Efficient Locally h-Clique Densest Subgraph Discovery via Divide-and-Conquer

Yingli Zhou
yinglizhou@link.cuhk.edu.cn
The Chinese University of Hong Kong, Shenzhen

Taohua Huang
work.thhuang@gmail.com
Xidian University, Xian

Yixiang Fang*
fangyixiang@cuhk.edu.cn
The Chinese University of Hong Kong, Shenzhen

ABSTRACT

Finding the densest subgraph (DS) from a graph is a fundamental problem in graph databases. It has been extensively studied in the literature and has found many real applications in a wide range of fields, such as biology, finance, and social networks. This paper studies how to efficiently discover the locally h-clique densest subgraph (LhCDS), which is a recently-proposed variant of DS. An LhCDS is a subgraph which is the densest among the "local neighbors". Given a graph G, a number of LhCDSes can be returned, which reflect different dense regions of G and thus give more information than DS. Existing LhCDS solutions suffer from low efficiency due to extensive redundant computation. To improve efficiency, in this paper, we propose a divide-and-conquer-based algorithm, which not only reduces the search space but also has an improved time complexity. Extensive experiments on 15 large real-world graph datasets show that our proposed algorithm is up to two orders of magnitude faster than the state-of-the-art.

plainempty **PVLDB Reference Format:**
Yingli Zhou, Taohua Huang, and Yixiang Fang. Efficient Locally h-Clique Densest Subgraph Discovery via Divide-and-Conquer. PVLDB, 19(10): 2577 - 2590, 2026.
doi:10.14778/3828612.3828616

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/s01bvral/DCLDS.

1 INTRODUCTION

Dense subgraph discovery is a fundamental research topic in graph mining that has been extensively studied for decades [3, 6, 13, 27, 53, 55, 60, 68, 70, 73]. It has found various applications in a wide range of fields, including biology [19, 29, 39, 62], finance [16, 23], and social network analysis [4, 17, 32, 33, 44, 81, 82]. In many of these applications, finding a "near-clique" is very important since a "near-clique" can be considered a clique in the forming stage or one with missing edges due to data corruption. For example, in social networks (e.g., Facebook), large near-cliques are useful for detecting fake news [77].

In the literature, the *h-clique densest subgraph* (CDS) problem that finds the subgraph formed by overlapping h-cliques has proven very effective in detecting the "near-cliques" [25, 45, 51, 56, 72, 83, 84]. By finding the subgraph with the highest h-clique-density, which is the ratio of the number of h-cliques over the number of vertices in it, CDS uncovers the highly-connected component that exhibits strong internal interactions [25, 72, 84]. Whereas, in the context of the real world, the discovery of a single CDS offers limited insights. Listing the top-k CDSes is desired, but due to the substantial overlap inherent in h-cliques [64, 74, 76], the top-k CDSes may refer to the same dense region, still providing limited structural insights.

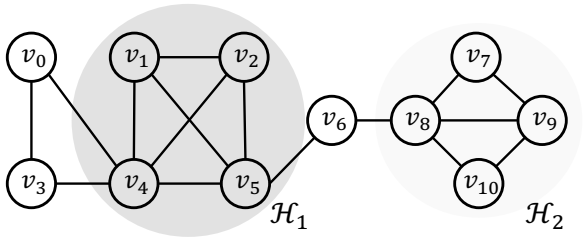


Figure 1: An example of locally h-clique densest subgraphs.

In light of this, detecting the top-k non-overlapping locally h-clique densest subgraphs (LhCDS) has attracted considerable interest in recent years [25, 45, 64, 74]. For example, in Figure 1, the 3-clique density of the subgraph $\mathcal{H}_1$ induced by $\{v_1, v_2, v_4, v_5\}$ (shaded region) is 1, since it contains four 3-cliques over four vertices. Moreover, $\mathcal{H}_1$ and $\mathcal{H}_2$ are identified as the top-2 L3CDSes: they are non-overlapping, and no other subgraph has a 3-clique density greater than theirs. We note that another closely related problem is the locally densest subgraph (LDS) discovery problem [53, 60], which can be viewed as a special case of the LhCDS problem when $h = 2$. That is, the LhCDS problem is a natural generalization of LDS. In this paper, we focus on the more general LhCDS problem and aim to develop efficient algorithms for discovering the top-k LhCDSes.

The LhCDS problem has found various real-world applications [25, 45, 51, 64, 74]. Since the LhCDS problem can be naturally viewed as identifying the top-k CDSes, it inherits similar practical applicability to the CDS problem. For example, as shown in [74], the LhCDS can be used to discover diverse communities. In addition, finding LhCDS is very useful for detecting the diverse "near-cliques" [25, 64], which can help discover biologically relevant functional groups [19, 43, 72, 73], find social communities [4, 14, 73], and detect anomalies [31, 69, 77]. Besides, finding LhCDS can help identify research communities in the DBLP network [64], and detect subnetworks with a specific function in the biology network [74].

Prior works. While the LhCDS is very useful in practice, it is computationally costly. In the literature, the maximum flow [64] and

*Corresponding author.

Proceedings of the VLDB Endowment, Vol. 19, No. 10 ISSN 2150-8097.
doi:10.14778/3828612.3828616

convex programming [74] based algorithms are proposed. The state-of-the-art algorithms are LTDS [64] and IPPV [74]. The LTDS [64] is a maximum flow-based solution. Specifically, LTDS iteratively finds a CDS and verifies whether it is an LhCDS via a maximum flow computation. However, LTDS is not scalable to large-scale graphs, as it requires executing a maximum flow computation on a large flow network. To overcome these limitations, IPPV [74] adopts a convex programming-based approach. It repeatedly solves a convex optimization problem to identify candidate LhCDSes and then verifies whether these candidates are valid LhCDSes through a maximum flow computation. In addition, IPPV integrates pruning strategies to eliminate vertices that cannot belong to any LhCDS. IPPV has been shown to be more general and more efficient, as demonstrated in [74]. Based on our careful analysis, we identify two key limitations of IPPV.

Limitations. (1) Prohibitive time complexity. To find the top-k LhCDSes, IPPV requires $O(N_{\text{flow}} \cdot (n + \Psi_h)^2 \cdot (n + \Psi_h \cdot h))$ time, where n is the number of vertices in G, Ψ_h is the number of h-cliques, and N_{flow} is the number of maximum-flow computations, at most k times the number of candidate LhCDSes. Consequently, when k is large or the graph contains many candidate LhCDSes, N_{flow} can grow to thousands, making IPPV highly time-consuming.

(2) Unnecessarily large search space. IPPV needs to explore candidate LhCDSes over almost the entire graph. In practice, however, only a small fraction of vertices can contribute to an LhCDS. This is mainly because a CDS maximizes the h-clique density by excluding vertices that participate in fewer h-cliques. This observation motivates us to first restrict the search to a relatively small subgraph before running the algorithm to find each LhCDS.

Our technical contributions. In this paper, we propose an efficient divide-and-conquer based algorithm for the LhCDS problem (called DCLDS), which runs in $O((n + \Psi_h)^2 \cdot (n + \Psi_h \cdot h))$ time, and removes the N_{flow} factor compared with the latest work IPPV [74]. Specifically, we first propose a new concept, the *locally h-clique dense subgraph*, and show that it admits a natural decomposition that generates a hierarchical tree structure, analogous to the well-known k-core decomposition in network science [8]. Afterwards, we prove that LhCDSes correspond exactly to some leaf nodes of this tree. This property enables the design of an efficient divide-and-conquer algorithm to extract the top-k LhCDSes, since the search can be localized to leaf nodes rather than the entire hierarchy.

To further improve efficiency, we propose a novel *clique orientation-based graph partition* method, which decomposes the original graph into a series of smaller subgraphs. The LhCDSes are then identified within each partition, thereby substantially reducing the overall search space. Specifically, we prove that after partitioning, every LhCDS of the original graph is fully contained within one of the subgraphs, ensuring no loss of results. By doing so, our method narrows the search space.

Extensive experimental evaluations on 15 real-world large graphs show that DCLDS achieves higher efficiency and scalability than the state-of-the-art algorithms on all datasets.

In summary, our main contributions are as follows.

- We propose a divide-and-conquer-based LhCDS discovery method to break the theoretical bottleneck of existing algorithms, achieving an improved time complexity.
- To further reduce the search space, we propose a novel clique-orientation-based graph partitioning technique and prove that each LhCDS is fully contained within one of the resulting subpartitions. This property allows DCLDS to identify all LhCDSes on these smaller subgraphs, rather than on the entire graph.
- We conduct experiments on 15 real-world large graphs to demonstrate the efficiency and scalability of our algorithm.

Outline. We introduce the preliminaries in Section 2. Section 3 analyzes the limitations of state-of-the-art algorithms. We introduce our divide-and-conquer method in Section 4, and present our DCLDS algorithm in Section 5. The experimental results are reported in Section 6. We review the related works in Section 7 and conclude in Section 8. For lack of space, all detailed proofs in this paper are included in our technical report [86] and we only show the proof sketches in our manuscript.

2 PRELIMINARIES

In this section, we first present the formal definition of the LhCDS problem and some key definitions.

Table 1: Notations and meanings.

Notation	Meaning
$G = (V, E)$	a graph with vertex set V and edge set E
$G[S]$	the subgraph of G induced by vertices in S
$\Psi_h(G)$	the set of h-cliques in G
$\Psi_h(v, G)$	the set of h-cliques containing v in G
$\Psi_h(X, Y)$	the set of cross h-cliques between $G[X]$ and $G[Y]$
$\mathcal{D}$	the locally h-clique densest subgraph of G
$\rho_h(H)$	the h-clique density of subgraph H
$\{B_i\}_{i=0}^{r}$	the nested chain of locally h-clique dense subgraphs of G.
$\eta_h(H)$	the h-clique compactness

In this paper, we consider an unweighted and undirected graph G=(V, E), where n and m denote the number of vertices and edges in the graph, respectively. Given a vertex set S, we use $G[S]$ = $(S, E(S))$ to denote the subgraph of G induced by S, where $E(S)$= $\{(u, v) \in E \mid u, v \in S\}$ denotes the set of edges in G contained in S. For a given graph H, we also denote its sets of vertices and edges by $V(H)$ and $E(H)$, respectively.

A h-clique is a complete graph with a set C of h vertices where there is an edge between every pair of vertices. In the case without ambiguity, we simply refer to a h-clique by its set of vertices. We use $\Psi_h(G)$ to represent the set of h-cliques in G. Denote by $\Psi_h(G)$ = $\{C \subseteq V \mid C$ is a h-clique of $G\}$. For each vertex $v \in V(G)$, we use $\Psi_h(v, G)$ to denote the set of h-cliques containing v in the graph G ($h \geq 3$). For two vertex sets X and Y, we use $\Psi_h(X, Y)$ to denote the set of *cross h-cliques* between $G[X]$ and $G[Y]$, i.e.,

$$\Psi_h(X, Y) = \{\, C \in \Psi_h(G) \mid C \cap X \neq \emptyset \text{ and } C \cap Y \neq \emptyset \,\}.$$

In other words, each h-clique $C \in \Psi_h(X, Y)$ contains at least one vertex from X and at least one vertex from Y. We summarize the frequently used notations in Table 1.

We now formally present the definition of h-clique density.

Definition 2.1 (h-clique density [27, 38, 56, 68, 72, 84]). Given a subgraph H of a graph G and a positive integer h, the h-clique

density of H, denoted by $\rho_h(H)$, is the average number of h-cliques per vertex in H, i.e.,

$$\rho_h(H) = \frac{|\Psi_h(H)|}{|V(H)|}. \quad (1)$$

Definition 2.2 (h-clique λ-compact [74]). Given a graph G=(V, E), we call G an h-clique λ-compact graph, if the following conditions are satisfied:

- G is the connected component.
- Removing any vertex set $S \subseteq V$ from G would result in the removal of at least $\lambda \times |S|$ h-cliques.

Intuitively, an h-clique λ-compact graph characterizes a connected subgraph whose h-clique density is at least λ. Hence, larger λ values enforce stronger structural cohesion and exclude loosely connected regions. This property makes the notion a natural foundation for defining LhCDS, where a feasible solution must be a maximal h-clique λ-compact graph. We illustrate this concept with an example below:

Example 1. *In Figure 1, $\mathcal{H}_1$ and $\mathcal{H}_2$ are 3-clique 1-compact and 3-clique $\frac{1}{2}$-compact graphs, respectively. This holds because both $\mathcal{H}_1$ and $\mathcal{H}_2$ are connected, and removing any vertex subset S eliminates at least $1 \times |S|$ and $\frac{1}{2} \times |S|$ triangles, respectively. Furthermore, both graphs are L3CDSes, since they are maximal subgraphs that satisfy the 3-clique λ-compactness property.*

It is easy to see that any h-clique λ-compact graph is also h-clique λ'-compact for any $\lambda' < \lambda$. A subgraph H of G is a maximal h-clique λ-compact subgraph of G if every proper supergraph of H in G is not h-clique λ-compact. The concept of locally h-clique densest subgraph (LhCDS) is defined based on the maximal h-clique λ-compact subgraph, as follows.

Definition 2.3 (locally h-clique densest subgraph (LhCDS) [49, 74]). A subgraph $\mathcal{D}$ of G is a locally h-clique densest subgraph if $\mathcal{D}$ is the maximal h-clique $\rho_h(\mathcal{D})$-compact subgraph in G.

LhCDSes of a graph G satisfy the following properties:

- The h-clique density of any subgraph of an LhCDS $\mathcal{D}$ is at most $\rho_h(\mathcal{D})$, i.e., any subgraph of an LhCDS is not denser than itself.
- Any proper supergraph of an LhCDS $\mathcal{D}$ is not h-clique $\rho_h(\mathcal{D})$-compact, i.e., any proper supergraph of an LhCDS is not as compact as itself.
- LhCDSes are disjoint, i.e., $V(\mathcal{D}) \cap V(\mathcal{D}')=\emptyset$ for any two distinct LhCDS $\mathcal{D}$ and $\mathcal{D}'$.

Note that an edge in a graph G is a 2-clique; therefore, the intensively studied LDS problem [53, 60] can be seen as an instance of the LhCDS problem when h = 2. Similarly, the LDS problem [49] is exactly the L3CDS problem. Therefore, the LhCDS problem studied in this paper is a more general problem. Next, we formally present the definition of the LhCDS problem:

Problem 1 (Locally h-clique densest subgraph (LhCDS) problem [25, 45, 49, 51, 74]). *Given a graph G, an integer k, and an integer $h \geq 3$, the locally h-clique densest subgraph (LhCDS) problem aims to find the top-k locally h-clique densest subgraphs with the highest h-clique densities in G.*

Table 2: Complexities of representative LhCDS algorithms. Compared with IPPV, our method removes the dependence on N_{flow}, yielding a better time complexity.

Method	Space Complexity (O)	Time Complexity (O)
Basic [35]	$n + \|\Psi_h\|$	$(n \cdot \Psi_h)^3 \cdot L$
LTDS [64]	$n + \|\Psi_h\|$	$(n + \Psi_h h)^2 \cdot (n + \Psi_h) \cdot \log^2(n + \|\Psi_h\|)$
IPPV [74]	$n + \|\Psi_h\|$	$N_{\text{flow}} \cdot (n + \Psi_h)^2 \cdot (n + \Psi_h h)$
DCLDS **(ours)**	$n + \|\Psi_h\|$	$(n + \Psi_h)^2 \cdot (n + \Psi_h h)$

* Note: n denotes the number of vertices in the graph G, and L is the number of iterations; $|\Psi_h|$ denotes the number of h-cliques in G.
* N_{flow} denotes the number of maximum flow calls.

Example 2. *In Figure 1, let h=3 and k=2, $\mathcal{H}_1$ and $\mathcal{H}_2$ are identified as the top-2 L3CDSes: they are non-overlapping, and no other subgraph has a 3-clique density greater than theirs.*

3 REVIEW OF THE STATE-OF-THE-ART ALGORITHMS

In this section, we review the two state-of-the-art LhCDS algorithms LTDS [49] and IPPV [74] and further analyze their limitations.

3.1 The LTDS algorithm

The core idea of LTDS is to iteratively identify the h-clique densest subgraph, derive the maximal h-clique λ-compact subgraph based on its h-clique density, and then verify whether the h-clique densest subgraph is an LhCDS.

In LTDS (Algorithm 1), it first initializes $\mathcal{L} \leftarrow \emptyset$, and $G' \leftarrow G$ (line 1), and finds the top-k LhCDSes via a while-loop (lines 2-9). Specifically, in each iteration of LTDS, it performs the following three steps: (1) find a CDS $\mathcal{D}_h$ from the current graph G'; (2) use the h-clique density of $\mathcal{D}_h$ to derive the maximal h-clique $\rho_h(\mathcal{D}_h)$-compact subgraph of G (line 4), and (3) verify whether this CDS is an LhCDS. If so, it is added to the result set (lines 5-7). This process repeats until the top-k LhCDSes are found, with $\mathcal{D}_h$ removed from G' before the next iteration (lines 8-9).

Algorithm 1: LTDS [64]

```
input : A graph G, two integers k and h.
output: top-k LhCDSes
1  L = ∅; G' ← G;
2  while |L| < k and Ψ_h(G') ≠ ∅ do
3      D_h ← find the h-clique densest subgraph from G';
4      C_h ← derive the h-clique ρ_h(D_h)-compact subgraph from G;
5      foreach connected component g of D_h do
6          if g ⊆ C_h and |L| < k then
7              L ← L ∪ {g};
8      S' = V(G') \ V(D_h);
9      Let G' be the subgraph of G induced by S';
10 return L;
```

Limitation of LTDS. The key limitation of LTDS is that it requires computing maximum flows over the almost entire graph for each LhCDS, making the approach computationally expensive.

Table 3: The values of N_{flow} (h=3).

Dataset	k=1	k=5	k=10	k=15	k=20
CA-CondMat	2	48	56	62	63
Amazon	3	142	147	152	157
DBLP	1	55	59	62	67
Email	88	99	105	109	110
loc-gowalla	136	147	201	205	209
web-Stanford	255	264	271	988	1,010

3.2 The IPPV algorithm

To support arbitrary values of h and improve efficiency, the current state-of-the-art algorithm for computing top-k LhCDSes is proposed in [74], which leverages convex programming via the `Frank-Wolfe` method. Note that the `Frank-Wolfe` algorithm [42] is a well-known method for solving convex optimization problems. As shown in [74], the LhCDSes can be found via solving the convex programming formulation. Specifically, IPPV follows the following three steps:

(1) **Optimization.** IPPV performs T optimization iterations and, based on the resulting solutions, extracts a set of candidate LhCDSes.
(2) **Verification.** IPPV then sequentially checks each candidate LhCDS using a maximum flow. If the candidate is verified as a valid LhCDS, it is appended to the result set. Otherwise, the algorithm repeats Steps (1) and (2) to generate new candidates.
(3) **Termination.** The above process continues until the top-k LhCDSes are found.

Limitation of IPPV. We list the two fundamental drawbacks of IPPV here. First, the algorithm suffers from high computational cost: computing the top-k LhCDSes requires $O(N_{\text{flow}} \cdot (n+\Psi_h)^2 \cdot (n+\Psi_h \cdot h))$, where N_{flow} may grow up to k times the number of candidate LhCDSes. To illustrate this cost, we report the values of N_{flow} obtained by IPPV on six datasets with $h=3$ and k ranging from 1 to 20. As shown in Table 3, the value of N_{flow} increases with larger k, and on some datasets it can reach up to thousands, which clearly makes IPPV much slower to run. Second, IPPV repeatedly searches for candidate LhCDSes across nearly the entire graph, resulting in extensive redundant computations and high runtime overhead. Hence, the limitations above motivate us to design an efficient algorithm, a theoretically and practically better algorithm.

4 DIVIDE-AND-CONQUER METHOD FOR LhCDS DISCOVERY

In this section, we present the design of our divide-and-conquer algorithm for LhCDS discovery.

Roadmap. The design of our algorithm proceeds as follows: (1) We first demonstrate that maximal h-clique λ-compact subgraphs across different λ values naturally form a hierarchical structure, where each LhCDS corresponds to a leaf node that is not connected to any other leaf nodes with higher h-clique density, and their densities monotonically increase from left to right (see Section 4.1). (2) To efficiently identify these structures, we introduce the concept of the *locally h-clique dense subgraph* and prove that it forms the same hierarchical structure as the maximal h-clique λ-compact subgraphs across all λ values (see Section 4.2). (3) Based on this theoretical equivalence and the hierarchical property, we develop an efficient divide-and-conquer algorithm to compute all LhCDSes.

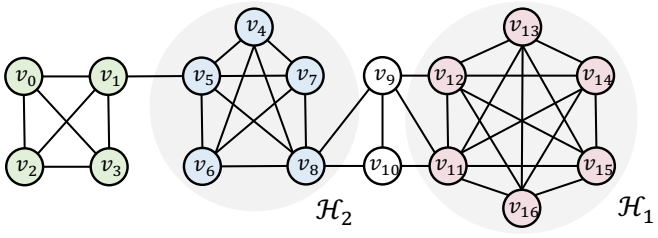


Figure 2: An example graph to illustrate our algorithm.

4.1 Hierarchical structure of maximal compact subgraphs

In this section, we demonstrate that all maximal h-clique λ-compact subgraphs across different λ values collectively form a hierarchical tree structure. Furthermore, we show that the leaf nodes of this tree — those that are not connected to any denser locally h-clique dense subgraphs — correspond precisely to the LhCDSes.

Union property. We first establish that overlapping or adjacent h-clique λ-compact subgraphs can be merged without violating compactness.

Lemma 4.1. *Given a graph G and two h-clique λ-compact subgraphs $G[X]$ and $G[Y]$, if $X \cap Y \neq \emptyset$ or there exists a cross h-clique between them (i.e., $\Psi_h(X,Y) \neq \emptyset$), then $G[X \cup Y]$ is also h-clique λ-compact subgraph of G.*

Proof Sketch. If $X \cap Y \neq \emptyset$ or the two subgraphs share at least one h-clique (i.e., $\Psi_h(X,Y) \neq \emptyset$), then the induced subgraph $G[X \cup Y]$ is connected. Let $S \subseteq X \cup Y$ be any non-empty vertex subset to be removed, and decompose it as $S_X = S \cap X$ and $S_Y = S \cap (Y \setminus X)$. Since every h-clique in $G[X]$ (resp. $G[Y]$) that intersects S_X (resp. S_Y) also exists in $G[X \cup Y]$, the number of removed h-cliques in $G[X \cup Y]$ is no smaller than in either subgraph. Given that both X and Y are h-clique λ-compact, deleting S_X and S_Y eliminates at least $\lambda \cdot |S_X|$ and $\lambda \cdot |S_Y|$ h-cliques, respectively. Summing up yields $|\Delta\Psi_h(G[S])| \geq \lambda(|S_X| + |S_Y|) = \lambda \cdot |S|$. Hence, $G[X \cup Y]$ also satisfies the h-clique λ-compact condition. □

Hierarchy property. The above lemma implies that any two distinct maximal h-clique λ-compact subgraphs at the same λ are disjoint; otherwise, their union would remain λ-compact, contradicting maximality. That is, for maximal h-clique λ-compact subgraphs corresponding to different λ values, we establish that they are either nested or disjoint, thus inducing a hierarchical structure by the following lemma.

Lemma 4.2. *Let $G[X]$ and $G[Y]$ be maximal h-clique λ-compact and λ'-compact subgraphs, respectively, with $\lambda > \lambda'$. Then either $X \subseteq Y$ or $X \cap Y = \emptyset$ and $\Psi_h(X,Y) = \emptyset$.*

Proof. Assume for contradiction that $X \not\subseteq Y$ and they are not disjoint (i.e., $X \cap Y \neq \emptyset$ or $\Psi_h(X,Y) \neq \emptyset$). Since $\lambda > \lambda'$, $G[X]$ is also h-clique λ'-compact. By Lemma 4.1, $G[X \cup Y]$ is h-clique λ'-compact

subgraph. Because $X \cup Y$ strictly contains Y, this contradicts the maximality of $G[Y]$. □

Example 3. *Consider the graph illustrated in Figure 2 with $h = 3$. Figure 3(a) presents the hierarchical tree of maximal h-clique λ-compact subgraphs corresponding to different λ values. We observe that a maximal h-clique λ-compact subgraph at a higher λ value is always nested within that of a lower λ.*

Characterizing LhCDSes. We now establish the necessary and sufficient condition under which a maximal h-clique λ-compact subgraph corresponds to an LhCDS. To do this, we first define a measure of compactness to link λ-compactness with h-clique density.

Definition 4.3 (h-clique compactness). Given a graph $G = (V, E)$, the *h-clique compactness* of a connected subgraph H, denoted $\eta_h(H)$, is the maximum value of λ such that H is h-clique λ-compact subgraph.

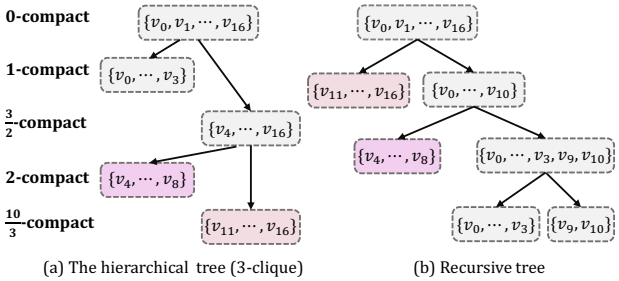
The h-clique compactness connects the LhCDS and the maximal h-clique λ-compact subgraph based on the following theorem.

Theorem 4.4. *A maximal h-clique λ-compact subgraph $\mathcal{H}$ is an LhCDS if and only if it contains no h-clique λ'-compact subgraph with $\lambda' > \eta_h(\mathcal{H})$.*

Finally, we establish the connection between the hierarchical structure and the LhCDSes. Lemmas 4.1–4.2 and Theorem 4.4 together reveal that all maximal h-clique λ-compact subgraphs form a hierarchical tree structure, where (1) subgraphs with larger λ values are nested within those with smaller λ, and (2) leaf nodes in this hierarchy, those not connected to sibling subgraphs with higher h-clique compactness correspond precisely to the LhCDSes. Therefore, discovering all LhCDSes is equivalent to identifying all leaf nodes in the hierarchical tree of maximal h-clique λ-compact subgraphs.

Example 4. *As shown in Figure 3(a), there are three leaf nodes. Among them, the subgraphs induced by $\{v_4, \ldots, v_8\}$ and $\{v_{11}, \ldots, v_{16}\}$ are LhCDSes, whereas the leaf node $\{v_0, \ldots, v_3\}$ is not, since it is connected to $\{v_4, \ldots, v_8\}$, which has a higher h-clique density. To be more specific, let us suppose that the subgraph induced by $\{v_0, \ldots, v_3\}$ (denoted by H) is an LhCDS. By definition, no proper supergraph of H can remain 3-clique $\rho_3(H)$-compact, where $\rho_3(H) = 1$. However, the subgraph induced by $\{v_0, \ldots, v_8\}$ is also 3-clique $\rho_3(H)$-compact. This contradicts the definition of LhCDS. Therefore, the subgraph induced by $\{v_0, \ldots, v_3\}$ cannot be an LhCDS.*

However, directly constructing the entire hierarchical tree of maximal h-clique λ-compact subgraphs is computationally inefficient. This is because, building the full hierarchy requires enumerating maximal h-clique λ-compact subgraphs for all feasible λ values, which incurs substantial computational overhead. Moreover, such exhaustive construction is unnecessary for our objective, since we only aim to retrieve the top-k LhCDSes rather than materialize the complete hierarchy. To address this issue, we introduce the notion of a *locally h-clique dense subgraph*, which preserves the same hierarchical structure while enabling more efficient computation. The hierarchy induced by locally h-clique dense subgraphs naturally supports a divide-and-conquer framework that reduces the search space when computing candidate subgraphs and allows early termination once the top-k results are identified. Consequently, the algorithm avoids exploring unpromising branches of the search tree and eliminates the overhead of constructing the full hierarchy, thereby achieving superior runtime performance.



(a) The hierarchical tree (3-clique) (b) Recursive tree

Figure 3: Illustration of the hierarchical tree.

4.2 Locally h-clique dense subgraph

In this subsection, we introduce a new concept, termed the *locally h-clique dense subgraph*, which preserves the same hierarchical structure while allowing for more efficient computation and analysis.

Definition 4.5 (h-clique outer density). Given a graph G and two disjoint vertex subsets X and Y, the *h-clique outer density* of X with respect to Y, denoted by $\rho_h(X, Y)$, is defined as

$$\rho_h(X, Y) = \frac{|\Psi_h(G[X])| + |\Psi_h(X, Y)|}{|X|}.$$

Based on this definition, we formally define the concept of a locally h-clique dense subgraph.

Definition 4.6 (Locally h-clique dense subgraph). Given a graph $G = (V, E)$, a subgraph induced by a vertex set $S \subseteq V$ is called a *locally h-clique dense subgraph* if there exist no non-empty subsets $X \subseteq S$ and $Y \subseteq V \setminus S$ such that $\rho_h(X, S \setminus X) \leq \rho_h(Y, S)$.

Example 5. *Consider the graph illustrated in Figure 2 with $h = 3$. The subgraph constructed by $\{v_0, \cdots, v_3\}$ is a locally 3-clique dense subgraph. However, it is not an L3CDS since it is connected with a denser locally 3-clique dense subgraph $\{v_4, \cdots, v_8\}$. By contrast, $S = \{v_9, \cdots, v_{16}\}$ is not a locally 3-clique dense subgraph since there exists $X = \{v_9, v_{10}\}$ and $Y = \{v_8\}$ such that $\rho_3(X, S \setminus X) = 1 \leq \rho_3(Y, S) = 1$.*

To ensure terminological clarity, throughout this paper, we use the full term "locally h-clique dense subgraph" for the concept in Definition 4.6, while reserving the acronym "LhCDS" for "locally h-clique **densest** subgraph."

Decomposition of locally h-clique dense subgraphs. We next show that locally h-clique dense subgraphs exhibit an inherent hierarchical structure due to their nested relationships. The following theorem establishes the nesting property that underlies this hierarchy.

Theorem 4.7. *Let $G[U]$ and $G[W]$ be two locally h-clique dense subgraphs. Then either $U \subseteq W$ or $W \subseteq U$.*

Proof. Assume for contradiction that neither subgraph is contained within the other. Let $X = U \setminus W$ and $Y = W \setminus U$; both are non-empty. Based on the definition 4.6, we must have either $\rho_h(X, U \cap W) \leq \rho_h(Y, U \cap W])$ or $\rho_h(X, U \cap W) > \rho_h(Y, U \cap W)$. If the former condition holds, we have $\rho_h(X, U \setminus X) = \rho_h(X, U \cap W) \leq \rho_h(Y, U \cap W) \leq \rho_h(Y, U)$, where the last inequality follows since $\Psi_h(Y, U \cap W) \subseteq \Psi_h(Y, U)$. This contradicts the assumption that U is locally h-clique dense. The opposite case, $\rho_h(X, U \cap W) > \rho_h(Y, U \cap W)$, leads to a symmetric contradiction for W. Hence, one subgraph must be contained in the other. □

Theorem 4.7 implies that all locally h-clique dense subgraphs are mutually nested, forming a hierarchical chain analogous to the structure of k-cores [8].

Corollary 4.8. *The locally h-clique dense subgraphs form an ordered hierarchy of induced subgraphs of G, defined by a vertex sequence $B_0 \subset B_1 \subset \cdots \subset B_r$, wherer $\leq |V|$, and their outer densities satisfy $\rho_h(B_i \setminus B_{i-1}, B_{i-1}) > \rho_h(B_{i+1} \setminus B_i, B_i)$ for all $1 \leq i < r$.*

This corollary ensures that all locally h-clique dense subgraphs of a graph can be organized into a hierarchical tree structure.

We establish the equivalence between two hierarchical structures: the hierarchy of locally h-clique dense subgraphs and the hierarchy of maximal h-clique λ-compact subgraphs. This equivalence provides the theoretical foundation of our algorithm, ensuring that both perspectives yield an identical decomposition of G.

Bridging function. We first introduce a key optimization function that connects these two concepts.

Definition 4.9 (h-clique net-gain function $F_h(\lambda)$). Given a graph $G = (V, E)$ and a real value $\lambda \geq 0$, the *h-clique net-gain function* $F_h(\lambda)$ returns the vertex subset $S \subseteq V$ that maximizes the following objective:

$$F_h(\lambda) = \arg\max_{S \subseteq V} \left(|\Psi_h(G[S])| - \lambda |S| \right).$$

Ties are broken in favor of the largest such subset. Intuitively, $F_h(\lambda)$ balances the "gain" from including h-cliques against the "cost" of adding vertices, where λ represents the per-vertex penalty.

Connection to maximal λ-compact subgraphs. We next show that the structure induced by $F_h(\lambda)$ coincides with the set of maximal h-clique λ-compact subgraphs.

Lemma 4.10. *The connected components of the induced subgraph $G[F_h(\lambda)]$ are exactly the maximal h-clique λ-compact subgraphs of G.*

Proof. Let $\mathcal{M}_\lambda$ denote the collection of maximal h-clique λ-compact subgraphs, and let $S^* = F_h(\lambda)$. (1) First, we show that every connected component of $G[S^*]$ must be a maximal h-clique λ-compact subgraph. Indeed, if some component were not maximal h-clique λ-compact, then it could be replaced by a subgraph with larger h-clique density, which would strictly increase the net-gain value. This contradicts the optimality of S^*. (2) Second, we show that every maximal h-clique λ-compact subgraph is contained in S^*. Fix any $M \in \mathcal{M}_\lambda$, and suppose for contradiction that $V(M) \nsubseteq S^*$. Let $g_\lambda(S) = |\Psi_h(G[S])| - \lambda |S|$. Since M is h-clique λ-compact, no subset of $V(M)$ has larger net gain than $V(M)$. Thus, for any $A \subseteq V(M)$,

$$g_\lambda(V(M)) \geq g_\lambda(A).$$

In particular, this holds for $A = S^* \cap V(M)$.

Moreover, g_λ is a supermodular function. Applying supermodularity to S^* and $V(M)$, we obtain

$$g_\lambda(S^* \cup V(M)) + g_\lambda(S^* \cap V(M)) \geq g_\lambda(S^*) + g_\lambda(V(M)).$$

Together with

$$g_\lambda(V(M)) \geq g_\lambda(S^* \cap V(M)),$$

this implies

$$g_\lambda(S^* \cup V(M)) \geq g_\lambda(S^*).$$

Therefore, adding $V(M)$ to S^* does not decrease the objective value, so $S^* \cup V(M)$ is also an optimal solution. However, since $V(M) \nsubseteq S^*$, the set $S^* \cup V(M)$ strictly contains S^*, contradicting the tie-breaking rule that selects the inclusion-wise largest optimal set. Hence, $V(M) \subseteq S^*$ for every $M \in \mathcal{M}_\lambda$. □

Connection to locally h-cique dense subgraphs. We now show that varying λ produces a sequence of $F_h(\lambda)$ subgraphs that are nested with respect to inclusion, thereby forming the hierarchical structure of locally h-clique dense subgraphs.

Proposition 4.11. *Let $\{B_i\}_{i=0}^r$ denote the nested chain of the vertex sets of locally h-clique dense subgraphs of G. For any $\rho_h(B_{i+1} \setminus B_i, B_i) < \lambda \leq \rho_h(B_i \setminus B_{i-1}, B_{i-1})$, we have $F_h(\lambda) = B_i$.*

Proof. By the optimality of $F_h(\lambda)$, removing any subset $X \subseteq S^* = F_h(\lambda)$ cannot improve the objective, implying $\rho_h(X, S^* \setminus X) \geq \lambda$ for all non-empty $X \subseteq S^*$. Similarly, adding any external subset $Y \subseteq V \setminus S^*$ strictly decreases the objective, implying $\rho_h(Y, S^*) < \lambda$. These conditions exactly match those defining a locally h-clique dense subgraph. Now fix an index i such that $\rho_h(B_{i+1} \setminus B_i, B_i) < \lambda \leq \rho_h(B_i \setminus B_{i-1}, B_{i-1})$. If $F_h(\lambda) = B_j$ with $j > i$, then removing $B_j \setminus B_{j-1}$ would increase the net-gain value. If $j < i$, adding $B_{j+1} \setminus B_j$ would not reduce the net-gain value. Therefore, $j = i$, and hence $F_h(\lambda) = B_i$. □

By combining Lemma 4.10 and Proposition 4.11, we arrive at the following theorem.

Theorem 4.12. *The hierarchy of locally h-clique dense subgraphs is identical to the hierarchy of maximal h-clique λ-compact subgraphs. In particular, the connected components of each locally dense subgraph correspond exactly to the maximal λ-compact subgraphs, and the leaf nodes of this hierarchy are the LhCDSes.*

This theorem formally establishes the equivalence and interchangeability of the two hierarchies. In the next section, we leverage this equivalence to design a divide-and-conquer algorithm that efficiently identifies the top-k LhCDSes.

4.3 Our divide-and-conquer algorithm

In this subsection, we present a divide-and-conquer algorithm to construct the hierarchical tree structure of the locally h-clique dense subgraph decomposition. Recall that each leaf node in this hierarchy corresponds to an LhCDS if and only if it is not connected to any subgraph with a higher h-clique density. Our divide-and-conquer strategy recursively explores the graph from regions of higher to lower density, enabling efficient identification of all LhCDSes.

Before presenting the algorithm, we first introduce two key lemmas that form its theoretical foundation.

Lemma 4.13. *Let $\{B_i\}_{i=0}^{r}$ be the sequence of locally h-clique dense subgraphs. For any two indices i and j such that $0 \le i < j \le r$, let $B_l = F_h(\lambda)$ with $\lambda = \rho_h(B_j \setminus B_i, B_i)$ Then:*

- *If $i + 1 < j$, then $i < l < j$.*
- *If $i + 1 = j$, then $l = j$.*

Proof Sketch. By corollary 4.8, the outer densities $\rho_h(B_k \setminus B_{k-1}, B_{k-1})$ strictly decrease along the sequence $\emptyset = B_0 \subset B_1 \subset \cdots \subset B_r = V$. Expressing $\rho_h(B_k \setminus B_{k-1}, B_{k-1}) = c_k/v_k$, the combined density $\rho_h(B_j \setminus B_i, B_i) = \frac{\sum_{k=i+1}^{j} c_k}{\sum_{k=i+1}^{j} v_k}$ lies between c_{i+1}/v_{i+1} and c_j/v_j, with equality only when $i + 1 = j$; otherwise, it lies strictly between them. Hence $\rho_h(B_j \setminus B_{i+1}, B_{i+1}) < \rho_h(B_j \setminus B_i, B_i) < \rho_h(B_{i+1} \setminus B_i, B_i)$, implying $i < l < j$ (and $l = j$ when $i + 1 = j$). □

This lemma directly yields the recursive principle driving our divide-and-conquer method.

Corollary 4.14. *Given two locally h-clique dense subgraphs $G[X]$ and $G[Y]$ such that $X \subset Y$, and let $Z = F_h(\lambda)$ where $\lambda = \rho_h(Y \setminus X, X)$. Then, we have:*

- *If $Z = Y$, no locally h-clique dense subgraph $G[W]$ exists such that $X \subset W \subset Y$.*
- *If $Z \neq Y$, then $G[Z]$ is a locally h-clique dense subgraph satisfying $X \subset Z \subset Y$.*

Proof. Let $X = B_i$ and $Y = B_j$ for some $i < j$. If $j = i + 1$, Lemma 4.13 gives $Z = Y$, implying no intermediate subgraph. If $j > i + 1$, the lemma ensures the existence of B_l for some $i < l < j$, i.e., an intermediate locally dense subgraph. □

Recursive construction. Based on lemma 4.13 and corollary 4.14, we design a recursive divide-and-conquer algorithm to construct the hierarchical tree of locally h-clique dense subgraphs. Given two nested locally h-clique dense subgraphs $B_i \subset B_j$, the algorithm finds an intermediate subgraph $B_l = F_h(\rho_h(B_j \setminus B_i, B_i))$ and recursively solves the two subproblems (B_i, B_l) and (B_l, B_j). The recursion terminates when $F_h(\lambda) = B_j$, which corresponds to a leaf node.

Starting from the trivial boundaries $X = \emptyset$ and $Y = V$, the algorithm progressively refines the hierarchy by discovering intermediate layers until all locally h-clique dense subgraphs are identified. When $X = \emptyset$ and $Y = V$, the first call extracts the top layer B_0 from $Y \setminus X$, partitioning V into two disjoint regions that serve as new recursion boundaries. Repeating this process yields a complete hierarchical decomposition.

Identifying LhCDSes. As shown in theorem 4.4, each LhCDS corresponds to a leaf node in the hierarchy tree that has no connection to any denser locally h-clique dense subgraph. Within our recursive framework, such leaf nodes correspond to regions $B_i \setminus B_{i-1}$ that contain no further nested locally h-clique dense subgraphs. The following lemma provides a precise criterion for identifying these LhCDSes.

Lemma 4.15. *Given the sequence of locally h-clique dense subgraphs $\{B_i\}_{i=0}^{r}$, the LhCDSes of G are exactly the connected components of $G[B_i \setminus B_{i-1}]$ that have no cross-edges with B_{i-1}, for all $i \in \{1, \ldots, r\}$.*

Algorithm 2: DeriveLhCDS

input : A graph G, two integers k and h.
output : Top-k LhCDSes in G.

```
1  L ← ∅;
2  Divide(∅, V, k, L);
3  return L;
4  Function Divide(X, Y, k, L):
5      λ ← ρ_h(Y \ X, X); Z ← F_h(λ); Z ← F_h(λ)
6      if Z = Y then
7          foreach connected component g of G[Y \ X] do
8              if g has no cross-edges with X then
9                  L ← L ∪ {g};
10     else
11         if |L| < k then
12             Divide(X, Z, k − |L|, L);
13         if |L| < k then
14             Divide(Z, Y, k − |L|, L);
```

Proof. An LhCDS must be included entirely within B_i but be disjoint from B_{i-1}; therefore, it can only exist inside the induced subgraph $G[B_i \setminus B_{i-1}]$. Intuitively, B_{i-1} represents a denser "inner layer", while $B_i \setminus B_{i-1}$ corresponds to the newly added region in building the hierarchical tree. If a connected component H of $G[B_i \setminus B_{i-1}]$ has no cross-h-cliques linking it to B_{i-1}, then H cannot merge with any other subgraph to form a denser or larger compact structure. According to theorem 4.4, such a component is precisely an LhCDS. Conversely, if a connected component of $G[B_i \setminus B_{i-1}]$ has cross-h-cliques with B_{i-1}, it can still be extended into a larger subgraph with higher h-clique compactness. □

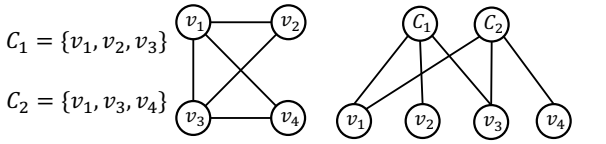
Lemma 4.15 ensures that the base cases of the recursion yield exactly the LhCDSes. The procedure is summarized in Algorithm 2.

Complexity analysis. In the worst case, the `Divide` function is invoked up to $2r - 3$ times, where r is the number of locally h-clique dense subgraphs in G and $r \le n + |\Psi_h|$. Since each invocation computes $F_h(\lambda)$ via a maximum-flow procedure [71] on the h-clique augmented network [27], whose size is $|V| = n + |\Psi_h|$ vertices and $|\mathcal{E}| = n + |\Psi_h|h$ edges, we instantiate the flow complexity using Orlin's state-of-the-art $O(|V||\mathcal{E}|)$ algorithm [59]. Therefore, a single computation costs $O\big((n + |\Psi_h|)\,(n + |\Psi_h|h)\big)$ time. Since the procedure is invoked at most $n + |\Psi_h|$ times, the overall time complexity of the proposed method is: $O\big((n + |\Psi_h|)^2 \cdot (n + |\Psi_h|h)\big)$.

Example 6. *Let us continue with the graph illustrated in Figure 2, where $h = 3$. Figure 3(b) shows the recursive tree generated by Algorithm 2. We can observe that the first two leftmost leaf nodes correspond to the LhCDSes, as they are not connected to any other leaf nodes with higher h-clique density. In contrast, the remaining leaf nodes, $\{v_0, \ldots, v_3\}$ and $\{v_9, v_{10}\}$, are not LhCDSes because they are connected to leaf nodes with higher h-clique density.*

5 OUR DCLDS ALGORITHM

In this section, we present our proposed method, DCLDS, which builds upon the DeriveLhCDS framework introduced in Section 4. We first introduce a clique-orientation-based graph partitioning



(a) A graph G (b) h-clique augmented graph $\mathcal{G}_h$

Figure 4: An example to illustrate the augmented graph.

technique designed to further reduce the search space of DeriveLhCDS. Building on this foundation, we then describe our DCLDS algorithm.

5.1 Clique orientation-based graph partition

In this subsection, we introduce a novel concept, called the *clique orientation-based graph partition*, which aims to reduce the search space of DeriveLhCDS. The key idea is to partition the input graph into disjoint regions guided by *clique orientations*, such that each LhCDS is entirely contained within one region, without crossing multiple partitions. By isolating the graph into localized partitions, the overall search space for discovering LhCDS is significantly reduced. To this end, we first introduce the h-clique oriented augmented graph, which serves as the cornerstone of the proposed graph partition.

Construction of the h-clique oriented augmented graph. Let $G = (V, E)$ be an undirected graph. We construct its *h-clique augmented graph*, denoted as $\mathcal{G}_h = (V \cup V_C, E')$, by transforming each h-clique in G into a new *clique vertex* that connects to all vertices belonging to that clique. Formally,

$$V_C = \{ v_C : C \in \Psi_h(G) \}, \qquad E' = \bigcup_{C \in \Psi_h(G)} \{ (u, v_C) : u \in C \}.$$

Here, V_C represents the set of all clique vertices, and E' denotes the bipartite edges linking each clique vertex v_C to the member vertices of its corresponding clique C. This transformation results in an augmented bipartite structure that jointly encodes vertex-level and clique-level relationships within G.

Example 7. *As shown in Figure 4, let h=3, we provide a very simple graph contains two triangles C_1=$\{v_1, v_2, v_3\}$ and C_2=$\{v_1, v_3, v_4\}$. The Figure 4(b) shows its 3-clique augmented graph $\mathcal{G}_3$.*

For each h-clique $C \in \Psi_h(G)$, choose an arbitrary vertex $\alpha_C \in C$. This choice induces an *h-clique orientation graph* $\vec{\mathcal{G}}_h = (V \cup V_C, \vec{E}')$, where each clique C is represented by a clique vertex $v_C \in V_C$. For each clique vertex v_C, we orient the edge between v_C and the chosen vertex α_C as $\langle v_C, \alpha_C \rangle$. For every other vertex $u \in C \setminus \{\alpha_C\}$, we orient the edge between u and v_C as $\langle u, v_C \rangle$. Thus, each h-clique contributes exactly one unit of in-degree to one original vertex, namely the chosen vertex α_C.

We use $\langle u, v \rangle$ to denote a directed edge and (u, v) to denote an undirected edge; reversing $\langle u, v \rangle$ gives $\langle v, u \rangle$. Since $\vec{\mathcal{G}}_h$ is bipartite between original vertices and clique vertices, any directed path alternates between these two types of vertices, capturing reachability through shared h-cliques.

Reversible path and balanced orientation. We define two key concepts that characterize the orientation structure.

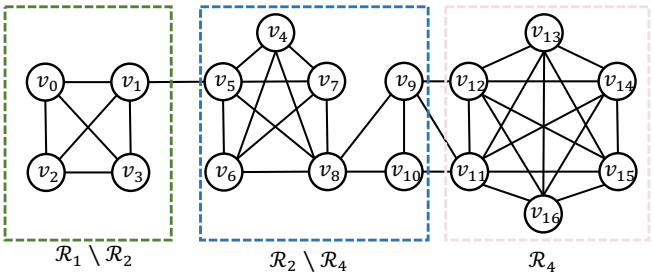


Figure 5: An example to illustrate our graph partition.

Definition 5.1 (Reversible path). Given $\vec{\mathcal{G}}_h$, a path $s \to t$ is called a *reversible path* if

$$d(t, \vec{\mathcal{G}}_h) - d(s, \vec{\mathcal{G}}_h) \geq 2,$$

where $d(u, \vec{\mathcal{G}}_h)$ denotes the in-degree of vertex u.

Definition 5.2 (h-clique balanced orientation). An orientation graph $\vec{\mathcal{G}}$ is called a *balanced orientation* if it contains no reversible path.

The balanced orientation offsets the direction of clique-induced edges, ensuring that no vertex can increase its in-degree through path reversal. This equilibrium is essential for forming a stable, well-defined partition of the graph.

h-Clique density-based graph partition. We now introduce the *h-clique density-based graph partition.*

Given an h-clique balanced orientation graph $\vec{\mathcal{G}}_h$, the *h-clique density-based graph partition* aims to split the graph G into a series of small subgraphs of G induced by vertex sets $\Theta_h = \{\mathcal{R}_i\}_{i=0}^{\delta_h}$ of G, where $\mathcal{R}_i$ denotes the i-th clique partition set.

We write $u \rightsquigarrow v$ if u is reachable from v in $\vec{G}_h$. For example, as illustrated in Figure 4, we can see $v_1 \rightsquigarrow v_3$. With this notation, $\mathcal{R}_i$ can be characterized as

$$\mathcal{R}_i = \left\{ u \in V \mid d(u, \vec{G}_h) \geq i \text{ or } \exists v \in V \text{ such that } u \rightsquigarrow v \text{ and } d(v, \vec{G}_h) \geq i \right\}.$$

The resulting clique partition sets exhibit progressively increasing h-clique densities; for any two integers i and j with $i < j$, the subgraph $G[\mathcal{R}_j]$ is denser than $G[\mathcal{R}_i]$. This characterization is closely related to the notion of pseudoarboricity. In particular, δ_h coincides with the pseudoarboricity of the augmented graph $\mathcal{G}_h$. Recall that the pseudoarboricity of a graph is the minimum number of pseudoforests into which its edge set can be partitioned. Equivalently, it can be characterized by an orientation with bounded out-degree:

$$\delta_h = \min \left\{ i \;\middle|\; E(\mathcal{G}_h) \text{ can be oriented such that } d(u, \vec{\mathcal{G}}_h) \leq i,\ \forall u \in V(\mathcal{G}_h) \right\}.$$

Since for all $i \in [1, \delta_h - 1]$, $\mathcal{R}_{i+1} \subseteq \mathcal{R}_i$, the difference $\mathcal{R}_i \setminus \mathcal{R}_{i+1}$ forms a disjoint vertex set. Thus, the corresponding induced subgraphs collectively constitute a valid graph partition.

Example 8. *Let us consider the graph shown in Figure 2. The corresponding partitioned graph is illustrated in Figure 5. As observed, $\mathcal{R}_i \setminus \mathcal{R}_{i+1}$ for $i \in 1, 2, 3$, forming the partition. It is worth noting that $\mathcal{R}_3 = \mathcal{R}_4$, thereby we omit $\mathcal{R}_3$ in the figure.*

Theoretical properties. We next establish the relationship between partition subgraphs and locally h-clique dense subgraphs.

Lemma 5.3. *Given a partition set $\mathcal{R}_i$, for any non-empty sets $S \subseteq \mathcal{R}_i$ and $T \subseteq V \setminus \mathcal{R}_i$, the following inequality holds:*

$$\rho_h(S, \mathcal{R}_i \setminus S) > i - 1 \geq \rho_h(T, \mathcal{R}_i). \quad (2)$$

Proof. Specifically, the number of cross-h-cliques between S and $\mathcal{R}_i \setminus S$ exceeds $(i-1)\cdot|S|$, while the number between T and $\mathcal{R}_i$ is at most $(i-1)\cdot|T|$. By the definition of outer density (definition 4.5), the inequality holds. □

Theorem 5.4. *Every clique partition set $\mathcal{R}_i$ is a locally h-clique dense subgraph.*

Proof. By lemma 5.3, for any $S \subseteq \mathcal{R}_i$ and $T \subseteq V \setminus \mathcal{R}_i$, it holds that $\rho_h(S, \mathcal{R}_i \setminus S) > \rho_h(T, \mathcal{R}_i)$. Hence, the condition of the local h-clique dense subgraph (definition 4.6) is satisfied. □

We now present the algorithm for computing the h-clique density-based graph partition for a given graph G.

Definition 5.5 (h-clique orientation flow network). Given an h-clique orientation graph $\vec{\mathcal{G}}_h$, a vertex set $X \subseteq V$, and an integer threshold d, let $V_C(X) = \{v_C \in V_C \mid C \subseteq X\}$. The *$h$-clique orientation flow network* on X, denoted by $\mathcal{F}_h^d(X) = (X \cup V_C(X) \cup \{s,t\}, \mathcal{E}, c(\cdot))$, is a capacitated directed graph constructed as follows.

(1) **Internal edges.** For each directed edge $\langle u, v\rangle$ in $\vec{\mathcal{G}}_h[X \cup V_C(X)]$, include $\langle u, v\rangle$ in $\mathcal{E}$ with capacity $c(\langle u, v\rangle) = 1$.
(2) **Source edges.** For each vertex $u \in X$ with $d(u, \vec{\mathcal{G}}_h) < d$, add an edge $\langle s, u\rangle$ to $\mathcal{E}$ with capacity $c(\langle s, u\rangle) = d - d(u, \vec{\mathcal{G}}_h)$.
(3) **Sink edges.** For each vertex $u \in X$ with $d(u, \vec{\mathcal{G}}_h) > d$, add an edge $\langle u, t\rangle$ to $\mathcal{E}$ with capacity $c(\langle u, t\rangle) = d(u, \vec{\mathcal{G}}_h) - d$.

Based on this flow network, we can obtain the h-clique density-based graph partition. At first glance, our problem may appear related to the integer densest subgraph decomposition problem studied in [80]. However, the two problems have fundamentally different objectives. The goal of [80] is to decompose a graph into layers according to edge-density levels, whereas our goal is to construct an h-clique density-based partition to reduce the search space of our computation. Nevertheless, their layer-based decomposition framework provides useful algorithmic insight for our setting. Building on this insight, and extending it from edge density to h-clique density, we design a novel divide-and-conquer algorithm, called `Orientation-par`, to compute the h-clique density-based graph partition Θ_h. The full procedure is provided in our technical report [86].

Theorem 5.6. *Given an h-clique orientation graph $\vec{\mathcal{G}}_h$, the procedure* `Orientation-par` *correctly outputs the h-clique density-based partition Θ_h.*

Complexity analysis. Notably, in `Orientation-Par`, the procedure invokes `Divide-Par` at most δ_h times. In each invocation, the h-clique orientation network reverses all such augmenting paths simultaneously via a single maximum-flow computation. We apply Dinic's algorithm [10] to solve the flow problem; the running time of each maximum-flow computation is $O(|\mathcal{E}|^{3/2})$. Therefore, the overall time complexity is $O(|\mathcal{E}|^{3/2} \cdot \delta_h)$, where the number of edges satisfies $|\mathcal{E}| = n + |\Psi_h|h$, and $\delta_h \ll n + |\Psi_h|$.

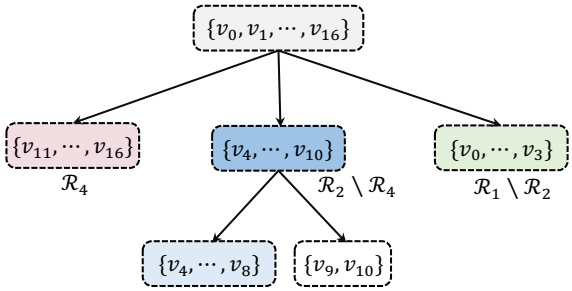


Figure 6: An example to illustrate the `DCLDS` algorithm.

5.2 Our `DCLDS` method

In this section, we are ready to present our `DCLDS` method, which builds upon the clique orientation-based graph partitioning and our divide-and-conquer-based algorithm.

Algorithm 3: `DCLDS`

```
input : A graph G, two integers k and h
output: Top-k LhCDSes in G
1  L ← ∅, D ← ∅;
2  G⃗_h ← compute h-clique orientation graph of G;
3  Θ_h ← Orientation-Par(G, G⃗_h);
4  foreach block P ∈ Θ_h in non-increasing density order do
5      C ← DeriveLhCDS(G[P], k − |L|, h);
6      foreach H ∈ C do
7          if ∄(u, v) ∈ E(G) such that u ∈ V(H) and v ∈ D then
8              L ← L ∪ {H};
9              if |L| ≥ k then break;
10     D ← D ∪ P;
11     if |L| ≥ k then break;
12 return L;
```

As shown in Algorithm 3, it first invokes the `Orientation-Par` to split the entire graph into a series of small subgraphs (lines 2-3). Afterwards, it repeatedly calls `DeriveLhCDS` to identify LhCDSes until the top-k results are found (lines 4–6).

Example 9. *As shown in Figure 6, we provide an example to illustrate the running process of* `DCLDS`. *Compared to the recursion tree in Figure 3(b), the recursion depth is significantly reduced. This is because, after graph partitioning,* `DCLDS` *identifies LhCDSes within smaller subgraphs rather than the entire graph, thereby substantially reducing the search space.*

Theorem 5.7. *Given a graph G, and two parameters k and h,* `DCLDS` *takes $(n + \Psi_h)^2 \cdot (n + \Psi_h h)$ time to identify top-k LhCDSes.*

Extension to support motif-based patterns. `DCLDS` can be naturally extended to support the locally pattern densest subgraph problem [74], which generalizes higher-order densest subgraph models, since cliques are special cases of general graph patterns. Following similar design principles to those in [27, 74], our framework can be adapted to handle arbitrary motif-based patterns with only minor modifications to the pattern enumeration component, yielding a new method `DCLDS-Pat`. Detailed algorithmic techniques for such extensions can be found in [27, 74].

Table 4: Datasets used in our experiments.

Dataset	Category	$\|V\|$	$\|E\|$
CA-GrQc (CA)	Collaboration	5,242	14,484
web-webbase-2001 (WW)	Web	16,062	25,593
fb-pages-company (FB)	Social	14,113	52,126
CA-CondMat (CC)	Collaboration	23,133	93,439
soc-epinions (SE)	Social	26,588	100,120
Email (EA)	Communication	36,692	183,831
Amazon (AM)	Purchasing	334,863	925,872
loc-gowalla (LG)	Location	196,591	950,327
DBLP (DP)	Collaboration	317,080	1,049,866
web-Stanford (WS)	Web	281,904	1,992,636
youtube (YT)	Friendship	1,134,890	2,987,624
web-Google (WG)	Web	875,713	5,105,039
Patent (PT)	Hyperlink	3,033,374	16,518,947
soc-pokec (SP)	Social	1,632,803	22,301,964
Zhishi-baidu (ZB)	Hyperlink	7,827,193	62,246,014

6 EXPERIMENTS

We now present the experimental results. Section 6.1 discusses the setup. We discuss the results in Sections 6.2 and 6.3.

6.1 Setup

Datasets. We use 15 real-world datasets from different domains, which are downloaded from the Stanford Network Analysis Platform [1], Laboratory of Web Algorithmics [2], Network Repository [3], and Konect [4]. Their detailed descriptions can also be found on these websites. Table 4 reports the statistics of these graphs.

Competitors. We evaluate the following algorithms for the LhCDS problem:

- `LTDS` [64]: the maximum flow-based algorithm, which is recapped in Section 3.
- `LTDS-Core`: an optimized variant of `LTDS` that incorporates the graph reduction technique proposed in [27].
- `IPPV` [74]: the state-of-the-art LhCDS discovery algorithm, which is discussed in Section 3.
- `DCLDS`: our proposed algorithm in Section 5.2.

Note that the original implementation of `LTDS` supports only the case $h = 3$. To enable a fair comparison, we extended it to support arbitrary values of h, whereas both `IPPV` and our method `DCLDS` are natively applicable to any h. We implement all the algorithms in C++ and run experiments on a machine having an Intel(R) Xeon(R) Gold 6338R 2.0GHz CPU and 512GB of memory, with Ubuntu installed. If an algorithm cannot finish in two weeks, we mark its running time as **INF** in the Figure and "—" in the Table. In our experiments, we have already included the time cost of enumerating h-cliques in all results, which means that our algorithm is purely online without offline preprocessing.

[1] http://snap.stanford.edu/data/
[2] http://law.di.unimi.it/datasets.php
[3] https://networkrepository.com/network-data.php
[4] http://konect.cc/networks/

Table 5: Memory usage of `IPPV` and `DCLDS`.

Dataset	Memory usage (MB)		Reduction ($\frac{\texttt{IPPV}}{\texttt{DCLDS}}$)
	`IPPV`	`DCLDS`	
Amazon	941.92	251.81	3.74×
loc-gowalla	2,308.39	958.45	2.40×
web-Stanford	10,144.42	3,273.50	3.09×
web-Google	11,169.66	3,986.41	2.80×
Zhishi-baidu	≥2,097,152	123,950.51	≥16.91×

6.2 Comparison with existing LhCDS algorithms

In this section, we extensively compare `DCLDS` with `LTDS` and `IPPV` by various experiments.

1. Effect of k. Figure 7 compares the average running time of these four algorithms on 15 datasets by varying the size of k, where k=5∼20 and h=3. Clearly, `DCLDS` achieves up to two orders of magnitude speedup over all three competitors. This remarkable efficiency stems from its lower theoretical time complexity and its ability to discover LhCDSes within the smaller subgraphs obtained through the clique orientation–based partitioning. For example, on the DBLP dataset, `DCLDS` completes the computation within 10 seconds, whereas the baselines require more than 1,000 seconds. Furthermore, as the parameter k increases, both `IPPV` and `LTDS-Core` exhibit steadily rising runtime, while our method maintains stable performance, consistent with the analysis in Section 1. Interestingly, we also observe that `LTDS-Core` outperforms `IPPV` and `LTDS` on most datasets, which implies that the graph reduction technique is very useful in LhCDS problem.

2. Effect of h. We evaluate the effect of h using four datasets from different domains, and the values of h are set to 4 and 5, respectively. The experimental results are reported in Figure 8, which clearly shows that `DCLDS` outperforms the other algorithms on all datasets. Particularly, on the Amazon dataset with $h = 4$, our method is up to three orders of magnitude faster than `IPPV` and two orders of magnitude faster than `LTDS-Core`. We observe that in some cases, the improvement of our method is less pronounced, for example, on the FB dataset with $h = 5$. This is because, in such settings, the clique enumeration step dominates the total runtime.

3. Memory costs. We examine the memory usage of `IPPV` and `DCLDS` on the five datasets, reporting their results in Table 5. Although both `IPPV` and `DCLDS` share the same space complexity of $O(n + \Psi_h)$, where Ψ_h denotes the number of h-cliques in the graph, `DCLDS` typically consumes less memory in practice thanks to its divide-and-conquer strategy, which processes smaller subgraphs sequentially. This is an optimization technique at the practice level.

4. Effect of patterns. We evaluate `IPPV-Pat` and `DCLDS-Pat` on four datasets using four widely adopted patterns (see our technical report [86]). The results are shown in Figure 9, we can see that our method `DCLDS-Pat` consistently outperforms `IPPV-Pat` in runtime.

6.3 Detailed analysis of `DCLDS`

We perform an in-depth evaluation and analysis of `DCLDS`.

1. Scalability test. In this experiment, we evaluate the scalability of the `IPPV` and `DCLDS`. Specifically, for each graph, we first

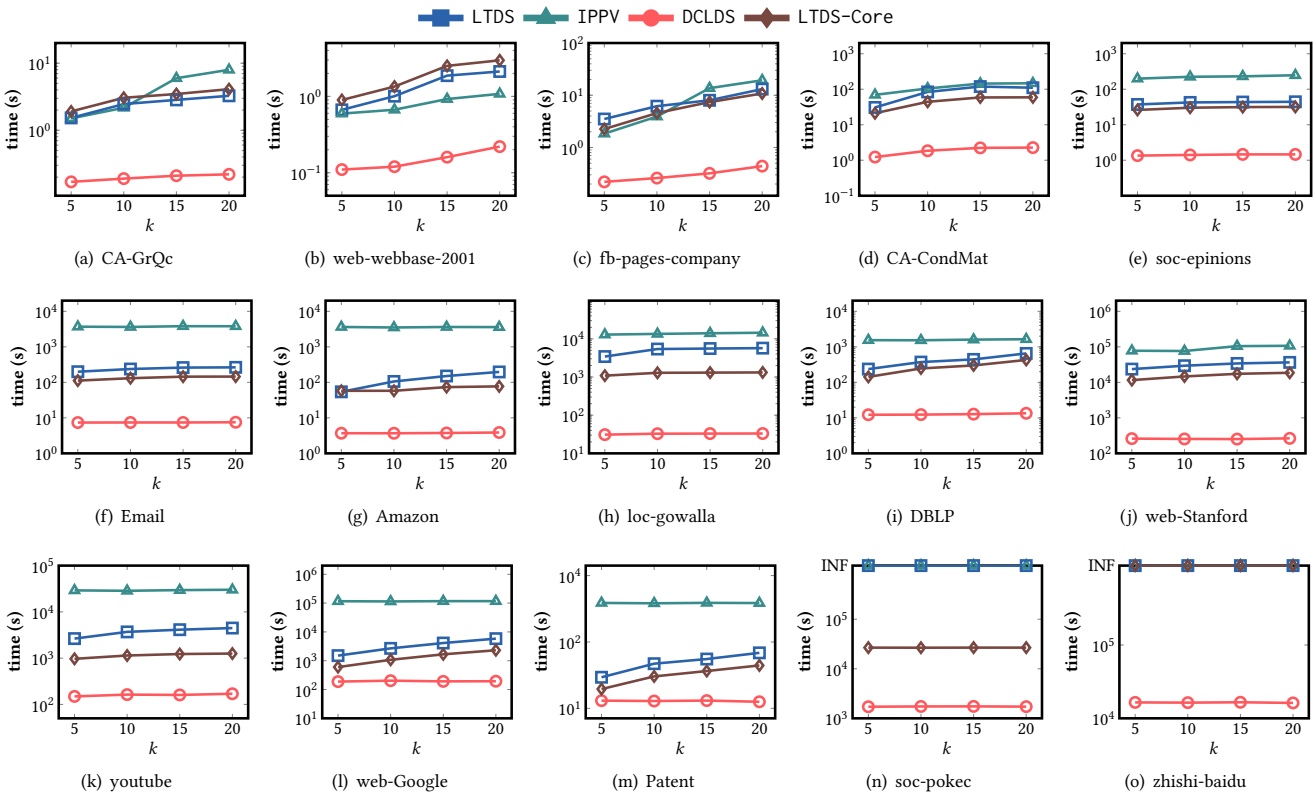


Figure 7: Effect of k on the efficiency of DCLDS, IPPV, LTDS, and LTDS-Core.

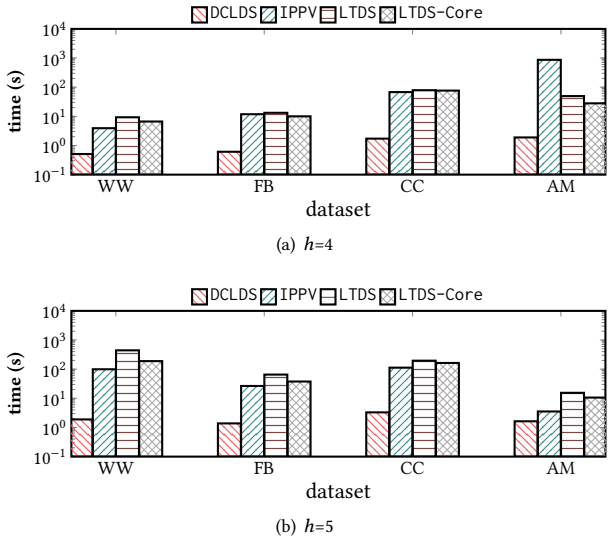


Figure 8: Effect of parameter h on the efficiency of LhCDS discovery algorithms.

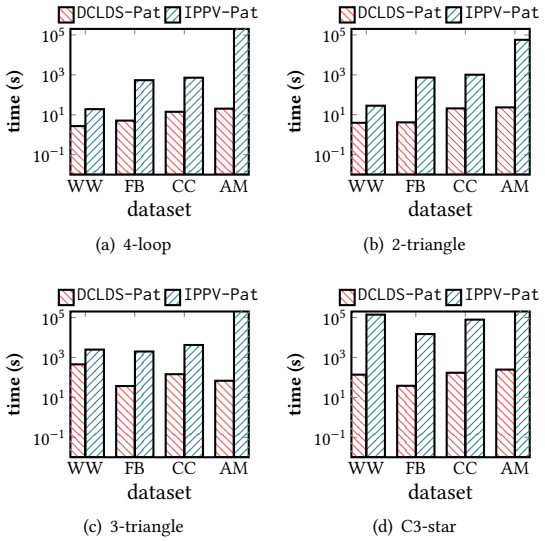
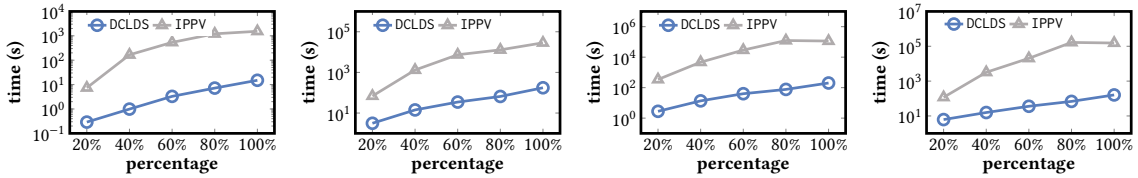


Figure 9: Effect of patterns on the efficiency of algorithms.

randomly select 20%, 40%, 60%, 80%, and 100% of its vertices and then obtain five sub-graphs induced by these vertices, respectively. Afterwards, we run IPPV and DCLDS on these sub-graphs, and report both the average efficiency results for each sub-graph. The results are shown in Figure 10. We observe that DCLDS demonstrates significantly better scalability than IPPV, as its runtime increases much more slowly with graph size.

2. Ablation study of DCLDS. In this experiment, we evaluate the effect of two key optimization techniques: our divide-and-conquer strategy and the clique orientation-based graph partition. We develop four different algorithms to compare the variants between



(a) DBLP (b) youtube (c) web-google (d) patent

Figure 10: Scalability test for DCLDS algorithm.

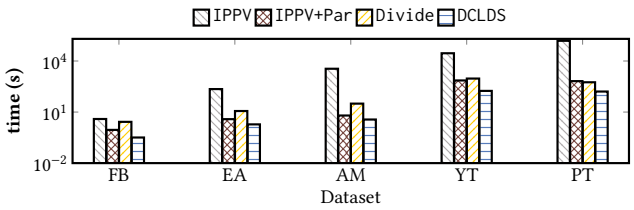


Figure 11: Ablation experiments.

IPPV and DCLDS. The detailed descriptions of the different variants are shown in our technical report [86]. We then run these four algorithms on five datasets with h=3 and k=10, report the running time in Figure 11. We make the following observations and analysis: Our DCLDS significantly outperforms all three variants, demonstrating the effectiveness of its two core components. The divide-and-conquer strategy notably reduces the overall time complexity by diminishing the item, N_{flow}, compared to IPPV, which explains why Divide runs much faster than IPPV. In addition, the clique orientation–based graph partitioning effectively reduces the search space by decomposing the original graph into smaller, independent subgraphs. As a result, IPPV+Par achieves up to two orders of magnitude speedup over IPPV. When both components are integrated, DCLDS attains the best performance, combining efficiency from divide-and-conquer recursion with scalability from partitioned computation.

7 RELATED WORKS

This section reviews the existing works on the densest subgraph problem and its variants. The dense subgraph decomposition problem is also discussed.

Densest subgraph problem and its variants. The densest subgraph discovery (DSD) problem seeks to identify the subgraph that maximizes the average degree of vertices [2, 4–7, 9, 12, 13, 24, 34, 37, 46, 65, 70, 85]. A classical approach to solving DSD transforms it into a parametric maximum-flow formulation [34], which enables performing a binary search over possible density values and uses the flow network to verify whether a given density threshold is achievable. To capture higher-order structural patterns beyond edge connectivity, DSD has been generalized to the h-clique densest subgraph (CDS) problem, which aims to find subgraphs maximizing h-clique density [25, 27, 38, 51, 56, 72, 84, 85]. This formulation better identifies "near-clique" regions in graphs. Notably, when $h = 2$, CDS degenerates to the classical DSD problem. Beyond the basic formulation, numerous variants of DSD have been proposed to address specialized objectives [3, 20, 28, 53, 60, 65, 75]. The densest-k-subgraph (DkS) problem, which seeks the subgraph with k vertices maximizing the number of edges, is well-known to be NP-hard [28]. To discover local dense regions, Qin et al. [60] introduced the top-k locally densest subgraph problem, and Ma et al. [53] developed a convex relaxation based on density-friendly graph decomposition [21]. In addition, several other extensions have also been explored, including anchored DSD problem [20, 88], and fair/-diverse DSD formulations [1, 57]. The DSD problem is extended to other types of graphs, such as directed graphs [47, 52, 54, 55, 87], uncertain graphs [61], and heterogeneous graphs [15, 58].

Dense subgraph decomposition problem. The dense subgraph decomposition problem seeks to decompose a graph into nested cohesive subgraphs according to different cohesive definitions. Classical approaches include the k-core [8, 66], k-truss [18, 63, 81], and k-ECC [40, 79]. To unify these paradigms, nucleus decomposition [67] generalizes them under a clique-based framework, organizing all (r, s)-nucleus into a containment hierarchy that reveals overlapping dense communities and their relationships. This nucleus perspective captures both k-core (r=1, s=2) and k-truss (r=2, s=3) as special cases. Besides, density-based decomposition [21, 70] partitions a graph into subgraphs according to their density, progressively decomposing it from regions of higher density to lower density. Subsequent work has extended dense subgraph decompositions to directed [26, 30, 50], temporal [78], bipartite [22, 48, 89], uncertain [11, 41], and heterogeneous [36] graphs.

8 CONCLUSIONS

In this paper, we investigate the problem of efficient locally h-clique densest subgraph (LhCDS) discovery. The existing LhCDS algorithms suffer from high time complexity and redundant computations, making them inefficient on large graphs. To address this limitation, we propose a novel *divide-and-conquer* algorithm to break the time complexity bottleneck of the existing methods. Furthermore, we design a *clique orientation-based graph partitioning* technique that divides the original graph into multiple smaller subgraphs, enabling our algorithm to efficiently identify all LhCDSes within these subgraphs rather than over the entire graph. Our experimental results on 15 real-world large graphs show that our proposed algorithm is efficient for the LhCDS problem and achieves up to two orders of magnitude faster than the state-of-the-art algorithm. In the future, we will design distributed algorithms for the LhCDS problem to handle extremely large graphs.

9 ACKNOWLEDGMENTS

This work was supported by the 1+1+1 CUHK-CUHK(SZ)-GDSTC Joint Collaboration Fund under Grant 2025A0505000045 and Grant 2025A0505000079.

REFERENCES

[1] Aris Anagnostopoulos, Luca Becchetti, Adriano Fazzone, Cristina Menghini, and Chris Schwiegelshohn. 2020. Spectral relaxations and fair densest subgraphs. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 35–44.

[2] Venkat Anantharam and Justin Salez. 2016. The densest subgraph problem in sparse random graphs. (2016).

[3] Reid Andersen and Kumar Chellapilla. 2009. Finding dense subgraphs with size bounds. In *International workshop on algorithms and models for the web-graph*. Springer, 25–37.

[4] Albert Angel, Nick Koudas, Nikos Sarkas, Divesh Srivastava, Michael Svendsen, and Srikanta Tirthapura. 2014. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *The VLDB journal* 23 (2014), 175–199.

[5] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. 2002. Complexity of finding dense subgraphs. *Discrete Applied Mathematics* 121, 1-3 (2002), 15–26.

[6] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Densest subgraph in streaming and mapreduce. *arXiv preprint arXiv:1201.6567* (2012).

[7] Oana Denisa Balalau, Francesco Bonchi, TH Hubert Chan, Francesco Gullo, and Mauro Sozio. 2015. Finding subgraphs with maximum total density and limited overlap. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 379–388.

[8] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[9] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos Tsourakakis. 2015. Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 173–182.

[10] Markus Blumenstock. 2016. Fast algorithms for pseudoarboricity. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 113–126.

[11] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. 2014. Core decomposition of uncertain graphs. In *KDD*. ACM, 1316–1325.

[12] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. 2020. Flowless: Extracting densest subgraphs without flow computations. In *Proceedings of The Web Conference 2020*. 573–583.

[13] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *International workshop on approximation algorithms for combinatorial optimization*. Springer, 84–95.

[14] Jie Chen and Yousef Saad. 2010. Dense subgraph extraction with application to community detection. *IEEE Transactions on knowledge and data engineering* 24, 7 (2010), 1216–1230.

[15] Lu Chen, Chengfei Liu, Rui Zhou, Kewen Liao, Jiajie Xu, and Jianxin Li. 2023. Densest Multipartite Subgraph Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 17, 4 (2023), 699–711.

[16] Tianyi Chen and Charalampos Tsourakakis. 2022. Antibenford subgraphs: Unsupervised anomaly detection in financial networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2762–2770.

[17] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. 2003. Reachability and distance queries via 2-hop labels. *SIAM J. Comput.* 32, 5 (2003), 1338–1355.

[18] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16, 3.1 (2008).

[19] Guangyu Cui, Yu Chen, De-Shuang Huang, and Kyungsook Han. 2008. An algorithm for finding functional modules and protein complexes in protein-protein interaction networks. *Journal of Biomedicine and Biotechnology* 2008 (2008).

[20] Yizhou Dai, Miao Qiao, and Lijun Chang. 2022. Anchored densest subgraph. In *Proceedings of the 2022 International Conference on Management of Data*. 1200–1213.

[21] Maximilien Danisch, T-H Hubert Chan, and Mauro Sozio. 2017. Large scale density-friendly graph decomposition via convex programming. In *Proceedings of the 26th International Conference on World Wide Web*. 233–242.

[22] Danhao Ding, Hui Li, Zhipeng Huang, and Nikos Mamoulis. 2017. Efficient fault-tolerant group recommendation using alpha-beta-core. In *CIKM*. 2047–2050.

[23] Xiaoxi Du, Ruoming Jin, Liang Ding, Victor E Lee, and John H Thornton Jr. 2009. Migration motif: a spatial-temporal pattern mining approach for financial markets. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[24] Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. 2015. Efficient densest subgraph computation in evolving graphs. In *Proceedings of the 24th international conference on world wide web*. 300–310.

[25] Yixiang Fang, Wensheng Luo, and Chenhao Ma. 2022. Densest subgraph discovery on large graphs: Applications, challenges, and techniques. *Proceedings of the VLDB Endowment* 15, 12 (2022), 3766–3769.

[26] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2018. Effective and efficient community search over large directed graphs. *TKDE* 31, 11 (2018), 2093–2107.

[27] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks VS Lakshmanan, and Xuemin Lin. 2019. Efficient algorithms for densest subgraph discovery. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1719–1732.

[28] Uriel Feige, Michael Seltser, et al. 1997. *On the densest k-subgraph problem*. Citeseer.

[29] Eugene Fratkin, Brian T Naughton, Douglas L Brutlag, and Serafim Batzoglou. 2006. MotifCut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics* 22, 14 (2006), e150–e157.

[30] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. 2013. D-cores: measuring collaboration of directed graphs based on degeneracy. *KAIS* 35, 2 (2013), 311–343.

[31] David Gibson, Ravi Kumar, and Andrew Tomkins. 2005. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st international conference on Very large data bases*. 721–732.

[32] Aristides Gionis, Flavio PP Junqueira, Vincent Leroy, Marco Serafini, and Ingmar Weber. 2013. Piggybacking on social networks. In *VLDB 2013-39th International Conference on Very Large Databases*, Vol. 6. 409–420.

[33] Aristides Gionis and Charalampos E Tsourakakis. 2015. Dense subgraph discovery: Kdd 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2313–2314.

[34] Andrew V Goldberg. 1984. Finding a maximum density subgraph. (1984).

[35] Donald Goldfarb and Shucheng Liu. 1990. An O (n 3 L) primal interior point algorithm for convex quadratic programming. *Mathematical programming* 49, 1 (1990), 325–340.

[36] Yucan Guo, Chenhao Ma, and Yixiang Fang. 2024. Efficient Core Decomposition Over Large Heterogeneous Information Networks. In *ICDE*. IEEE, 2393–2406.

[37] Elfarouk Harb, Kent Quanrud, and Chandra Chekuri. 2022. Faster and scalable algorithms for densest subgraph and decomposition. *Advances in Neural Information Processing Systems* 35 (2022), 26966–26979.

[38] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2023. Scaling Up k-Clique Densest Subgraph Detection. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[39] Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. 2005. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics* 21, suppl_1 (2005), i213–i221.

[40] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2016. Querying minimal steiner maximum-connected subgraphs in large graphs. In *CIKM*. 1241–1250.

[41] Xin Huang, Wei Lu, and Laks V. S. Lakshmanan. 2016. Truss Decomposition of Probabilistic Graphs: Semantics and Algorithms. In *SIGMOD Conference*. ACM, 77–90.

[42] Martin Jaggi. 2013. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*. PMLR, 427–435.

[43] Shweta Jain and C Seshadhri. 2020. Provably and efficiently approximating near-cliques using the Turán shadow: PEANUTS. In *Proceedings of The Web Conference 2020*. 1966–1976.

[44] Ruoming Jin, Yang Xiang, Ning Ruan, and David Fuhry. 2009. 3-hop: a high-compression indexing scheme for reachability query. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 813–826.

[45] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. 2023. A survey on the densest subgraph problem and its variants. *arXiv preprint arXiv:2303.14467* (2023).

[46] 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.

[47] Luocheng Liang, Yingli Zhou, and Yixiang Fang. 2026. Accelerated Coordinate Descent for Directed Densest Subgraph Discovery. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 783–794.

[48] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2020. Efficient (α, β)-core computation in bipartite graphs. *The VLDB Journal* 29, 5 (2020), 1075–1099.

[49] Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H Eugene Stanley. 2016. The H-index of a network node and its relation to degree and coreness. *Nature communications* 7, 1 (2016), 10168.

[50] Wensheng Luo, Yixiang Fang, Chunxu Lin, and Yingli Zhou. 2024. Efficient Parallel D-Core Decomposition at Scale. *Proceedings of the VLDB Endowment* 17, 10 (2024), 2654–2667.

[51] Wensheng Luo, Chenhao Ma, Yixiang Fang, and Laks VS Lakshman. 2023. A Survey of Densest Subgraph Discovery on Large Graphs. *arXiv preprint arXiv:2306.07927* (2023).

[52] Wensheng Luo, Zhuo Tang, Yixiang Fang, Chenhao Ma, and Xu Zhou. 2023. Scalable algorithms for densest subgraph discovery. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 287–300.

[53] Chenhao Ma, Reynold Cheng, Laks VS Lakshmanan, and Xiaolin Han. 2022. Finding locally densest subgraphs: a convex programming approach. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2719–2732.

[54] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, and Xiaolin Han. 2022. A convex-programming approach for efficient directed densest

subgraph discovery. In *Proceedings of the 2022 International Conference on Management of Data*. 845–859.
[55] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2020. Efficient algorithms for densest subgraph discovery on large directed graphs. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1051–1066.
[56] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. 2015. Scalable large near-clique detection in large-scale networks via sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 815–824.
[57] Atsushi Miyauchi, Tianyi Chen, Konstantinos Sotiropoulos, and Charalampos E Tsourakakis. 2023. Densest Diverse Subgraphs: How to Plan a Successful Cocktail Party with Diversity. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1710–1721.
[58] Yudong Niu, Yuchen Li, Jiaxin Jiang, and Laks V. S. Lakshmanan. 2025. SANS: Efficient Densest Subgraph Discovery over Relational Graphs without Materialization. In *WWW*. ACM, 4250–4263.
[59] James B Orlin. 2013. Max flows in O(nm) time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 765–774.
[60] Lu Qin, Rong-Hua Li, Lijun Chang, and Chengqi Zhang. 2015. Locally densest subgraph discovery. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 965–974.
[61] Arkaprava Saha, Xiangyu Ke, Arijit Khan, and Cheng Long. 2023. Most Probable Densest Subgraphs. In *ICDE*. IEEE, 1447–1460.
[62] Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. 2010. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Research in Computational Molecular Biology: 14th Annual International Conference, RECOMB 2010, Lisbon, Portugal, April 25-28, 2010. Proceedings 14*. Springer, 456–472.
[63] Kazumi Saito, Takeshi Yamada, and Kazuhiro Kazama. 2008. Extracting communities from complex networks by the k-dense method. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 91, 11 (2008), 3304–3311.
[64] Raman Samusevich, Maximilien Danisch, and Mauro Sozio. 2016. Local triangle-densest subgraphs. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 33–40.
[65] Saurabh Sawlani and Junxing Wang. 2020. Near-optimal fully dynamic densest subgraph. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 181–193.
[66] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.
[67] Jessica Shi, Laxman Dhulipala, and Julian Shun. 2023. Theoretically and practically efficient parallel nucleus decomposition. In *Proceedings of the 2023 ACM Workshop on Highlights of Parallel Computing*. 7–8.
[68] Bintao Sun, Maximilien Danisch, TH Hubert Chan, and Mauro Sozio. 2020. Kclist++: A simple algorithm for finding k-clique densest subgraphs in large graphs. *Proceedings of the VLDB Endowment (PVLDB)* (2020).
[69] Brian K Tanner, Gary Warner, Henry Stern, and Scott Olechowski. 2010. Koobface: The evolution of the social botnet. In *2010 eCrime Researchers Summit*. IEEE, 1–10.
[70] Nikolaj Tatti and Aristides Gionis. 2015. Density-friendly graph decomposition. In *Proceedings of the 24th International Conference on World Wide Web*. 1089–1099.
[71] Tran Ba Trung, Lijun Chang, Nguyen Tien Long, Kai Yao, and Huynh Thi Thanh Binh. 2023. Verification-Free Approaches to Efficient Locally Densest Subgraph Discovery. In *ICDE*. IEEE, 1–13.
[72] Charalampos Tsourakakis. 2015. The k-clique densest subgraph problem. In *Proceedings of the 24th international conference on world wide web*. 1122–1132.
[73] 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 *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 104–112.
[74] Xiaojia Xu, Haoyu Liu, Xiaowei Lv, Yongcai Wang, and Deying Li. 2024. An efficient and exact algorithm for locally h-clique densest subgraph discovery. *Proceedings of the ACM on Management of Data* 2, 6 (2024), 1–26.
[75] Yichen Xu, Chenhao Ma, Yixiang Fang, and Zhifeng Bao. 2023. Efficient and Effective Algorithms for Generalized Densest Subgraph Discovery. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.
[76] Yi Yang, Chenhao Ma, Reynold Cheng, Laks VS Lakshmanan, and Xiaolin Han. 2025. Finding Locally Densest Subgraphs: Convex Programming with Edge and Triangle Density. *arXiv preprint arXiv:2504.10937* (2025).
[77] Kaiqiang Yu and Cheng Long. 2021. Graph Mining Meets Fake News Detection. In *Data Science for Fake News: Surveys and Perspectives*. Springer, 169–189.
[78] Michael Yu, Dong Wen, Lu Qin, Ying Zhang, Wenjie Zhang, and Xuemin Lin. 2021. On querying historical k-cores. *Proceedings of the VLDB Endowment* (2021).
[79] Long Yuan, Lu Qin, Xuemin Lin, Lijun Chang, and Wenjie Zhang. 2017. I/O efficient ECC graph decomposition via graph reduction. *The VLDB Journal* 26, 2 (2017), 275–300.
[80] Yalong Zhang, Rong-Hua Li, Qi Zhang, Hongchao Qin, and Guoren Wang. 2024. Efficient algorithms for density decomposition on large static and dynamic graphs. *Proceedings of the VLDB Endowment* 17, 11 (2024), 2933–2945.
[81] Yang Zhang and Srinivasan Parthasarathy. 2012. Extracting analyzing and visualizing triangle k-core motifs within networks. In *ICDE*. IEEE, 1049–1060.
[82] Feng Zhao and Anthony KH Tung. 2012. Large scale cohesive subgraphs discovery for social network visual analysis. *Proceedings of the VLDB Endowment* 6, 2 (2012), 85–96.
[83] Yingli Zhou, Qingshuo Guo, and Yixiang Fang. 2025. Efficient k-Clique Densest Subgraph Discovery: Towards Bridging Practice and Theory. *Proceedings of the VLDB Endowment* 18, 10 (2025), 3490–3503.
[84] Yingli Zhou, Qingshuo Guo, Yixiang Fang, and Chenhao Ma. 2024. A Counting-based Approach for Efficient k-Clique Densest Subgraph Discovery. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.
[85] Yingli Zhou, Qingshuo Guo, Yi Yang, Yixiang Fang, Chenhao Ma, and Laks Lakshmanan. 2024. In-depth Analysis of Densest Subgraph Discovery in a Unified Framework. *arXiv preprint arXiv:2406.04738* (2024).
[86] Yingli Zhou, Taohua Huang, and Yixiang Fang. 2025. Efficient Locally h-Clique Densest Subgraph Discovery via Divide-and-Conquer (technical report). https://github.com/JayLZhou/TechnicalReport/blob/main/VLDB2026_LCDS_appendix.pdf.
[87] Yingli Zhou, Luocheng Liang, and Yixiang Fang. 2025. Efficient and Scalable Directed Densest Subgraph Discovery. *Proc. ACM Manag. Data* 3, 6 (2025), 1–27.
[88] Yingli Zhou, Youran Sun, and Yixiang Fang. 2026. Efficient Anchored Densest Subgraph Discovery: Improved Time Complexity and Practical Performance. *Proceedings of the ACM on Management of Data* 4, 3 (SIGMOD (2026), 1–27.
[89] Zhaonian Zou. 2016. Bitruss decomposition of bipartite graphs. In *DASFAA*. Springer, 218–233.