.
- **Triangle-based reduction rule:** Remove any edge in G_i involved in fewer than

$$\max(0, 2 \lceil lb \cdot \gamma \rceil - lb - 1)$$

triangles.

The correctness of the triangle-based reduction rule is ensured by the following property.

LEMMA 5.2 (TRIANGLE REQUIREMENT). *For any edge in a γ -CQC g , the number of triangles containing it is at least*

$$\max(0, 2 \lceil \gamma \cdot (|V(g)| - 1) \rceil - |V(g)|).$$

These reduction rules are applied iteratively until convergence, effectively shrinking the local subgraph of each sub-query. For brevity, the complete pseudocode of the final MCQC-TD algorithm, which integrates both the MCQC-Heu method and this decomposition strategy, is provided in Appendix D of the technical report [1].

Time and space complexity. The time complexity of MCQC-Heu is bounded by $O(n \cdot (\delta + \delta d_{\max}))$, where $d_{\max}$ is the maximum degree of the input graph. This bound arises because the method processes at most n ego-networks, each containing at most $\delta d_{\max}$ vertices. For the decomposition approach, generating and reducing n subgraphs incurs a cost of $O(n \cdot (\delta d_{\max})^2)$ [14]. Regarding space complexity, MCQC-TD operates in $O(|V| + |E|)$ space, as it only requires maintaining the original graph structure alongside at most one local subgraph at any given time. Synthesizing these components, the overall worst-case time complexity of our integrated framework is formally established below. Detailed proofs and analyses of both the time and space complexities are deferred to Appendix D of the technical report [1].

THEOREM 5.3. *With MCQC-Heu and the query decomposition approach, MCQC-TD runs in $O(n^2 \cdot \delta^{3.5} \cdot d_{\max}^{4.5} \cdot \lambda_t^{\delta d_{\max}})$.*

Remark on practical performance. While the worst-case time complexity of MCQC-TD remains exponential due to the NP-hard nature of the MCQC problem, the algorithm is highly efficient in practice. First, real-world graphs typically exhibit small degeneracy δ (as shown in Appendix E of the technical report [1]), which keeps the relevant parameters in our theoretical bound small and contributes to high scalability in practice. Second, even on dense graphs with large degeneracy (e.g., the 2nd DIMACS instances G1–G10), MCQC-TD remains robust. This is because the synergy between our top-down size-constrained search and the structural reduction rule (**RR**) aggressively prunes unpromising branches at very shallow depths. Consequently, the practical search space is substantially smaller than the theoretical worst-case bound, enabling MCQC-TD to consistently outperform baselines even in such challenging scenarios.

6 Experiments

In this section, we conduct extensive experiments to evaluate the practical performance of the proposed MCQC-TD query method.

Setup. All algorithms are implemented in C++ and compiled with -O3. Experiments are conducted on a server equipped with an Intel(R) Xeon(R) Platinum 8358P CPU @ 2.60GHz and 256GB of main memory, running Ubuntu 20.04.6. We enforce a strict **3-hour** timeout (10,800 seconds) per query and report **OOT** (Out of Time) for incomplete executions. To simulate diverse query workloads, we vary the density parameter $\gamma \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$. Due to space constraints, results for some γ values are deferred to Appendix E of the technical report [1]. We confirm that those deferred results show similar patterns to those in the main text. The code and data archive is publicly available at <https://github.com/liubufan1998/ConnectivityBasedQuasiClique>.

Algorithms. We compare MCQC-TD with the following baselines.

- **EIA⁴**: the state-of-the-art solution for the MCQC query [54].
- **MCQC-Enum**: Our proposed BB-based baseline (Algorithm 1).

Datasets. We evaluate the algorithms on three collections of graphs, which encompass a total of **664 synthetic** and **real-world** datasets.

⁴We re-implemented the algorithm proposed in [54]. Our implementation achieves better performance than that reported in their paper.

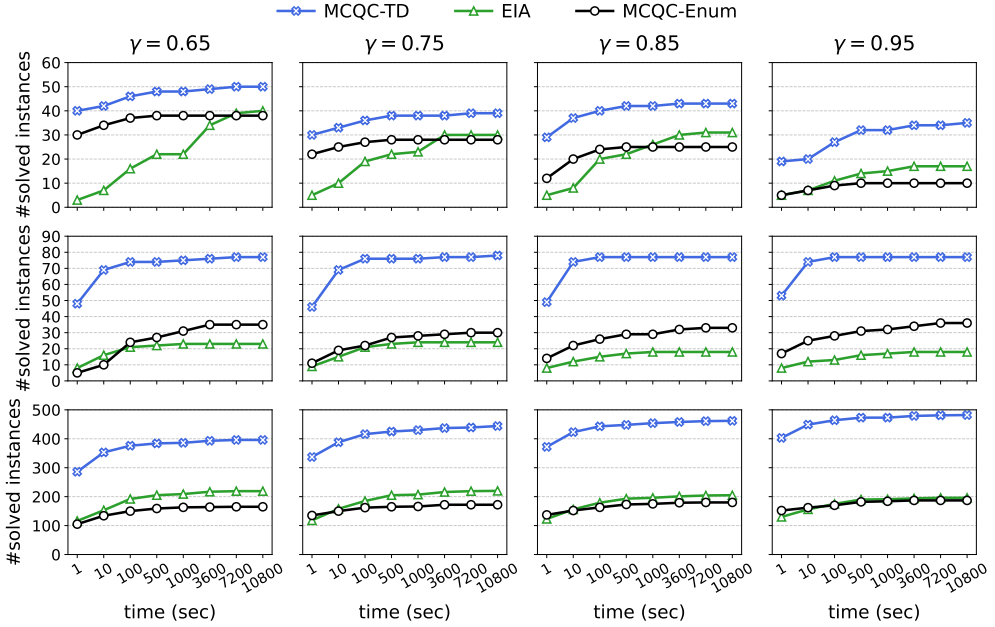


Fig. 5. Number of solved query instances. Top row: 2nd DIMACS graphs; Middle row: 10th DIMACS graphs; Bottom row: Real-world graphs. Columns from left to right correspond to $\gamma = 0.65, 0.75, 0.85, 0.95$.

- The **2nd DIMACS graphs** collection:⁵ 80 synthetic graphs from the 2nd DIMACS implementation challenge.
- The **10th DIMACS graphs** collection:⁶ 84 benchmark graphs from the 10th DIMACS implementation challenge.
- The **real-world graphs** collection:⁷ 500 real-world graphs from *Network Repository*, including biological networks (36), dynamic networks (85), interaction networks (29), labeled networks (104), road networks (15), scientific computing (11), social networks (75), facebook (114), and web networks (31).

To facilitate detailed analysis, particularly for query time evaluation, we select 40 representative graphs across these three collections. Specifically, we select 10 graphs (G1–G10) from the 2nd DIMACS set, 10 graphs (G11–G20) from the 10th DIMACS set, and 20 graphs (G21–G40) from the real-world collection, ensuring a comprehensive coverage of diverse domains. Due to space constraints, the detailed statistics of these 40 graphs (including vertex/edge counts, density, and degeneracy) are deferred to Appendix E of the technical report [1]. Overall, these representative graphs vary significantly in scale and structure: the number of vertices ranges from 64 to 59×10^6 , the number of edges ranges from 704 to 117×10^6 , and the graph degeneracy (δ) ranges from 2 to 894.

6.1 Efficiency Evaluation

Exp1: Comparison of the number of solved instances on all graphs. Figure 5 illustrates the number of successfully processed queries across all graphs under varying γ values. Overall, the results demonstrate that MCQC-TD solves the highest number of instances within the time limit. Specifically, on the 2nd DIMACS graphs (top row of Figure 5), MCQC-TD exhibits a substantial speedup over EIA. Within 1 second, MCQC-TD already matches or exceeds the total number of queries that EIA processes within the 3-hour (10,800 seconds) limit for several γ values. For example, when $\gamma = 0.65$, MCQC-TD finishes 40 instances within 1 second, matching the total number EIA achieves after 3 hours. Similarly, on the 10th DIMACS graphs (middle row), this dominance is even more obvious. With $\gamma = 0.65$, MCQC-TD processes 48 instances in 1 second, more than double the total instances (23) solved by EIA in 3 hours. The superiority of MCQC-TD extends to the collection of 500 real-world graphs (bottom row). That is, MCQC-TD solves significantly more instances within 1 second than EIA solves in 3 hours across all γ values. Specifically, at $\gamma = 0.65$, MCQC-TD solves 286 instances in 1 second, far surpassing the 219 instances solved by EIA after 3 hours. The performance of our BB-based baseline, MCQC-Enum, is comparable to that of EIA. For instance, on real-world graphs with $\gamma = 0.85$, MCQC-Enum solves 180 instances, slightly fewer than EIA's 205. This suggests that a naive BB-based method without structural optimizations is insufficient to outperform the MIP-based method. However, MCQC-TD consistently outperforms both baselines by a wide margin in all settings, verifying that the advanced pruning techniques and search strategies are the key drivers of performance improvement rather than the BB framework alone.

Exp2: Query time comparison on representative graphs. Table 1 presents the query time on 40 representative graphs with $\gamma = 0.75$. Overall, MCQC-TD achieves the lowest query time on 36 graphs and ties for the lowest time on two others. It surpasses EIA by up to five orders of magnitude (e.g., G25), as EIA frequently times out due to the high cost of mixed integer programming. Our BB-based baseline, MCQC-Enum, is also generally far slower, failing on instances like G2-G4 that MCQC-TD solves immediately. We note that MCQC-Enum is only slightly faster on a few small and dense graphs (e.g., G5, G6). In such cases, the maximum CQCs span almost the entire graph, meaning the computational overhead of MCQC-TD's advanced techniques outweighs their pruning benefits. Nevertheless, across the vast majority of datasets, the strong efficiency of MCQC-TD is attributed to the combined pruning power of **RR**, **HPS**, the top-down search, and query decomposition.

Exp3: Scalability evaluation on a large graph. To evaluate scalability, we conduct experiments on a massive real-world graph, *com-Friendster*⁸, which contains 66M vertices and 2B edges. We generate five subgraphs by randomly sampling between 20% and 100% of the vertices. Figure 6 reports the query time of all methods under $\gamma = 0.85$ and 0.95. The results show that MCQC-TD scales robustly, successfully processing all queries within the time limit. In contrast, both EIA and MCQC-Enum time out on every instance, and are unable to solve even the 20% subgraph. These findings demonstrate that MCQC-TD is uniquely capable of handling large-scale graphs that existing exact approaches cannot manage.

Exp4: Memory analysis on representative graphs. The complete peak-memory (Resident Set Size) results on 40 representative graphs with $\gamma = 0.75$ are deferred to Appendix E of the technical report [1]. Overall, MCQC-TD shows comparable memory usage to the baselines on most datasets, validating our $O(|V| + |E|)$ theoretical space complexity. MCQC-TD consumes more memory than EIA on a few massive graphs, approximately $1.9\times$ on G29 and $2.3\times$ on G32. This occurs because

⁵<http://archive.dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique/>

⁶<https://sites.cc.gatech.edu/dimacs10/downloads.shtml>

⁷<https://networkrepository.com/index.php>

⁸<https://snap.stanford.edu/data/com-Friendster.html>

Table 1. Query time in seconds on 40 graphs with $\gamma = 0.75$

ID	MCQC-TD	EIA	Enum	ID	MCQC-TD	EIA	Enum
G1	0.01	0.69	7665.08	G21	49.64	OOT	OOT
G2	0.01	4.72	OOT	G22	2.16	OOT	OOT
G3	0.01	3.74	OOT	G23	0.18	70.68	28.02
G4	0.01	45.84	OOT	G24	0.48	65.36	0.55
G5	1.50	3277.11	0.45	G25	0.17	OOT	0.25
G6	8.53	OOT	4.37	G26	0.09	0.64	0.12
G7	0.01	0.22	0.01	G27	1.18	26.80	1.19
G8	0.01	1.20	OOT	G28	0.17	1.86	0.19
G9	0.01	0.38	0.01	G29	9.05	OOT	OOT
G10	24.91	OOT	OOT	G30	2.59	OOT	OOT
G11	23.40	OOT	OOT	G31	5.27	OOT	OOT
G12	1.95	OOT	OOT	G32	204.04	OOT	OOT
G13	22.28	OOT	OOT	G33	2369.81	OOT	OOT
G14	2.25	OOT	OOT	G34	6.59	OOT	OOT
G15	7.10	OOT	OOT	G35	58.20	OOT	OOT
G16	5.26	OOT	OOT	G36	58.99	OOT	OOT
G17	0.65	28.97	0.67	G37	8.72	OOT	OOT
G18	0.17	1.88	0.20	G38	9.09	OOT	OOT
G19	3.29	OOT	OOT	G39	88.00	OOT	OOT
G20	0.35	1.78	19.32	G40	70.92	OOT	OOT

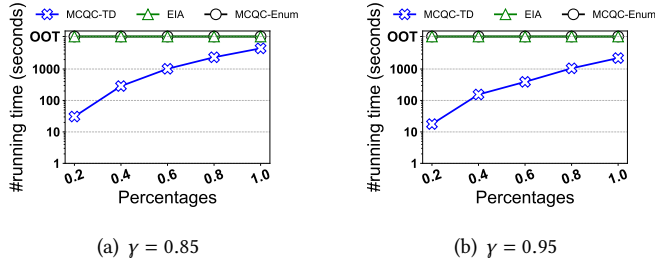


Fig. 6. Scalability test on the graph com-Friendster

EIA times out early on these hard instances, so its reported memory reflects only the initial MIP formulation stage rather than the full search process.

Exp5: Number of top-down iterations. The complete iteration counts on 40 representative graphs across all γ values, including $\gamma = 0.75$, are deferred to Appendix E of the technical report [1]. Overall, the search converges in few iterations (often ≤ 5) for the majority of instances. The primary reason for this high efficiency is that our optimized initialization strategy (MCQC-Heu) provides a tight lower bound. This tightly bounded initial solution narrows the gap to the theoretical size upper bound from the very beginning, allowing the top-down search to terminate quickly.

6.2 Ablation Studies

We conduct extensive ablation studies to evaluate the effectiveness of the key techniques integrated into MCQC-TD. Specifically, we compare MCQC-TD against four variants: (1) noHeu, which disables the optimized initialization method MCQC-Heu (Algorithm 4); (2) noHPS, which replaces the hybrid

Table 2. Ablation of Initialization Strategy (noHeu)

ID	Ours	noHeu	ID	Ours	noHeu	ID	Ours	noHeu	ID	Ours	noHeu
G1	0.01	0.01	G11	23.40	21.05	G21	49.64	50.84	G31	5.27	18.84
G2	0.01	0.01	G12	1.95	1.41	G22	2.16	3.05	G32	204.04	240.86
G3	0.01	0.01	G13	22.28	10.95	G23	0.18	0.27	G33	2369.80	2400.80
G4	0.01	0.01	G14	2.25	1.57	G24	0.48	0.74	G34	6.59	5.70
G5	1.50	1.66	G15	7.10	5.15	G25	0.17	0.31	G35	58.20	58.59
G6	8.53	15.97	G16	5.26	3.24	G26	0.09	0.17	G36	58.99	58.57
G7	0.01	0.01	G17	0.65	1.03	G27	1.18	1.73	G37	8.72	17.92
G8	0.01	0.01	G18	0.17	0.28	G28	0.17	0.28	G38	9.09	16.68
G9	0.01	0.01	G19	3.29	2.66	G29	9.05	29.14	G39	88.00	191.45
G10	24.91	24.62	G20	0.35	0.49	G30	2.59	2.95	G40	70.92	98.60

Table 3. Ablation of Query Decomposition (noDC)

ID	Ours	noDC	ID	Ours	noDC	ID	Ours	noDC	ID	Ours	noDC
G1	0.01	2.35	G11	23.40	OOT	G21	49.64	OOT	G31	5.27	OOT
G2	0.01	OOT	G12	1.95	1.83	G22	2.16	2.13	G32	204.04	737.91
G3	0.01	155.0	G13	22.28	22.06	G23	0.18	0.19	G33	2369.8	OOT
G4	0.01	OOT	G14	2.25	OOT	G24	0.48	0.46	G34	6.59	128.40
G5	1.50	0.51	G15	7.10	OOT	G25	0.17	0.17	G35	58.20	OOT
G6	8.53	4.57	G16	5.26	OOT	G26	0.09	0.10	G36	58.99	OOT
G7	0.01	0.01	G17	0.65	0.64	G27	1.18	1.18	G37	8.72	19.32
G8	0.01	0.11	G18	0.17	0.18	G28	0.17	0.18	G38	9.09	20.32
G9	0.01	0.01	G19	3.29	2.72	G29	9.05	8.89	G39	88.00	OOT
G10	24.91	OOT	G20	0.35	2.88	G30	2.59	OOT	G40	70.92	OOT

Table 4. Ablation of Hybrid Pivot Selection (noHPS)

ID	Ours	noHPS	ID	Ours	noHPS	ID	Ours	noHPS	ID	Ours	noHPS
G1	0.01	0.01	G11	23.40	23.29	G21	49.64	OOT	G31	5.27	5.24
G2	0.01	0.25	G12	1.95	1.93	G22	2.16	2.32	G32	204.04	206.97
G3	0.01	0.01	G13	22.28	22.27	G23	0.18	0.18	G33	2369.8	OOT
G4	0.01	0.53	G14	2.25	2.41	G24	0.48	0.48	G34	6.59	2222.0
G5	1.50	1.78	G15	7.10	7.43	G25	0.17	0.18	G35	58.20	OOT
G6	8.53	8.27	G16	5.26	4.69	G26	0.09	0.12	G36	58.99	OOT
G7	0.01	0.01	G17	0.65	0.69	G27	1.18	1.31	G37	8.72	8.96
G8	0.01	0.24	G18	0.17	0.19	G28	0.17	0.19	G38	9.09	9.10
G9	0.01	0.01	G19	3.29	2.62	G29	9.05	10.00	G39	88.00	73.12
G10	24.91	OOT	G20	0.35	0.36	G30	2.59	2.80	G40	70.92	70.51

pivot selection (HPS) with the cut-deprioritized pivot selection (CDPS) described in Section 4.3; (3) noDC, which removes the query decomposition strategy introduced in Section 5.2; and (4) noTD, which substitutes the top-down iterative search with a bottom-up strategy. To facilitate a clear assessment, Tables 2 to 5 present the isolated ablation results on 40 representative graphs ($\gamma = 0.75$). Overall, these results reveal a clear hierarchy of importance: the decomposition strategy (noDC) is the absolute cornerstone for scalability, closely followed by the top-down search (noTD) and hybrid pivot (noHPS), which prevent combinatorial explosions. Meanwhile, the optimized initialization (noHeu) serves as a crucial performance accelerator.

Exp6: Ablation of the initialization optimizer. As shown in Table 2, MCQC-TD outperforms noHeu on most datasets. For instance, on the graph G31, MCQC-TD completes the query in 5.27 seconds, while noHeu requires 18.84 seconds. This advantage stems from MCQC-Heu identifying larger initial γ -CQCs, enabling more aggressive graph preprocessing and allowing the iterative process to terminate earlier. Furthermore, a detailed comparison in Appendix E of the technical report [1] confirms that MCQC-Heu finds larger initial subgraphs than the basic initialization in the majority of instances, with size differences reaching 951.

Table 5. Ablation of Top-down Search Strategy (noTD)

ID	Ours	noTD	ID	Ours	noTD	ID	Ours	noTD	ID	Ours	noTD
G1	0.01	0.01	G11	23.40	34.35	G21	49.64	115.33	G31	5.27	38.16
G2	0.01	0.01	G12	1.95	2.01	G22	2.16	3.35	G32	204.04	OOT
G3	0.01	0.01	G13	22.28	29.17	G23	0.18	0.18	G33	2369.8	OOT
G4	0.01	0.02	G14	2.25	2.29	G24	0.48	0.48	G34	6.59	37.02
G5	1.50	OOT	G15	7.10	7.42	G25	0.17	0.16	G35	58.20	82.46
G6	8.53	18.79	G16	5.26	4.74	G26	0.09	0.09	G36	58.99	82.83
G7	0.01	0.01	G17	0.65	0.65	G27	1.18	1.21	G37	8.72	8.88
G8	0.01	0.01	G18	0.17	0.16	G28	0.17	0.16	G38	9.09	11.35
G9	0.01	OOT	G19	3.29	2.58	G29	9.05	100.44	G39	88.00	OOT
G10	24.91	29.43	G20	0.35	0.35	G30	2.59	2.67	G40	70.92	1625.0

Exp7: Ablation of the query decomposition strategy. Table 3 demonstrates the critical role of the decomposition strategy in handling large graphs. Without it, noDC fails to solve 15 instances within the time limit (e.g., G14, G21, G30). This occurs because the decomposition strategy breaks the original MCQC query into multiple smaller, independent sub-queries. Solving these smaller instances is significantly more efficient for our branch-and-bound method than tackling the massive original graph directly.

Exp8: Ablation of the hybrid pivot selection method. Comparing MCQC-TD with noHPS (Table 4), the latter suffers from reduced efficiency and times out on 5 complex instances (e.g., G33, G35, G36). Although both **HPS** and CDPS guarantee the theoretical performance of **RR**, **HPS** is much more practical. It frequently selects pivots based on simple minimum degree checks (DDPS), avoiding the costly maximum flow computations required by CDPS at every step, thereby substantially reducing pivot selection overhead.

Exp9: Ablation of the top-down iterative search strategy. As shown in Table 5, MCQC-TD exhibits a clear advantage over noTD, which times out on 5 instances (e.g., G5, G9, G32). The superiority of the top-down strategy lies in its ability to terminate immediately once the first valid γ -CQC of the target size is found. In contrast, the bottom-up strategy must exhaustively search up to the size upper bound. In early stages (e.g., searching for size $lb + 1$), the target size is small, leading to ineffective graph reduction and forcing the algorithm to explore an unnecessarily large search space.

6.3 Case Study

Exp10: Evaluation of query predicate effectiveness. This study assesses whether the γ -CQC predicate provides a superior structural definition for querying cohesive substructures compared to degree-based (γ -DQC), edge-based (γ -EQC), and absolute vertex-connectivity-based (s -bundle) models. Since we focus on *exact algorithms for processing the maximum query*, we evaluate the quality of the query result (the largest subgraph) returned by each model against ground-truth semantics. We use the Amazon co-purchase graph⁹ ($n=335K$, $m=926K$), containing 151K ground-truth communities defined by product categories, as a benchmark. We compute global maximum subgraphs using state-of-the-art exact solvers: our MCQC-TD (γ -CQC), IterQC [59] (γ -DQC), QClique [47] (γ -EQC), and SymBD [39] (s -bundle). The results are matched against ground-truth categories to benchmark the modeling power of each predicate.

Table 6 reports the evaluation metrics. To facilitate a direct comparison, we align the proportional parameter γ and the absolute parameter s from loose to strict (e.g., $\gamma = 0.55$ corresponds to the loosest setting $s = 14$). Overall, the γ -CQC predicate consistently achieves the most robust performance, attaining the highest average F1-score (0.74) and significantly outperforming γ -DQC (0.63) and

⁹<https://snap.stanford.edu/data/com-Amazon.html>

Table 6. Semantic quality assessment of query results returned by different query predicates

Setting	Metric	Model				Setting	Metric	Model			
		γ -CQC	γ -DQC	γ -EQC	s -bundle			γ -CQC	γ -DQC	γ -EQC	s -bundle
$\gamma = 0.55$ ($s = 14$)	Precision	0.71	1.00	0.27	0.95	$\gamma = 0.85$ ($s = 5$)	Precision	1.00	1.00	1.00	1.00
	Recall	1.00	0.33	0.08	0.42		Recall	0.89	0.89	0.31	0.17
	F1-score	0.83	0.50	0.12	0.58		F1-score	0.94	0.94	0.48	0.29
	Density	0.73	0.75	0.56	0.40		Density	0.93	0.93	0.87	0.76
$\gamma = 0.65$ ($s = 11$)	Precision	1.00	1.00	0.36	0.94	$\gamma = 0.95$ ($s = 2$)	Precision	0.71	0.57	0.63	0.25
	Recall	0.30	0.30	0.33	0.37		Recall	1.00	0.80	0.71	0.67
	F1-score	0.47	0.47	0.35	0.53		F1-score	0.83	0.67	0.67	0.36
	Density	0.78	0.78	0.65	0.44		Density	1.00	1.00	0.96	0.93
$\gamma = 0.75$ ($s = 8$)	Precision	0.44	0.44	0.90	0.79	Average	Precision	0.77	0.80	0.63	0.79
	Recall	1.00	0.80	0.39	0.85		Recall	0.84	0.62	0.36	0.50
	F1-score	0.62	0.57	0.55	0.82		F1-score	0.74	0.63	0.43	0.52
	Density	0.86	0.81	0.76	0.59		Density	0.86	0.85	0.76	0.62

γ -EQC (0.43). In the Amazon graph, degree-based models enforce rigid per-vertex constraints, excluding structurally supported peripheral products and leading to fragmented results. In contrast, the global robustness of γ -CQC maintains connectivity through multiple vertex-disjoint paths, retaining complete community structures and yielding the highest average Recall (0.84).

Furthermore, the results highlight a crucial semantic advantage of γ -CQC over the s -bundle model. While s -bundle occasionally achieves high F1-scores under specific intermediate settings (e.g., $s = 8$), its performance degrades significantly when the constraint is too loose or too strict. This behavior stems from its use of a fixed integer threshold s , which lacks adaptability: a fixed s is inevitably too strict for massive communities and too loose for small ones. In contrast, the proportional parameter γ allows γ -CQC to naturally adapt its connectivity requirements to the community's size, guaranteeing stable, high-quality results across all parameter settings and a higher average F1-score (0.74 vs. 0.52).

7 Related Work

Quasi-clique query models. Research on processing γ -quasi-clique queries can be categorized by the structural constraint enforced. The first category concerns *degree-based* constraints (γ -DQC) [28, 32, 35, 37, 42, 44, 48, 59, 64, 66], requiring each vertex to connect to at least $\gamma \cdot (n - 1)$ neighbors. The second category involves *edge-based* constraints (γ -EQC), requiring the subgraph to contain a density of at least γ [2, 3, 9, 41, 46, 51–53, 55]. While algorithms for finding maximum instances of these models are well-studied, they overlook structural robustness. Consequently, existing query processing techniques designed for γ -DQC or γ -EQC cannot be directly transferred to the connectivity-based γ -CQC.

The third line of research addresses the connectivity-based model (γ -CQC), originally highlighted by Pattillo et al. [45] for its superior cohesiveness. Veremyev et al. [54] formalized the maximum γ -CQC problem and proposed the *Exact Iterative Algorithm* (EIA). EIA maps the graph query to a mixed integer programming (MIP) problem. It exploits the observation that a graph g is a γ -CQC if there exists an integer k such that $\kappa(g) \geq k$ and $\gamma \cdot (|V(g)| - 1) \leq k$. Thus, EIA encodes the query into a sequence of MIP constraints to find the largest subgraph satisfying these conditions. However, EIA relies on generic MIP solvers which act as black-boxes, ignoring the topological locality essential for efficient graph query processing. This results in poor scalability on large graphs (e.g., the largest graph tested in [54] has only 2K vertices). Furthermore, EIA is bound by

a trivial worst-case complexity of $O^*(2^n)$, whereas our MCQC-TD achieves a non-trivial bound of $O^*(\lambda_t^n)$ with $\lambda_t < 2$.

Other structural graph queries. Querying cohesive subgraphs with various structural constraints is a core topic in graph data management. Representative models include k -plex [14, 15, 24, 26, 33, 34, 57], k -defective clique [11, 18, 19, 25], k -clique [12, 56, 65], densest subgraph [40, 60], and s -bundle [27, 30, 39, 45, 67].

Particularly related to our study is the s -bundle model, which also enforces vertex connectivity. Recent works have proposed efficient algorithms for processing s -bundles, such as finding the maximum s -bundle [39] or enumerating all maximal s -bundles [30]. These state-of-the-art algorithms typically exploit local vertex connectivity to design powerful reduction rules. However, the s -bundle model is governed by an absolute integer threshold s and rigorously satisfies the hereditary property (i.e., any induced subgraph of an s -bundle remains an s -bundle). This structural property allows existing methods to rely heavily on monotonicity for pruning. In sharp contrast, the γ -CQC model is governed by a proportional constraint γ that adapts to the subgraph size, making it fundamentally non-hereditary. This non-hereditary nature completely invalidates traditional monotonicity-based pruning techniques. To overcome this unique challenge, our reduction rule (RR) is derived from a novel equivalent definition of γ -CQC (Theorem 4.3), explicitly distinguishing our algorithmic design from existing s -bundle search methods. For a broader overview of other structural graph queries, interested readers are referred to recent surveys [13, 22, 31, 36].

8 Conclusion

We studied the efficient exact processing of the *Maximum Connectivity-based γ -Quasi-Clique* (MCQC) query and introduced MCQC-TD, a specialized branch-and-bound method. Its top-down search fixes a target solution size, allowing the reduction rule RR and hybrid pivot selection HPS to prune infeasible branches early. This design improves the worst-case time complexity from $O^*(2^n)$ to $O^*(\lambda_t^n)$, where $\lambda_t < 2$. The optimized initialization method MCQC-Heu and query decomposition further tighten the bounds and divide massive queries into smaller subproblems.

Experiments on 664 benchmark and real-world graphs show that MCQC-TD solves substantially more instances than the existing exact method and achieves speedups of up to five orders of magnitude. The ablation studies demonstrate the complementary benefits of its main components, while the case study confirms the value of the connectivity-based predicate for identifying ground-truth communities. Extending the current single-machine method to distributed graph environments while preserving exactness is an important direction for future work.

Acknowledgments

The research of Shengxin Liu was partially supported by the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2025A1515060015) and by the Shenzhen Science and Technology Program (Grant No. GXWD20231129111306002). Hejiao Huang was supported by the Shenzhen Science and Technology Program under Grant No. JCYJ20250604145359059 and Grant No. ZDSY20120613125016389. Kaiqiang Yu was supported by the “111 Center” (No. B26023) and the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118). Zhaoquan Gu was supported in part by the Shenzhen Science and Technology Program under Grant KJZD20240903103811016, in part by the Major Key Project of PCL under Grant PCL2024A05, and in part by the Science and Technology Development Fund, Macao SAR under Grant 0007/2024/AKP.

References

- [1] Technical Report and Source Code. <https://github.com/liubufan1998/ConnectivityBasedQuasiClique>.
- [2] James Abello, Panos M Pardalos, and Mauricio GC Resende. 1999. On maximum clique problems in very large graphs. *External Memory Algorithms* 50 (1999), 119–130.
- [3] James Abello, Mauricio GC Resende, and Sandra Sudarsky. 2002. Massive quasi-clique detection. In *Proceedings of the Latin American Symposium on Theoretical Informatics (LATIN)*. 598–612.
- [4] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. 2016. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems* 55 (2016), 278–288.
- [5] Vladimir Batagelj and Matjaž Zaveršnik. 2003. An $O(m)$ algorithm for cores decomposition of networks. *CoRR* cs.DS/0310049 (2003).
- [6] Punam Bedi and Chhavi Sharma. 2016. Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 6, 3 (2016), 115–135.
- [7] Matthias Bentert, Anne-Sophie Himmel, Hendrik Molter, Marco Morik, Rolf Niedermeier, and René Saitenmacher. 2019. Listing all maximal k -plexes in temporal graphs. *ACM J. Exp. Algorithmics* 24, 1 (2019), 1.13:1–1.13:27.
- [8] Malay Bhattacharyya and Sanghamitra Bandyopadhyay. 2009. Mining the largest quasi-clique in human protein interactome. In *Proceedings of the International Conference on Adaptive and Intelligent Systems*. 194–199.
- [9] Zvika Brakerski and Boaz Patt-Shamir. 2011. Distributed discovery of large near-cliques. *Distributed Computing* 24, 2 (2011), 79–89.
- [10] Lijun Chang. 2019. Efficient maximum clique computation over large sparse graphs. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (SIGKDD)*. 529–538.
- [11] Lijun Chang. 2023. Efficient maximum k -defective clique computation with improved time complexity. *Proceedings of the ACM on Management of Data (SIGMOD)* 1, 3 (2023), 1–26.
- [12] Lijun Chang, Rashmika Gamage, and Jeffrey Xu Yu. 2024. Efficient k -clique count estimation with accuracy guarantee. *Proceedings of the VLDB Endowment* 17, 11 (2024), 3707–3719.
- [13] Lijun Chang and Lu Qin. 2018. *Cohesive Subgraph Computation over Large Sparse Graphs*. Springer.
- [14] Lijun Chang, Mouyi Xu, and Darren Strash. 2022. Efficient maximum k -plex computation over large sparse graphs. *Proceedings of the VLDB Endowment* 16, 2 (2022), 127–139.
- [15] Lijun Chang and Kai Yao. 2024. Maximum k -plex computation: Theory and practice. *Proceedings of the ACM on Management of Data (SIGMOD)* 2, 1 (2024), 1–26.
- [16] Lijun Chang, Jeffrey Xu Yu, Lu Qin, Xuemin Lin, Chengfei Liu, and Weifa Liang. 2013. Efficiently computing k -edge connected components via graph decomposition. In *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*. 205–216.
- [17] Jiejiang Chen, Shaowei Cai, Shiwei Pan, Yiyuan Wang, Qingwei Lin, Mengyu Zhao, and Minghao Yin. 2021. NuQClq: An effective local search algorithm for maximum quasi-clique problem. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 12258–12266.
- [18] Xiaoyu Chen, Yi Zhou, Jin-Kao Hao, and Mingyu Xiao. 2021. Computing maximum k -defective cliques in massive graphs. *Computers & Operations Research* 127 (2021), 105131.
- [19] Qiangqiang Dai, Rong-Hua Li, Meihao Liao, and Guoren Wang. 2023. Maximal defective clique enumeration. *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)* 1, 1 (2023), 1–26.
- [20] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, et al. 2022. Graph pattern matching in GQL and SQL/PGQ. In *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*. 2246–2258.
- [21] Wenfei Fan, Tao He, Longbin Lai, Xue Li, Yong Li, Zhao Li, Zhengping Qian, Chao Tian, Lei Wang, Jingbo Xu, et al. 2021. GraphScope: a unified engine for big graph processing. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2879–2892.
- [22] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29 (2020), 353–392.
- [23] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhuan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.
- [24] Jian Gao, Jiejiang Chen, Minghao Yin, Rong Chen, and Yiyuan Wang. 2018. An exact algorithm for maximum k -plexes in massive graphs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1449–1455.
- [25] Jian Gao, Zhenghang Xu, Ruizhi Li, and Minghao Yin. 2022. An exact algorithm with new upper bounds for the maximum k -defective clique problem in massive sparse graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 10174–10183.
- [26] Shuohao Gao, Kaiqiang Yu, Shengxin Liu, and Cheng Long. 2024. Maximum k -plex search: An alternated reduction-and-bound method. *Proceedings of the VLDB Endowment*. 18, 2 (2024), 363–376.

- [27] Timo Gschwind, Stefan Irnich, and Isabel Podlinski. 2018. Maximum weight relaxed cliques and Russian Doll Search revisited. *Discrete Applied Mathematics* 234 (2018), 131–138.
- [28] Guimu Guo, Da Yan, M. Tamer Özsu, Zhe Jiang, and Jalal Khalil. 2020. Scalable mining of maximal quasi-cliques: An algorithm-system codesign approach. *Proc. VLDB Endow.* 14, 4 (2020), 573–585.
- [29] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3549–3568.
- [30] Shan Hu, Yi Zhou, Mingyu Xiao, Zhang-Hua Fu, and Zhipeng Lü. 2023. Listing maximal k -relaxed-vertex connected components from large graphs. *Information Sciences* 620 (2023), 67–83.
- [31] Xin Huang, Laks V. S. Lakshmanan, and Jianliang Xu. 2019. *Community Search over Big Graphs*. Morgan & Claypool Publishers.
- [32] Daxin Jiang and Jian Pei. 2009. Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data* 2, 4 (2009), 1–42.
- [33] Hua Jiang, Fusheng Xu, Zhifei Zheng, Bowen Wang, and Wei Zhou. 2023. A refined upper bound and inprocessing for the maximum k -plex problem. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 5613–5621.
- [34] Hua Jiang, Dongming Zhu, Zhichao Xie, Shaowen Yao, and Zhang-Hua Fu. 2021. A new upper bound based on vertex partitioning for the maximum k -plex problem. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1689–1696.
- [35] Jalal Khalil, Da Yan, Guimu Guo, and Lyuheng Yuan. 2022. Parallel mining of large maximal quasi-cliques. *The VLDB Journal* 31, 4 (2022), 649–674.
- [36] Victor E. Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. 2010. A Survey of Algorithms for Dense Subgraph Discovery. *Managing and Mining Graph Data* (2010), 303–336.
- [37] Guimei Liu and Limsoon Wong. 2008. Effective pruning techniques for mining quasi-cliques. In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*. 33–49.
- [38] Xiaowen Liu, Jinyan Li, and Lusheng Wang. 2008. Modeling protein interacting groups by quasi-bicliques: Complexity, algorithm, and application. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 7, 2 (2008), 354–364.
- [39] Yang Liu, Hejiao Huang, Kaiqiang Yu, Shengxin Liu, and Cheng Long. 2025. Efficient maximum s -bundle search via local vertex connectivity. *Proceedings of the ACM on Management of Data (SIGMOD)* 3, 1 (2025), 1–27.
- [40] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2021. On directed densest subgraph discovery. *ACM Transactions on Database Systems (TODS)* 46, 4 (2021), 1–45.
- [41] Foad Mahdavi Pajouh, Zhuqi Miao, and Balabhaskar Balasundaram. 2014. A branch-and-bound approach for maximum quasi-cliques. *Annals of Operations Research* 216, 1 (2014), 145–161.
- [42] Hideo Matsuda, Tatsuya Ishihara, and Akihiro Hashimoto. 1999. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science* 210, 2 (1999), 305–325.
- [43] Antonio Messina, Antonino Fiannaca, Laura La Paglia, Massimo La Rosa, and Alfonso Urso. 2018. BioGraph: A web application and a graph database for querying and analyzing bioinformatics resources. *BMC Systems Biology* 12, Suppl 5 (2018), 98.
- [44] Grigory Pastukhov, Alexander Veremyev, Vladimir Boginski, and Oleg A Prokopyev. 2018. On maximum degree-based-quasi-clique problem: Complexity and exact approaches. *Networks* 71, 2 (2018), 136–152.
- [45] Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko. 2013. On clique relaxation models in network analysis. *European Journal of Operational Research* 226, 1 (2013), 9–18.
- [46] Ahsanur Rahman, Kalyan Roy, Ramiza Maliha, and Townim Faisal Chowdhury. 2024. A fast exact algorithm to enumerate maximal pseudo-cliques in large sparse graphs. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (SIGKDD)*. 2479–2490.
- [47] Celso C Ribeiro and José A Riveaux. 2019. An exact algorithm for the maximum quasi-clique problem. *International Transactions in Operational Research* 26, 6 (2019), 2199–2229.
- [48] Seyed-Vahid Sanei-Mehri, Apurba Das, Hooman Hashemi, and Srikanta Tirthapura. 2021. Mining largest maximal quasi-cliques. *ACM Transactions on Knowledge Discovery from Data* 15, 5 (2021), 1–21.
- [49] Stephen B. Seidman. 1983. Network structure and minimum degree. *Social Networks* 5, 3 (1983), 269–287.
- [50] Stephen B. Seidman and Brian L Foster. 1978. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology* 6, 1 (1978), 139–154.
- [51] Shu Tadaka and Kengo Kinoshita. 2016. NCMine: Core-peripheral based functional module detection using near-clique mining. *Bioinformatics* 32, 22 (2016), 3454–3460.
- [52] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 2013. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (SIGKDD)*. 104–112.

- [53] Takeaki Uno. 2010. An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica* 56, 1 (2010), 3–16.
- [54] Alexander Veremyev, Oleg A. Prokopyev, Vladimir Boginski, and Eduardo L. Pasiliao. 2014. Finding maximum subgraphs with relatively large vertex connectivity. *European Journal of Operational Research* 239, 2 (2014), 349–362.
- [55] Alexander Veremyev, Oleg A Prokopyev, Sergiy Butenko, and Eduardo L Pasiliao. 2016. Exact MIP-based approaches for finding maximum quasi-cliques and dense subgraphs. *Computational Optimization and Applications* 64, 1 (2016), 177–214.
- [56] Kaixin Wang, Kaiqiang Yu, and Cheng Long. 2024. Efficient k -clique listing: An edge-oriented branching strategy. *Proceedings of the ACM on Management of Data (SIGMOD)* 2, 1 (2024), 1–26.
- [57] Zhengren Wang, Yi Zhou, Chunyu Luo, and Mingyu Xiao. 2023. A fast maximum k -plex algorithm parameterized by the degeneracy gap. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 5648–5656.
- [58] Dong Wen, Lu Qin, Ying Zhang, Lijun Chang, and Ling Chen. 2019. Enumerating k -vertex connected components in large graphs. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 52–63.
- [59] Hongbo Xia, Kaiqiang Yu, Shengxin Liu, Cheng Long, and Xun Zhou. 2025. Maximum degree-based quasi-clique search via an iterative framework. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (SIGKDD)*. 3285–3296.
- [60] Yichen Xu, Chenhao Ma, Yixiang Fang, and Zhifeng Bao. 2024. Efficient and effective algorithms for densest subgraph discovery and maintenance. *The VLDB Journal* 33, 5 (2024), 1427–1452.
- [61] Hai-Cheng Yi, Zhu-Hong You, De-Shuang Huang, and Chee Keong Kwoh. 2022. Graph representation learning in bioinformatics: Trends, methods and applications. *Briefings in Bioinformatics* 23, 1 (2022), bbab340.
- [62] Haiyuan Yu, Alberto Paccanaro, Valery Trifonov, and Mark Gerstein. 2006. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics* 22, 7 (2006), 823–829.
- [63] Kaiqiang Yu and Cheng Long. 2021. Graph mining meets fake news detection. In *Data Science for Fake News: Surveys and Perspectives*. Springer, 169–189.
- [64] Kaiqiang Yu and Cheng Long. 2023. Fast maximal quasi-clique enumeration: A pruning and branching co-design approach. *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)* 1, 3 (2023), 1–26.
- [65] Long Yuan, Lu Qin, Xuemin Lin, Lijun Chang, and Wenjie Zhang. 2016. Diversified top- k clique search. *The VLDB Journal* 25, 2 (2016), 171–196.
- [66] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. 2006. Coherent closed quasi-clique discovery from large dense graph databases. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (SIGKDD)*. 797–802.
- [67] Yi Zhou, Weibo Lin, Jin-Kao Hao, Mingyu Xiao, and Yan Jin. 2022. An effective branch-and-bound algorithm for the maximum s -bundle problem. *European Journal of Operational Research* 297, 1 (2022), 27–39.

Received 17 January 2026; revised 19 May 2026; accepted 12 June 2026