Colorful Star Motif Counting: Concepts, Algorithms and Applications

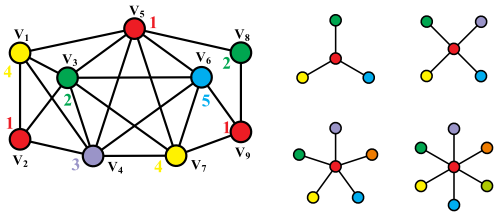
Hongchao Qin, Gao Sen, Rong-Hua Li, Hongzhi Chen, Ye Yuan, and Guoren Wang

***Abstract*—A colorful star motif is a star-shaped graph where any two nodes have different colors. Counting the colorful star motif can help to analyze the structural properties of real-life colorful graphs, model higher-order clustering, and accelerate the mining of the densest subgraph exhibiting h-clique characteristics in graphs. In this manuscript, we introduce the concept of colorful h-star in a colored graph and proposes two higher-order cohesive subgraph models, namely colorful h-star core and colorful h-star truss. We show that the colorful h-stars can be counted and updated very efficiently using a novel dynamic programming (DP) algorithm. Based on the proposed DP algorithm, we develop a colorful h-star core decomposition algorithm which takes $O(hm)$ time, $O(hn+m)$ space; and a colorful h-star truss decomposition algorithm which takes $O(hm^{1.5})$ time, $O(hm)$ space, where m and n denote the number of edges and nodes of the graph respectively. Moreover, we also propose a graph reduction technique based on our colorful h-star core model to accelerate the computation of the approximation algorithm for h-clique densest subgraph mining. The results of comprehensive experiments on 11 large real-world datasets demonstrate the efficiency, scalability and effectiveness of the proposed algorithms.**

***Index Terms*—Colorful star, K-core, K-truss, H-clique densest subgraph.**

I. Introduction

MOTIFS, also known as graphlets, patterns, or subgraphs, play a crucial role as the foundational elements in networks [1]. The analysis of motifs has proven ingaining insights into network evolution [2], [3], [4], developing advanced graph classification algorithms [5], [6], and fostering cutting-edge clustering techniques [7], [8].valuable in At the core of motif mining and analysis lies the essential task of motif counting, which entails efficiently computing the occurrences of a specific motif (e.g., a triangle, a clique, etc.) within a graph. Moreover, the exploration of motif counting boasts a rich historical background, starting from the origins of triangle counting to its current significant interest in recent years.



(a) The colored graph (b) The colorful stars

Fig. 1. Illustration of the colored graph and colorful stars.

Lately, there has been growing interest within the computer theory community regarding motif on colored graphs [9], [10], [11]. A graph is categorized as colored when each of its nodes is assigned a distinct color, and the nodes at the endpoints of every edge have different colors, as depicted in Fig. 1(a). While these studies provide complex algorithms and insightful theoretical findings for motif counting in colored graphs, they often overlook the practical applications of such algorithms. Our manuscript aims to bridge this gap by specifically focusing on the problem of colorful star motif counting, where the star motifs themselves are colored graphs, as illustrated in Fig. 1(b). In summary, the applications can be categorized into three main directions: *(i) Direct Analysis*, which involves directly examining the network's structure or behavior through colorful star motifs, allowing us to identify key characteristics or emerging patterns. *(ii) Clustering Modeling*, where we leverage colorful star motifs, as higher-order structures, to build clustering models that preserve the connectivity of these motifs, enabling us to capture more meaningful network structures compared to traditional clustering approaches. *(iii) Approximation as Cliques*, where we approximate higher-order motifs as cliques, simplifying the analysis of dense subgraphs while retaining essential properties, providing a more efficient alternative to the computationally expensive clique enumeration process. The detailed explanation is as follows:

Application I: analyzing the structure of real-life colored graphs. It is a most straightforward application which based on the definition itself. Colored graphs with distinct colors assigned to adjacent vertices have numerous real-world applications:

- *Exam Scheduling Graph:* In educational institutions, exams are often scheduled on different days and time slots. To prevent conflicts among students who are enrolled in multiple exams, a colorful graph is employed. Each vertex

Received 4 April 2024; revised 21 September 2024; accepted 26 November 2024. Date of publication 13 December 2024; date of current version 5 February 2025. This work was supported in part by the National Key R&D Program of China under Grant 2021YFB3301303, and in part by NSFC under Grant 62202053, Grant U2241211, and Grant 62072034. Recommended for acceptance by R. Akbarinia. *(Corresponding authors: Rong-Hua Li; Guoren Wang.)*

Hongchao Qin, Rong-Hua Li, Ye Yuan, and Guoren Wang are with the Department of Computer Science, Beijing Institute of Technology, Beijing 100811, China (e-mail: qhc.neu@gmail.com; lironghuascut@gmail.com; yuan-ye@bit.edu.cn; wanggrbit@126.com).

Gao Sen is with the Computer Science, School of Computing, National University of Singapore, Singapore 119077.

Hongzhi Chen is with ByteDance Infrastructure Team, Beijing 100098, China.

Digital Object Identifier 10.1109/TKDE.2024.3514997

1041-4347 © 2024 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

represents an exam, while edges signify common students between exams. The objective is to assign colors to nodes (exams) in a way that ensures no two adjacent vertices have the same color.

- *Wireless Communication Network:* Wireless communication networks rely on effectively scheduling nodes or routers to transmit data without interference. To achieve this, a colorful graph is utilized. Vertices represent nodes, and edges denote communication links between nodes. The goal is to assign a distinct color to each node, ensuring that no two adjacent nodes with common communication links share the same color.
- *Colorful Roadmap:* In cartography, map coloring is essential for creating easily readable and distinguishable maps. The process involves assigning different colors to regions on a map, ensuring that no adjacent regions have the same color. To accomplish this, a colorful graph is employed, with vertices representing map regions and edges indicating shared borders between regions.

A colorful star motif refers to a star-shaped graph in which each pair of nodes has distinct colors. Counting these colorful star motifs in a colored graph allows us to analyze the behavior and structure of this specific motif. For example, in a social network where nodes represent individuals and edges symbolize social connections, a colorful star motif represents an influential individual (the central node) connected to diverse groups of people (the leaf nodes).

By quantifying the occurrence of colorful star motifs, valuable insights into the structure and dynamics of the graph can be uncovered. A significant number of colorful star motifs might reveal important hubs or connectors, indicating that certain nodes play pivotal roles in the network. Conversely, a low count of these motifs may suggest a more homogeneous or decentralized network structure. Thus, counting colorful star motifs provides a deeper understanding of the network's key behaviors and interactions.

Application II: modeling the higher-order clustering in a traditional graph. Since the colorful star motif is inherently a higher-order structure, it can be utilized for clustering modeling. The advantage of using the colorful star motif over other motifs lies in its ability to identify key hubs within the network, such as popular superstars in social networks or central relays in communication networks. This is due to the colorful star motif's definition, where the central node has a high degree, and the nodes connected to it are differently colored, indicating a higher likelihood of direct connections within the network (since in a colored graph, two connected nodes must have different colors). However, to mine the higher-order cluster, one possible method is first listing or counting the higher-order motifs of each nodes, and then aggregating the nodes into clusters.

For example, we propose a novel higher-order k-core model based on a concept of colorful h-star. Specifically, we first color the graph using a linear-time greedy coloring algorithm [12], [13] such that any two adjacent nodes in the graph have different colors. An h-star is a tree with a central node connected to the other h-1 nodes. A colorful h-star, denoted by $\star^h$, is a star in which all nodes have different colors (Fig. 1(b)). Clearly, a colorful h-star is a relaxed definition of h-clique, as an h-clique must form a colorful h-star. The colorful h-star k-core is a maximal subgraph of G in which each node acts as the center of at least k colorful h-stars. It is easy to derive that the traditional k-core is a special case of our colorful h-star k-core when $h = 2$. Similar to the colorful h-star k-core, we also introduce another higher-order model, the colorful h-star k-truss, and present the method for its decomposition.

Application III: speeding up h-clique densest subgraph mining: Since two connected nodes must have different colors in a colored graph, the colorful star motif can be seen as an approximation of a clique There is a considerable amount of work using cliques for dense subgraph modeling, such as the h-clique core and the h-clique densest subgraph. However, the main challenge with these approaches is that clique enumeration is extremely time-consuming. The colorful star motif counting can also speed up a series of well-known algorithms, such as the algorithms for mining the h-clique core and h-clique densest subgraph [14], [15], [16], [17]. For example, the goal of the h-clique densest subgraph problem is to maximize the average number of h-cliques over the number of nodes among all possible subgraphs. However, the computation on mining the h-clique densest subgraph is often very costly on large graphs for a relatively large h (e.g., $h > 6$), since the number of the motif h-clique is exponential level rising with respect to h.

Unlike counting the h-cliques, we show that the number of colorful h-stars for each central node can be computed and updated in at most $O(h\chi)$ time by a novel dynamic programming (DP) algorithm, where χ is the number of colors used to color the graph. Based on our DP technique, we develop an $O(h \times m)$-time algorithm to compute the colorful h-star core decomposition of the graph using $O(hn + m)$ space, thus our model can be scalable to handle large graphs even for a relatively large k. In addition, based on the colorful h-star core model, we also develop a new graph reduction technique for the h-clique densest subgraph problem which can significantly speed up the state-of-the-art approximate h-clique densest subgraph mining algorithm [14].

Around the applications above, the main contributions of this manuscript are as follows.

Concepts: We first intrduce the concept of colorful h-star in a colorful graph, which is a star-shaped graph where any two nodes have different colors. Next, to model the higher-order clustering, we propose two novel higher-order cohesive subgraph models, i.e., colorful h-star core and colorful h-star truss.

Algorithms: To count the colorful star motif, we propose a novel DP-based updating technique which can dynamically update the number of colorful h-stars for a node when one of its neighbor node is deleted, without recomputing the colorful h-star counts from scratch. With this DP-based updating technique, we propose an efficient peeling algorithm to compute the colorful h-star core decomposition which consumes $O(hm)$ time and $O(hn + m)$ space. Furthermore, we introduce a decomposition algorithm for the colorful h-star truss, characterized by a time complexity of $O(hm^{1.5})$ and a space complexity of $O(hm)$. Since h is often very small (less than 10), our

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

work provides a near-linear time solution for higher-order graph analysis applications.

Applications: Once we have enumerated the number of colorful h-stars in the graph, we can gain valuable insights into the underlying structure of the colored graph (Application I). Building upon the innovative techniques for counting and updating colorful h-stars, we have devised a highly efficient peeling algorithm for decomposing the colorful h-stars into their respective cores or trusses (Application II). As the h-clique core can achieve a good approximation to the model of h-clique densest subgraph [14]. We establish that the colorful h-star core model can be regarded as a relaxation of the h-clique core model. To expedite the approximate mining of the densest subgraph exhibiting h-clique characteristics, we introduce a novel technique based on colorful h-star cores. This approach effectively reduces the graph size without compromising the approximation performance of the h-clique core based algorithm (Application III).

Experiments: We conduct extensive experiments on 11 large real-life datasets to evaluate our algorithms. The results show that: (1) the proposed colorful h -star core/truss decomposition algorithm is very efficient to handle large graphs which takes only a few seconds on the large graphs with more than 1M nodes and 10M edges even when k=6. (2) the proposed graph reduction technique can achieve one order of magnitude speedup over the state-of-the-art approximate h -clique densest subgraph mining algorithm. For example, on the largest datasets LiveJournal (more than 4M nodes and 40M edges), the state-of-the-art algorithms takes 113 seconds on the original graph. However, when integrating with our graph reduction technique, such an algorithm only takes 11 seconds. (3) Our colorful h-star k-core with maximum k also provides a very good approximation of the h-clique densest subgraph. On most datasets, it can achieve the same and even better approximation ratio than the state-of-the-art method [14]. (4) Graph coloring algorithms affect the performance of our algorithms and qualities of approximation, but nodes' distributions are generally similar according to different graph colorings. In addition, we also conduct a case study on DBLP, and the results show that our model can indeed identify some interesting and meaningful communities with different semantics compared to the previous models.

II. Concepts of Colorful Star

A. Colored Graph

The minimum graph coloring problem is a fundamental topic in graph theory. The goal of this problem is to assign colors to the vertices of a graph such that no two adjacent vertices share the same color, while minimizing the number of colors used. More formally, a colored graph can be defined as follows:

Definition 1 (Colored graph): A colored graph is a node-labeled graph $G = (V, E, \mathcal{C})$ where $V(E)$ are the set of nodes(edges), $\mathcal{C} : V \rightarrow C$ denotes the colored function where C is the set of all the colors, which satisfies that for each edge, the color of the two end nodes are different, i.e., $\mathcal{C}(u) \neq \mathcal{C}(u)$ for each $(u, v) \in E$.

However, the minimum graph coloring problem is NP-hard [18], which makes it challenging to obtain a colored graph with the smallest possible number of colors $|C|$. In this manuscript, we utilize a linear-time greedy coloring algorithm [12], [13] to obtain a valid coloring for the graph. Fortunately, the experimental results in Section V-D demonstrate that the various greedy graph coloring methods have little impact on the effectiveness of the colorful star motif counting algorithms. Therefore, in the remaining of this manuscript, we will skip the process of obtaining a colored graph and instead assume that a colored graph is readily given.

Below are some basic notations: let V_G, E_G denote the nodes and edges in G; and $n = |V_G|, m = |E_G|$ be the number of nodes and edges in G. Let N_u^G denotes the set of neighbor nodes of u in G, and $d_u^G = |N_u^G|$ denotes the degree of u in G. A subgraph $S = (V_S, E_S)$ is an induced subgraph of G if $V_S \subseteq V$ and $E_S = \{(u, v)|(u, v) \in E, u \in V_S, v \in V_S\}$.

B. Colorful Star

An h-star motif is a tree, with one central node having degree $h - 1$ and the other $h - 1$ nodes having degree 1. However, in a colored graph, *a colorful h-star* motif is a h-star motif in which any pair of nodes has different colors. Based on a valid colored graph, we define the concept of colorful h-star motif as follows.

Definition 2 (Colorful h-star motif): Given a colored graph $G = (V, E, \mathcal{C})$ and an integer $h \geq 2$, a colorful h-star motif (denotes by $\star^h$) is a subgraph in which (i) one central node has degree $h - 1$ and the other $h - 1$ nodes have degree 1; (ii) any pair of nodes has different colors, i.e., $\mathcal{C}(u) \neq \mathcal{C}(v)$ for each $u, v \in \star^h$.

Below, we show that counting the colorful h-star motif is practically computing the central degree of the colorful h-star.

Definition 3 (Central degree of a motif (motif degree)): Given a colored graph $G = (V, E, \mathcal{C})$ and an integer h, the central degree of a motif M for a node u in G, denoted by $d_u^G(M)$, is the number of the motif which centered on u.

For example, the central degree of h-star (h-star degree) for a node u in G is the number of h-stars centered on u; the central degree of colorful h-star (colorful h-star degree) for a node u in G, denoted by $d_u^G(\star^h)$, is the number of colorful h-stars centered on u; and the central degree of h-clique (h-clique degree) for a node u in G, denoted by $d_u^G(\boxtimes^h)$, is the number of h-cliques that u participates in.

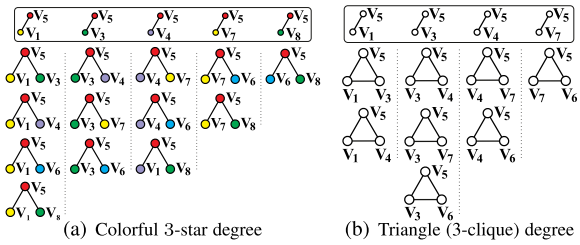
Clearly, in a traditional graph G, the central degree of h-star is $\binom{d_u^G}{h\text{-}1}$. But those h-stars are not all colorful in a colored graph. As shown in Fig. 1(a), consider V_3 to the be the central node and $h = 3$, we can easliy get that the number of h-stars which are centered on V_3 is $\binom{4}{2} = 6$. However, the color of V_2 and V_5 are same, so the subgraph of $\{V_3, V_2, V_5\}$ is not a colorful h-star. Therefore, the alglorithm for counting the h-star is different from that for the colorful h-star.

Note that, by Definition 2, any pair of nodes in a colorful h-star must have different colors, as shown in Fig. 1(b). The h-clique model also shares this property, and thus it must be a colorful h-star. Indeed, unlike the h-clique, the colorful h-star may miss some edges between two nodes, even when they have different

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

TABLE I
MAIN SYMBOLS

Symbols	Definitions
$G = (V, E, \mathcal{C})$	the colored graph
$\mathcal{C}(u)$	the color of node u in G
N_u^G, d_u^G	the neighborhood, degree of node u in G
$\star^h$	the colorful h-star (Definition 2)
$\boxtimes^h$	the h-clique
$d_u^G(M)$	the central degree of a motif M for u in G (Definition 3)
$d_u^G(\star^h)$, $d_u^G(\boxtimes^h)$	the central degree (motif degree) of $\star^h$, $\boxtimes^h$ for u in G
C_k	the k-core (Section 4.1.1)
c_u	the core number of a node u (Section 4.1.1)
$\mathsf{CSC}_k(\star^h)$ (abbr. CSC_k^h)	the colorful h-star k-core (Definition 4)
$csc_u(\star^h)$ (abbr. csc_u^h)	the colorful h-star core number of a node u
T_k	the k-truss (Section 4.1.2)
t_e	the k-truss number of an edge e (Section 4.1.2)
$\mathsf{CST}_k(\star^h)$ (abbr. CST_k^h)	the colorful h-star k-truss (Definition 6)
$cst_e(\star^h)$ (abbr. cst_e^h)	the colorful h-star truss number of an edge e
$\mathsf{CC}_k(\boxtimes^h)$ (abbr. CC_k^h)	the h-clique k-core (Definition 8)
$cc_u(\boxtimes^h)$ (abbr. cc_u^h)	the h-clique core number of a node u



(a) Colorful 3-star degree (b) Triangle (3-clique) degree

Fig. 2. Illustration of the central degree of v_5.

colors. Thus, we consider a subgraph induced by the nodes of a colorful h-star as a relaxed h-clique subgraph. Such a nice feature motivates us to use colorful h-stars to replace h-cliques as a building block to perform higher-order graph analysis.

Table I lists the main symbols used in this manuscript and their definitions; Example 1 illustrates the definition of central degree.

Example 1: Reconsider the graph G in Fig. 1(a). Suppose $h = 3$. For node v_5, we can easily derive that its central degree of 3-star is $\binom{d_{v_5}^G}{h\text{-}1} = 15$. Also, v_5 participates in eight 3-cliques listed in Fig. 2(b), thus its h-clique degree $d_{v_5}^G(\boxtimes^h)$ is equal to 8. However, the central degree of the colorful h-star of v_5, denoted by $d_{v_5}^G(\star^h)$, is 13, because there exist 13 colorful 3-stars which are centered on v_5 as enumerated in Fig. 2(a).

For higher-order graph analysis, it often needs to compute the central degree of a node. For example, the h-clique based higher-order graph analysis applications [15], [16], [17] require to calculate the h-clique degree of a node. When using our colorful h-star as a relaxation of h-clique for higher-order graph analysis, we also need to compute the colorful h-star degree of a node. However, computing such a quantity for each node is a nontrivial task. Moreover, some applications may also need to dynamically update the colorful h-star degree when the graph is updated by deleting an edge.

Based by Definitions 2 and 3, we show the problem formulation as follows.

Problem 1 (Colorful Star Motif Counting): Given a colored graph $G = (V, E, \mathcal{C})$ and parameter $h \geq 2$, the goal of colorful star motif counting is to calculate the central degree of the colorful h-star, i.e., $d_u^G(\star^h)$, for each node $u \in V$.

Note that, we define the problem of counting the colorful star motif centered on a given node u in a graph. However, if we extend this definition to counting colorful star motifs across the entire graph, the above definition remains valid. To count all the colorful star motifs in a colored graph $G = (V, E, \mathcal{C})$, we simply need to sum the colorful h-star degrees of all nodes, i.e., $\sum_{u \in V} d_u^G(\star^h)$, to obtain the result. This is because, in Definition 3, we specified that the motif is centered on node u. Therefore, the colorful h-star motif centered on node u will not be identical to the colorful h-star motif centered on node v (where $v \neq u$). To address this issue, our approach focuses on efficiently computing the colorful h-star degree for each node in the graph. Additionally, we propose a fast update algorithm that updates the colorful h-star degree when the neighbors of the central node u are removed.

Challenge: Clearly, the h-star degree can be derived by a combinatorial formula. However, there does not exist a combinatorial formula that can be used to compute the colorful h-star degree for a node u. A straightforward approach is to enumerate all h-stars of u and then count all the valid colorful h stars. Such a straightforward approach is clearly intractable for high-degree nodes due to combinatorial explosions. Therefore, a challenging problem is how can we develop practical solutions to compute the colorful h-star degree of a node without brute-force enumeration. Furthermore, when the graph is updated, how can we derive a fast solution to update the colorful h-star degree of a node without recomputing the colorful h-star degree from scratch. Below, we will develop efficient algorithms based on a technique of dynamic programming to tackle these challenges.

III. Algorithms for Colorful Star Counting

As counting the colorful star in the whole graph can be easily extended by calculating the central degree of the colorful h-star for each node $u \in V$. In this section, we first propose a dynamic programming (DP) algorithm to compute the colorful h-star degree of a given node u. Then, we develop an efficient updating algorithm to update the colorful h-star degree of the node u when one of its neighbors is removed.

A. The DP-Based Counting Algorithm

Algorithm 1 shows the pseudocode of our DP algorithm to calculate the colorful h-star degree of u. First, Algorithm 1 invokes the greedy coloring procedure [12], [13] following any ordering on nodes (break ties by node ID) to obtain a valid coloring for all nodes (line 1). After that, the algorithm computes $d_u^G(\star^h)$ using a DP approach (lines 2-6). Below, we present a detailed description of the DP procedure.

First, for any node u, the neighbors of u can be divided into χ groups in terms of their colors, here χ is the number of colors used by the greedy coloring algorithm. Let $\mathsf{Group}(i)$ be the set of u's neighbor nodes whose color values are equal to i, i.e., $\mathsf{Group}(i) = \{v | v \in N_u^G, \mathcal{C}(v) = i\}$. The size of each color group is denoted by $\mathsf{cnt}(i) = |\mathsf{Group}(i)|$. Let $\mathsf{DP}(i, j)$ be

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: The DP-Based Counting Algorithm.

Input: A graph G, a node u and a parameter h
Output: The colorful h-star degree $d_u^G(\star^h)$

$\mathcal{C}(u)$ for $u \in G \leftarrow \mathsf{GreedyColoring}(G)$;
$\chi \leftarrow$ the number of colors in $\mathcal{C}(v)$;
for $i = 1$ **to** χ **do**
  $\mathsf{Group}(i) \leftarrow \{v | v \in N_u^G, \mathcal{C}(v) = i\}$;
  $\mathsf{cnt}(i) \leftarrow |\mathsf{Group}(i)|$;
$d_u^G(\star^h) \leftarrow \mathsf{Counting}(\chi, h, \mathsf{cnt})$;
return $d_u^G(\star^h)$;

Procedure $\mathsf{GreedyColoring}(G)$
Let π' be any ordering on nodes;
$flag(i) \leftarrow -1$ for $i = 1, \cdots, \chi$;
for *each node* $v \in \pi'$ *in order* **do**
  for $u \in N_v^G$ **do**
    $flag(\mathcal{C}(u)) \leftarrow v$;
  $c \leftarrow \min\{i | i > 0, flag(i) \neq v\}$;
  $\mathcal{C}(v) \leftarrow c$;
return $\mathcal{C}(v)$ for all $v \in G$;

Procedure $\mathsf{Counting}(c, h, \mathsf{cnt})$
for $i = 0$ **to** c **do**
  for $j = 0$ **to** h **do**
    if $j = 0$ **then** $\mathsf{DP}(i, j) \leftarrow 1$;
    else if $i < j$ **then** $\mathsf{DP}(i, j) \leftarrow 0$;
    else
      $\mathsf{DP}(i, j) \leftarrow \mathsf{DP}(i-1, j-1) \times \mathsf{cnt}(i) + \mathsf{DP}(i-1, j)$;
return $\mathsf{DP}(c, h-1)$;

the number of ways to choose j nodes with different colors from $\mathsf{Group}(1) \cup \mathsf{Group}(2) \cup \cdots \cup \mathsf{Group}(i)$. The computation of $\mathsf{DP}(i, j)$ can be divided into two cases, by considering whether or not select a node from $\mathsf{Group}(i)$.

Case 1: If we choose a node from $\mathsf{Group}(i)$, we just need to choose $j-1$ nodes with different color values from $\mathsf{Group}(1) \cup \mathsf{Group}(2) \cup \cdots \cup \mathsf{Group}(i-1)$. Therefore, in this case, we obtain $\mathsf{DP}(i-1, j-1) \times \mathsf{cnt}(i)$ colorful $(j+1)$-stars centered at u.

Case 2: When we do not choose a node from $\mathsf{Group}(i)$, we must collect j nodes from $\mathsf{Group}(1) \cup \mathsf{Group}(2) \cup \cdots \cup \mathsf{Group}(i-1)$ to obtain $\mathsf{DP}(i-1, j)$ colorful $(j+1)$-stars.

We can derive the colorful h-star degree of a node by adding up the results of the above two cases. Specifically, we are able to obtain the following recursive DP equation:

$$\mathsf{DP}(i, j) = \mathsf{DP}(i-1, j-1) \times \mathsf{cnt}(i) + \mathsf{DP}(i-1, j), \quad (1)$$

for all $i \in [1, \ldots, \chi], j \in [1, \ldots, h], i \geq j$.

The base cases can be set as follows:

$$\mathsf{DP}(i, 0) = 1, \text{for all } i \in [0, \ldots, \chi]$$

$$\mathsf{DP}(i, j) = 0, \text{for all } i \in [0, \ldots, \chi], j \in [0, \ldots, h], i < j, \quad (2)$$

Based on (1) and (2), we can compute the colorful h-star degree of u by dynamic programming.

Example 2: Fig. 3 shows the computational procedure of v_5's colorful 3-star degree. The DP algorithm computes $d_{v_5}^G(\star^h)$ by gradually involving color groups from "2=Green" to "5=Blue". It is impossible to pick two neighbor nodes of different colors from the first two color groups ($\mathsf{DP}(2, 2) = 0$), since there does not exist a neighbor node of v_5 with color value "1=Red".

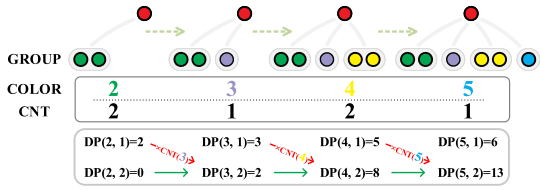


Fig. 3. Computation for colorful 3-star degree of v_5 in Fig. 1(a).

Obviously, there are two ways to choose a node from the first two color groups ($\mathsf{DP}(2, 1) = 2$), because v_5 is connected to two "2=Green" nodes. In order to choose two neighbor nodes with different colors from the first three color groups ($\mathsf{DP}(3, 2)$), the DP algorithm can pick one node from the first two color groups ($\mathsf{DP}(2, 1)$) and then pick another node from any of the "3=Purple" group ($cnt(3)$), or do not consider the third color group and choose two nodes with different colors directly from the first two color groups ($\mathsf{DP}(2, 2)$). Finally, v_5's colorful 3-star degree $d_{v_5}^G(\star^h) = \mathsf{DP}(5, 3-1) = 13$ can be computed in the similar method following an increasing order of color values as shown in Fig. 3.

Space optimization: Note that in the DP table ($\mathsf{DP}(i, j)$ for all i and j), calculating $\mathsf{DP}(i, j)$ relies only on the result of $\mathsf{DP}(i-1, j)$ and $\mathsf{DP}(i-1, j-1)$, which are both in the $\mathsf{DP}(i-1, \cdot)$ array. This means that all the $\mathsf{DP}(p, \cdot), p < i-1$ arrays contribute nothing to this calculation. Therefore, whenever computing on the DP table, there are only two active arrays, $\mathsf{DP}(i-1, \cdot)$ for *reading* and $\mathsf{DP}(i, \cdot)$ for *writing*. Based on this observation, we can use two rolling arrays to *read* and *write* alternately, which significantly reduces the space consumption to $O(2 \times h)$.

To further optimize the space usage, for each i we compute $\mathsf{DP}(i, j)$ following the descending order of j, i.e., varying j from h to 0. Furthermore, we simplify the data structure and reduce the two-dimensional DP table to a single array of size h. Note that we can calculate $\mathsf{DP}(i, j), \mathsf{DP}(i, j-1), \ldots \mathsf{DP}(i, j-1)$ if we have stored $\mathsf{DP}(i-1, j), \mathsf{DP}(i-1, j-1), \ldots \mathsf{DP}(i-1, j-1)$. After χ-round iterative calculation, we can obtain $\mathsf{DP}(h\text{-}1)$ which is equal to $d_u^G(\star^h)$. The algorithm only consumes $O(h)$ space by using this trick. The following theorem details the time and space complexity of our DP algorithm.

Theorem 1: Given a graph G, a node u and an integer h, Algorithm 1 computes the colorful h-star degree of u in $O(h \times \min\{\chi, d_u^G\})$ time using $O(h)$ space, where χ is the maximum color value of all nodes in G.

Proof: For each node u, the number of color groups of u's neighbor nodes is bounded by d_u^G. Thus, the color i in the DP equation in (1) is bounded by $\min\{d_u^G, \chi\}$. As a result, the time complexity of the DP algorithm for computing the colorful h-star degree of u is $O(h \times \min\{\chi, d_u^G\})$. For the space complexity, the algorithm only needs to maintain an $O(h)$ DP table using the proposed space optimization technique, thus the theorem is established. □

B. The DP-Based Updating Algorithm

Here we consider the problem of updating the colorful h-star degree of a node u when one of its neighbor node v is deleted. To

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2: Example of the Updating Algorithm.

Input: A graph G, a node u, the removed neighbor v and h
1 Initialize $\mathcal{C}(v), \chi, \mathsf{cnt}$ as lines 1-5 in Algorithm.1;
2 $d_u^G(\star^h) \leftarrow$ Counting+(χ, h, cnt); // record $\mathcal{DP}_u \leftarrow$ the table $\mathcal{DP}$ in the Counting+ procedure
3 $d_u^{G\setminus v}(\star^h) \leftarrow$ Updating$(\mathcal{DP}_u, \mathcal{C}(v))$;
4 **Procedure** Counting+(χ, h, cnt)
5 $\mathcal{F}(\cdot) \leftarrow 0; \mathcal{F}(0) \leftarrow 1; \chi' \leftarrow 1;$
6 **for** $i = 2$ **to** χ **do**
7 **for** $j = h$ **to** 1 **do**
8 $\mathcal{F}(j) \leftarrow \mathcal{F}(j-1) \times \mathsf{cnt}(i) + \mathcal{F}(j);$
9 **for** $j = 1$ **to** h **do**
10 $\mathcal{G}(j) \leftarrow \mathcal{F}(j-1) \times \mathsf{cnt}(\chi');$
11 $\mathcal{DP}(j) \leftarrow \mathcal{F}(j) + \mathcal{G}(j);$
12 **return** $\mathcal{DP}(h\text{-}1)$;
13 **Procedure** Updating$(\mathcal{DP}_u, \chi')$
14 $\mathcal{F}(0) \leftarrow 1;$
15 **for** $i = 1$ **to** h **do**
16 $\mathcal{G}(i) \leftarrow \mathcal{F}(i-1) \times \mathsf{cnt}(\chi');$
17 $\mathcal{F}(i) \leftarrow \mathcal{DP}_u(i) - \mathcal{G}(i);$
18 $\mathsf{cnt}(\chi') \leftarrow \mathsf{cnt}(\chi') - 1;$
19 **for** $i = 1$ **to** h **do**
20 $\mathcal{G}(i) \leftarrow \mathcal{F}(i-1) \times \mathsf{cnt}(\chi');$
21 $\mathcal{DP}_u(i) \leftarrow \mathcal{F}(i) + \mathcal{G}(i);$
22 **return** $\mathcal{DP}_u(h\text{-}1)$;

this end, a straightforward algorithm is to recompute the colorful h-star degree of u after removing v by using Algorithm 1, which takes $O(\chi h)$ time. Clearly, such an algorithm is inefficient when we need to frequently handle edge removals. A natural question is that can we have a better algorithm to update the colorful h-star degree of a node without recomputing from scratch? Below, we will develop a novel algorithm to achieve this goal.

Our updating algorithm is based on a key observation: the removal of v reduces $\mathsf{cnt}(\mathcal{C}(v))$, thus it only affects the result computed in *Case 1* of the DP equation. Therefore, it is unnecessary to re-calculate the entire DP table again, because the result in *Case 2* remains the same. Based on this observation, we need to decompose the DP equation into two different cases. Specifically, let us consider a node v and a color value χ'. Then, the colorful $(i+1)$-stars on v can be divided into two various types: the leaves of colorful $(i+1)$-stars are colored with or without χ'. Let $\mathcal{G}$ and $\mathcal{F}$ be two arrays where $\mathcal{F}(i)$ and $\mathcal{G}(i)$ denotes the number of the former type and the latter type respectively. More formally, let A be a set of nodes,

$$P(A): \forall\{u,w\}(\{u,w\} \subseteq A \rightarrow \mathcal{C}(u) \neq \mathcal{C}(w))$$
$$Q(A): \forall u(u \in A \rightarrow \mathcal{C}(u) \neq \chi')$$
$$\bar{Q}(A): \exists u(u \in A \wedge \mathcal{C}(u) = \chi')$$
$$\mathcal{F}(i) = |\{A | A \subseteq N_v^G, |A| = i, P(A), Q(A)\}|$$
$$\mathcal{G}(i) = |\{A | A \subseteq N_v^G, |A| = i, P(A), \bar{Q}(A)\}|$$
$$\mathcal{DP}(i) = |\{A | A \subseteq N_v^G, |A| = i, P(A)\}|.$$

Here $\mathcal{DP}(i)$ indicates the colorful $(i+1)$-star degree of v. Then, we have the following result.

Theorem 2: After removing a neighbor v, the colorful h-star degree of u can be updated by the following DP equation

$$\mathcal{G}(i) \leftarrow \mathcal{F}(i-1) \times \mathsf{cnt}(\mathcal{C}(v))$$
$$\mathcal{DP}(i) \leftarrow \mathcal{F}(i) + \mathcal{G}(i), \tag{3}$$

for all $i \in [1, \ldots, h]$. The updated $d_u^{G\setminus v}(\star^h)$ equals $\mathcal{DP}(h\text{-}1)$.

Proof: First, we set χ' as $\mathcal{C}(v)$. Apparently, $\mathcal{G}(i)$ can be derived by $\mathcal{F}(i-1) \times \mathsf{cnt}(\mathcal{C}(v))$, because the current colorful i-stars do not have the leaves with color value χ'. Thus, by picking any neighbor colored by χ' and adding it to colorful i-stars, we can obtain new colorful $(i+1)$-stars. All colorful $(i+1)$-stars contain colorful $(i+1)$-stars both with and without a neighbor colored by χ', therefore $\mathcal{DP}(i) = \mathcal{F}(i) + \mathcal{G}(i)$. □

The example of the updating algorithm is outlined in Algorithm 2. First, we initialize $\mathcal{C}(v), \chi, \mathsf{cnt}$ as lines 1-5 in Algorithm 1; count the colorful h-star degree of u by invoking Counting+; and record $\mathcal{DP}_u$ to be the table $\mathcal{DP}$ in the Counting+ procedure.

Second, we re-design a DP algorithm to compute the colorful h-star degree of u by using the arrays $\mathcal{F}$, $\mathcal{G}$ as defined in (3). The computation of $\mathcal{F}(i)$ is similar to the DP procedure in Algorithm 1. Note that the same trick that computing $\mathcal{F}$ following the descending order of i is also applied to reduce the space usage (lines 6-8). After that, based on the $\mathcal{F}$ array, we can compute $\mathcal{G}$ and $\mathcal{DP}$ (lines 9-11).

Third, in the updating stage, we develop a novel technique to update the colorful h-star degree of u in G/v when any neighbor v of u is removed, instead of recomputing from scratch. Our technique is based on the following intuition. When removing v, the colorful h-stars that are centered on u and contain v will disappear, and the other colorful h-stars remain unchanged. Therefore, only $\mathcal{G}$ needs to be updated. To achieve this, we need to restore $\mathcal{G}$ and $\mathcal{F}$ based on the $\mathcal{DP}$ array when we consider the node u and its deleted neighbor node v. Note that we can use (3) to restore $\mathcal{G}$ and $\mathcal{F}$ by $\mathcal{DP}$. A difference compared to the counting stage is that χ' must be set to $\mathcal{C}(v)$ (lines 3, 13), because only the number of colorful h-stars colored by $\mathcal{C}(v)$ will reduce. The proposed updating technique consists of three steps, restoring (lines 15-17), updating (line 18) and regenerating (lines 19-21).

S1: In the restoring step, our algorithm restores $\mathcal{F}(i)$ and $\mathcal{G}(i)$ on the basis of $\mathcal{DP}(i)$ and $\mathsf{cnt}(\mathcal{C}(v))$.
S2: In this step, Algorithm 2 updates the color set of u's neighbors.
S3: In the last step, Algorithm 2 calculates $\mathcal{G}(i)$ and $\mathcal{DP}(i)$ based on $\mathcal{F}(i)$ and the updated color groups of u 's neighbors.

Example 3: For the graph G in Fig. 1(a), consider the case of the removal of v_1 which is a neighbor node of v_5 with color value "4=Yellow". The entire updating procedure shown in Fig. 4 contains three steps. First, the updating algorithm refreshes the elements in $\mathcal{G}$ and $\mathcal{F}$ alternately according to $\mathcal{DP}$ following the <①,②,③,④> order. Then, the color group with regard to v_1 is updated after removing v_1. Finally, the elements of $\mathcal{DP}$ and $\mathcal{G}$ are replaced by new values indicated in Green, following the <⑤,⑥,⑦,⑧> order. The v_5's colorful

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

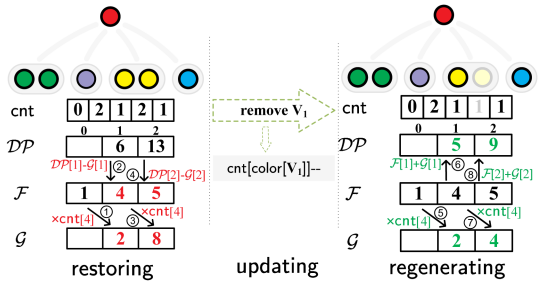


Fig. 4. Updating colorful 3-star degree of v_5 in Fig. 1(a).

3-star degree $d^G_{v_5}(\star^h) = \mathcal{DP}(2)$ will be updated to 9 when the algorithm terminated.

The correctness of Algorithm 2 can be guaranteed by Theorem 2. A nice feature of our update technique is that both the restoring and recomputing steps consume $O(h)$ time, which is much more efficient than the straightforward re-computation based updating algorithm. The following theorem shows the time and space complexity of our algorithm.

Theorem 3: Given a graph G, an integer h and a node u, after removing any neighbor of u, the updating procedure of Algorithm 2 updates its colorful h-star degree in $O(h)$ time using $O(\min\{\chi, d^G_u\} + h)$ space, where χ is the maximum color value of all nodes in G.

Proof: For the time complexity, both the restoring and the regenerating steps consumes $O(h)$ time, and the updating steps takes $O(1)$ time. Thus, the time complexity of our updating technique is $O(h)$. For the space complexity, we only need to store and maintain the arrays cnt and $\mathcal{DP}$, which consume $O(\min\{\chi, d^G_u\} + h)$ space for each node. This is because the size of the array cnt is bounded by $O(\min\{\chi, d^G_u\}$, while $\mathcal{DP}$ consumes $O(h)$ space. □

IV. Applications of Colorful Star Counting

There are three primary applications for counting colorful stars: analyzing the structural properties of real-life colorful graphs, modeling higher-order clustering, and accelerating the mining of the densest subgraph exhibiting h-clique characteristics in traditional graphs. In the following sections, we will focus on elucidating how the techniques of colorful star counting contribute to achieving the latter two applications.

A. Modeling the Higher-Order Clustering

The colorful h-star can serve as a fundamental building block for constructing higher-order clustering. Currently, the basic units of higher-order clustering models primarily include traditional motifs (such as triangles and four-node stars), cliques, and hyper-edges. Clustering models that aggregate these elements mainly consist of spectral clustering and dense subgraph models (such as k-core [19], k-truss [20], k-ECC [21], and so on. Since methods like spectral clustering and k-means require a predetermined number of clusters and lack a definitive structural commonality in the final clustering results, we focus on using dense subgraph models for higher-order clustering of the colorful h-star. Due to space limitations, this manuscript discusses only the colorful h-star k-core and k-truss models; however, other models, such as k-ECC, can also be adapted for extension with some customization. Therefore, this manuscript aims to explore how to use the colorful h-star as a foundational element for constructing higher-order clustering, an idea that may guide future researchers in pursuing more interesting work on specific problems.

Moreover, using the colorful h-star to model higher-order clustering enables us to capture both the strength of connections and the diversity of nodes within a network. While standard k-core and k-truss models primarily focus on item-based connectivity, they often overlook the variety and diversity of connections. By incorporating node colors and considering higher-order structures like colorful h-stars, we can represent more complex relationships. This approach allows us to identify subgraphs that are not only tightly connected but also involve diverse and meaningful interactions among nodes.

In this section, we introduce two innovative models for higher-order cohesive subgraphs: the colorful h-star k-core/truss. Next, we devise a near-linear peeling algorithm, leveraging the counting and updating techniques specifically designed for colorful h-stars.

1) The Colorful H-Star Core Model: We start by reviewing the definition of the classical k-core [19], and then introduce the concept of colorful h-star core. Given a graph G and an integer k, a k-core, denoted by C_k, is a maximal induced subgraph of G such that every node in C_k has a degree no smaller than k, i.e., $d^{\mathsf{C}_k}_u \geq k$ for every $u \in \mathsf{C}_k$ [19]. The core number of a node u, denoted by c_u, is the largest integer k such that there exists a k-core containing u [19]. The maximum core number of a graph G, denoted by c_G, is the maximum value of core numbers among all nodes in G. The maximum core number c_G is also referred to as the degeneracy of G [22]. The k-cores have four crucial properties: (a) k-cores are nested, more formally, $C_i \subseteq C_j$ if $i > j$; (b) a k-core can be disconnected; (c) $d^{\mathsf{C}_k}_u \geq c_u$; (d) the k-core is unique and the core decomposition can be computed in $O(m + n)$ time [23].

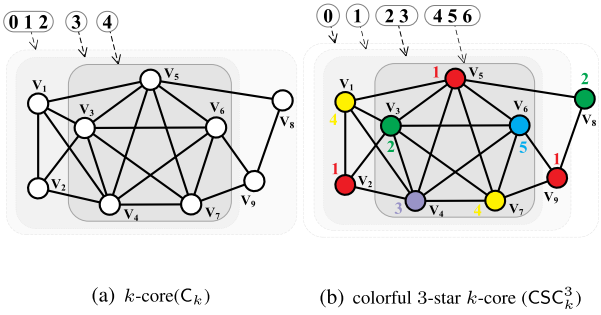
Inspired by the definition of the k-core, we define the colorful h-star k-core as follows:

Definition 4 (Colorful h-star k-core): Given a colored graph G, an integer k and the size h of a colorful star $\star^h$, a colorful h-star k-core, denoted $\mathsf{CSC}_k(\star^h)$ (abbr. CSC^h_k), is a maximal subgraph such that $\forall u \in V_{\mathsf{CSC}^h_k}$, $d^{\mathsf{CSC}^h_k}_u(\star^h) \geq k$.

Based on Definition 4, the colorful h-star core number of u, denoted by $csc_u(\star^h)$ (abbr. csc^h_u), is the largest k such that there exists a colorful h-star k-core containing u. And the maximum colorful h-star core number of a graph G, denoted by $\hat{csc}$, is the maximum value of colorful h-star core numbers among all nodes.

Example 4: Let $\star^3$ be a colorful 3-star. Fig. 5(a) depicts k-cores of the graph in Fig. 1(a). The number k in each rectangle indicates the k-core contained in that rectangle. For example, the subgraph induced by $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ is a 3-core. The entire graph is the 0-core, 1-core and 2-core. Fig. 5(b) shows all CSC^3_k of the graph. The number k in each rectangle indicates the CSC^3_k contained in that rectangle. For example, the subgraph

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.



(a) k-core(C_k) (b) colorful 3-star k-core (CSC_k^3)

Fig. 5. k-core and colorful h-star k-core.

induced by $\{v_3, v_4, v_5, v_6, v_7\}$ is a CSC_6^3 since each node in the 5-clique participates in 6 colorful 3-stars. We can see that k-cores and $(\star^3, k)$-cores are the same for $k = 3, 4$, but obviously different when $k = 1, 2$.

Similar to k-cores, it is easy to derive that the colorful h-star k-core also have the following properties: (a) CSC_k^h are nested, that is for any two integers i and j, if $i > j$, then $\mathsf{CSC}_j^h \subseteq \mathsf{CSC}_j^h$; (b) a CSC_k^h may be disconnected; (c) $csc_u^h \leq d_u^G(\star^h)$; (d) CSC_k^h are unique, which means that there exists only one CSC_k^h for a specific h in G. Below, we show that the colorful h-star cores can be computed in near-linear time.

Based on Definition 4, we define the colorful h-star core decomposition problem as follows.

Problem 2 (Colorful h-star Core Decomposition): Given a colored graph G and an integer h, the goal of colorful h-star core decomposition is to compute the colorful h-star core numbers for all nodes in G.

The challenge of mining a colorful h-star k-core arises primarily from the difficulty of efficiently counting colorful h-stars, especially for nodes with high degrees. A straightforward approach, which involves enumerating all h-stars for each node and counting the valid colorful ones, is computationally intractable due to combinatorial explosion. This problem is further compounded by the lack of a combinatorial formula for directly computing the colorful h-star degree of a node. To address this, practical solutions must be developed to compute the colorful h-star degree without resorting to brute-force enumeration. Additionally, when the graph is updated, a fast method is needed to update the colorful h-star degree without recomputing it from scratch. Moreover, mining the colorful h-star k-core, which involves finding maximal subgraphs where each node has at least k colorful h-stars, presents further complexity. This requires simultaneously preserving the structural and colorful constraints, making the problem even more challenging in large-scale graphs.

It is easy to derive that a colorful h-star k-core can be obtained by iteratively removing nodes whose colorful h-star degrees are less than the given k. Therefore, we can devise a peeling algorithm to determine the core numbers for all nodes, which iteratively deletes the node with the smallest colorful h-star degree. Such an peeling algorithm is outlined in Algorithm 3.

First, Algorithm 3 computes the colorful h-star degree for each node in V using the DP algorithm given in Algorithm 2

Algorithm 3: The Colorful h-Star Core Decomposition.

Input: A graph $G = (V, E)$ and an integer h
Output: The colorful h-star core number for each $v \in V$

```
1  Initialize C(v), χ, cnt as lines 1-5 in Algorithm.1;
2  for u ∈ V do deg[u] ← Counting+(u); // Algorithm 2
3  Sort nodes in a non-decreasing order of their colorful h-star
     degrees;
4  H ← G; max_core ← 0;
5  for i ← 1 to |V| do
6      u ← arg min_{v∈V_H} deg[v];
7      if deg[u] > max_core then
8          max_core ← deg[u];
9      csc_u^h ← max_core;
10     for each w ∈ N_u^H do
11         deg[w] ← Updating(DP_u, C(w)); // Algorithm 2
12     Delete u from H;
13     Resort the nodes of H;
14 return csc_u^h for each node u ∈ V;
```

(lines 1-2). Then, the nodes are sorted in a non-decreasing order of their colorful h-star degrees (line 3). Next, the algorithm removes the node with the minimum colorful h-star degree (lines 5-13). Note that in each iteration, we assign the current updated max_core to the colorful h-star core number of u (lines 7-9). After that, we update the colorful h-star degrees of neighbor nodes of u by invoking the proposed Updating algorithm (lines 10-11). Algorithm 3 deletes u from H and re-sorts the nodes of the remaining graph (lines 12-13). Finally, we return csc_u^h for each node $v \in V$ (line 14). Obviously, the The colorful h-star core number can be derived from the final max_core. Below, we analyze the time and space complexity of Algorithm 3.

Theorem 4: Given a graph G and an integer h, Algorithm 3 computes the colorful h-star core decomposition in $O(hm)$ time and $O(hn + m)$ space.

Proof: The graph G can be colored in linear time by using a greedy coloring algorithm [12], [13]. For each node u, we need to compute colorful h-star degree of u, i.e., $d_u^G(\star^h)$, which takes $O(h\chi)$ time, where χ is the maximum color value of all nodes in G. Note that, since we only need to consider the neighbors of u when computing the colorful h-star degree of u, the time overhead of the DP algorithm can be further reduced to $O(h \min\{d_u, \chi\})$. Thus, the total time overhead for calculating the colorful h-star degrees for all nodes is bounded by $O(hm)$. Similar to the linear-time core decomposition algorithm, we can employ a bucket sort technique [24] to sort and re-sort nodes of H, which takes linear time. When removing a node u, the time overhead for updating the colorful h-star degree of a neighbor v is $O(h)$, thus the total time to update the colorful h-star degrees for all u's neighbors can be bounded by $O(d_u h)$. Since each node is deleted once, the total updating time of the algorithm is $O(hm)$. For the space complexity, the algorithm takes $O(h + \min\{\chi, d_u\})$ space to compute and update the colorful h-star degree of a node u, thus the total space complexity of our algorithm is $O(hn + m)$. □

Note that when $h = 2$, a colorful 2-star is exactly an edge, and $d_u^G(\star^h) = d_u^G$. In this case, the colorful 2-star core decomposition turns into the classical core decomposition. The

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

Algorithm 4: The Colorful h-Star Truss Decomposition.

Input: A graph $G = (V, E)$ and an integer h
Output: the h-colorful k-truss classes $\overline{\mathsf{CST}_k^h}$ for $k \in [2 : k_{max}]$
1 Initialize $\mathcal{C}(v), \chi, \mathsf{cnt}$ as lines 1-5 in Algorithm.1;
2 **for** *each* $e_{uv} \in E$ **do**
3 $sup^h(G, e_{uv}) \leftarrow$ counting the (h-2)-colorful node set in $N(u) \cap N(v)$; // a variant of Counting+ in Algorithm 2
4 $k \leftarrow 2; R \leftarrow E; CST \leftarrow \emptyset$;
5 **while** $R \neq \emptyset$ **do**
6 $Q \leftarrow \emptyset; CST[k] \leftarrow \emptyset$;
7 **for** *each* $e_{uv} \in R$ **do**
8 **if** $sup^h(G, e_{uv}) \leq k$ **then**
9 $Q.pop(e_{uv})$;
10 **while** $Q \neq \emptyset$ **do**
11 $e_{uv} \leftarrow Q.push(); CST[k].add(e_{uv}); R.rm(e_{uv})$;
12 **for** $(u, w) \in R$ **do**
13 $sup^h(G, e_{uw}) \leftarrow$ updating by removing node v // a variant of Updating in Algorithm 2
14 **if** $sup^h(G, e_{uw}) < k$ **then** $Q.push(e_{uw})$;
15 **for** $(v, w) \in R$ **do**
16 $sup^h(G, e_{vw}) \leftarrow$ updating by removing u;
17 **if** $sup^h(G, e_{vw}) < k$ **then** $Q.push(e_{vw})$;
18 Output $CST[k]$ to be an h-colorful k-truss class;
19 $k \leftarrow k + 1$;

core decomposition takes $O(m)$ time using $O(m + n)$, which is consistent with our results.

2) The Colorful H-Star Truss Model: A k-truss is the maximal subgraph where every node each edge is contained in at least k-2 triangles [25]. A triangle $\triangle_{uvw}$ is a 3-length cycle which contains the edges $(u, v), (v, w)$ and (v, w). The support of an edge $e_{uv} = (u, v)$ in graph $G = \{V, E\}$ is the number of triangles containing e_{uv}, i.e., $sup(G, e_{uv}) = |\{\triangle_{uvw} | w \in V\}|$. Therefore, a k-truss in G, denoted by T_k, is a maximal subgraph H, such that $\forall e_{uv} \in E, sup(H, e_{uv}) \geq k - 2$. Besides, the k-truss number of the edge e is the maximal truss number such that there is a k-truss containing e.

Inspired by the definition of the k-truss, we define the colorful support of the edge e_{uv} and the h-star k-truss as follows:

Definition 5 (h-colorful support): Given a colored graph G=$(V, E, \mathcal{C})$ and parameter $h \geq 3$, the h-colorful support of an edge e_{uv}=(u, v) in G, i.e., $sup^h(G, e_{uv})$, is number of the (h-2)-colorful node sets in $N(u) \cap N(v)$.

Note that, a h-colorful node set is a set of h nodes in which any pair of nodes has different colors (defined in Section II).

Definition 6 (Colorful h-star k-truss): Given a colored graph G, parameters h and k, a colorful h-star k-truss, denoted by $\mathsf{CST}_k(\star^h)$ (abbr. CST_k^h), is a maximal subgraph such that $\forall e_{uv} \in E_{\mathsf{CST}_k^h}, sup^h(\mathsf{CST}_k^h, e_{uv}) \geq k$.

The h-colorful truss number of an edge $e \in E_H$, denoted by $cst_e(\star^h)$ (abbr. cst_e^h), which equals $\max\{k | e \in \mathsf{CST}_k^h\}$. It follows that given $cst_e^h = k$, $e \in \mathsf{CST}_k^h$ but $e \notin \mathsf{CST}_{k+1}^h$. Based on the colorful truss number, the colorful h-star k-truss class $\overline{\mathsf{CST}_k^h}$ is a maximal set of edges in which the colorful truss number of each equals k, i.e., $\overline{\mathsf{CST}_k^h} = \{e \in E_G | cst_e^h = k\}$.

Based on Definition 6, we define the colorful h-star truss decomposition problem as follows.

Problem 3 (Colorful h-star Truss Decomposition): Given a colored graph G and an integer h, the goal of colorful h-star truss decomposition is to compute all the h-colorful k-truss classes, i.e., $\overline{\mathsf{CST}_k^h}$, for $2 \leq k \leq k_{max}$.

The challenge of mining a colorful h-star k-truss is more complex compared to the colorful h-star k-core due to the stricter requirements for edge support. Both problems deal with the difficulty of counting colorful h-stars, but while the k-core focuses on vertex degrees, the k-truss requires each edge to be supported by at least $k - 2$ colorful h-stars, making it a denser and more computationally intensive task. In the k-truss, however, the focus shifts to edge support, which requires counting the number of triangles that each edge forms, adding complexity to both the counting and maintenance of the subgraph structure. Additionally, truss decomposition, which involves removing edges based on insufficient support, is a more complex operation than the vertex removal in the core. Thus, while both problems face challenges related to counting and dynamic updates, the k-truss presents a greater difficulty due to the need to maintain denser subgraph structures and ensure adequate edge support in colorful h-stars.

The method of truss decomposition in question is also based on a bottom-up approach. It is evident that a $\overline{\mathsf{CST}_{k+1}^h}$ must necessarily be a $\overline{\mathsf{CST}_k^h}$. Therefore, we can attempt to find a $\overline{\mathsf{CST}_{k+1}^h}$ within a $\overline{\mathsf{CST}_k^h}$ by deleting certain edges. The pseudocode for this algorithm is elaborated in Algorithm 4.

Algorithm 4 begins by initializing $\mathcal{C}(v)$, χ, and cnt as outlined in lines 1-5 of Algorithm 1 (line 1). Subsequently, the algorithm initializes the computation of the h-colorful support values for all edges (lines 2-3), which can be performed using a dynamic programming algorithm similar to Algorithm 2. Next, the algorithm creates a set R to store the collection of edges under consideration, as well as a set CST to store all the h-colorful k-truss classes (lines 4). If the set R is not empty, then all nodes in R are placed into the queue Q if the h-colorful support $sup^h(G, e_{uv}) \leq k$ (lines 6-9). The algorithm then proceeds to consider each edge e_{uv} in the queue Q one by one. For each edge that includes node u or v, the h-colorful support values are updated (lines 12-17). Note that this update procedure can be obtained using a Updating procedure akin to Algorithm 2. For a given k, once the queue Q is empty, it signifies that the remaining edges have an h-colorful support of k or greater. Therefore, these remaining $CST[k]$ are output as an colorful h-star k-truss class (line 18). The value of k is then incremented by 1, and the loop continues to find all colorful h-star k-truss classes.

Example 5: Fig. 6 depicts the colorful 3-star k-trusses of the graph in Fig. 1(a). We can see that the whole graph with all the edges are colorful 3-star 0-truss since the edges (v_5, v_8) and (v_8, v_9) have 3-colorful support of 0 since $N(v_5) \cap N(v_5)$ and $N(v_8) \cap N(v_9)$ are $\emptyset$. After peeling the edges (v_5, v_8) and (v_8, v_9), the truss number of the edges in the remained graph is 1 since all the edges are contained in at least one triangle. Next, after removing the edges (v_6, v_9) and (v_7, v_9), the remained graph is a CST_k^2. Finally, the graph induced by $\{v_3, v_4, v_5, v_6, v_7\}$ is a CST_k^3 since the 3-colorful support of each edge in this subgraph is 3.

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

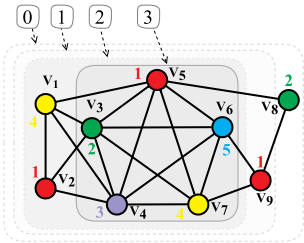


Fig. 6. colorful 3-star k-truss(CST_k^3).

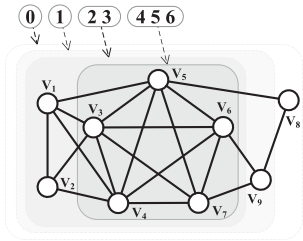


Fig. 7. h-clique k-core(CC_k^h).

Theorem 5: Given a graph G and an integer h, Algorithm 4 computes the colorful h-star truss decomposition in $O(hm^{1.5})$ time and $O(hm)$ space.

Proof: Consistent with Algorithms 3 and 4 initially colors the graph in linear time by employing a greedy coloring algorithm. Subsequently, for each edge within the colored graph, the algorithm calculates the h-colorful support for that edge. To accomplish this, for each edge e_{uv}, it is necessary to determine $N(u) \cap N(v)$, which requires $O(m \times \max(N(u) \cap N(v)))$. Alternatively, if we consider enumerating all triangles within the graph, the $N(u) \cap N(v)$ for each edge e_{uv} can be naturally obtained. The time complexity for enumerating triangles is $O(m^{1.5})$ as referenced in [25]. Once $N(u) \cap N(v)$ is ascertained, according to Algorithm 2, computing the h-colorful support for these m edges demands $O(mh)$. Therefore, the time complexity for lines 1-3 is $O(mh + m^{1.5})$. Moving forward, each removal of an edge necessitates an update of the h-colorful support for the endpoints of the edge, which takes $O(h)$ time (lines 13,16). The number of such updates is proportional to the number of triangles in the graph. Consequently, updating all edges requires time complexity of $O(hm^{1.5})$. In summary, the total time complexity of Algorithm 4 is $O(hm^{1.5})$. Regarding space complexity, the algorithm allocates $O(hm)$ space for the computation and update of the h-colorful support of each edge, hence the overall space complexity of the algorithm is $O(hm)$. □

B. Speeding up H-Clique Densest Subgraph Mining

In this section, we show that our colorful h-star core decomposition technique can be used to speed up the state-of-the-art approximate algorithm for mining h-clique densest subgraph. Below, we first briefly review this problem. Note that existing algorithms for the h-clique densest subgraph problem can be classified in two categories: exact algorithms and approximation algorithms. Even though the exact algorithms based on the max-flow technique can solve this problem in polynomial time for small h values [15], this problem can be NP-hard for a large h. Because the number of h-cliques in a graph is exponential in the size h, counting h-clique is often deemed infeasible when h is large. Thus, we focus mainly on the approximation algorithms, and discuss two state-of-the-art approximation algorithms (Section IV-B2) to solve this problem. We propose an efficient graph reduction technique, based on the colorful h-star core decomposition, to significantly prune the unnecessary nodes from the given graph without sacrificing approximation performance (Section IV-B3).

1) The H-Clique Densest Subgraph: Given a graph G, an h-clique ($\boxtimes^h$) is a subgraph with h nodes such that each pair of nodes is connected with an edge. The h-clique number of G is the number of h-cliques in G. We also denote $d_u^G(\boxtimes^h)$ to the number of h-cliques that u participates in.

Definition 7 (: h-CLIQUE DENSITY) Given a graph G and an integer h, for any induced subgraph H, $V_H \subseteq V_G$, its h-clique density is defined as $\sigma_h(H) = \frac{\sum_{u \in H} d_u^G(\boxtimes^h)}{|V_H|}$.

Problem 4 (: H-CLIQUE-DS-PROBLEM) Given a graph G and an integer h, the problem is to find a subgraph H^* that achieves the largest h-clique density among all subgraphs of G, and let $\sigma_h(H^*)$ be the density of the h-Clique densest subgraph.

Fang et al. [14] introduced a concept of h-clique core, which can help achieve a good approximation to the H-CLIQUE-DS-PROBLEM (abbreviated as h-CDS).

Definition 8 (: h-CLIQUE k-CORE [14]) Given a graph G, parameters k and h, the h-clique k-core, i.e., $\mathsf{CC}_k(\boxtimes^h)$ (abbr. CC_k^h), is a maximal subgraph in G, such that $\forall u \in \mathsf{CC}_k^h$, $d_u(\mathsf{CC}_k^h, \boxtimes^h) \geq k$.

We denote the h-clique core number of a node $u \in V$ by $cc_u(\boxtimes^h)$ (abbr. cc_u^h), which is the largest k such that there exists an h-clique k-core containing u. The maximum h-clique core number of a graph G, denoted by $\hat{cc}$, is the maximum value of h-clique core numbers among all nodes.

Example 6: Fig. 7 illustrates the h-clique k-cores of the graph. Each rectangle displays the number k, representing the CC_k^3 contained within it. For instance, the subgraph induced by $\{v_3, v_4, v_5, v_6, v_7\}$ corresponds to the CC_6^3. In this case, every node in the 5-clique participates in six 3-cliques. Furthermore, $cc_{v_1}^3 = 3$ as v_1 is part of the CC_3^3 and is not a member of CC_4^3.

The following theorem shows that the CC_k^h is a good approximation of the h-clique densest subgraph [14].

Theorem 6 (Approximation Solution [14]): Given a graph G and an h-clique, the $\mathsf{CC}_{\hat{cc}}^h$ is a $\frac{1}{h}$-approximation solution to h-clique densest subgraph problem, such that $\sigma_h(\mathsf{CC}_{\hat{cc}}^h) \geq \frac{\sigma_h(H^*)}{h}$.

2) Existing Approximation Algorithms: Core-based approximation algorithm: Fang et al. established lower and upper bounds on the h-clique density for each CSC_k^h [14]. Based on these bounds, they computed the upper and lower bounds of the density of the h-Clique densest subgraph. They also found that the h-clique densest subgraph is located in some specific

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

CSC_k^h, and the $\mathsf{CC}_{\hat{c}c}^h$ is a $\frac{1}{h}$-approximation solution. Therefore, to obtain the $\mathsf{CC}_{\hat{c}c}^h$, unlike the straightforward method which greedily peels the node with the minimum h-clique degree, they proposed the CoreApp algorithm which focuses on computing the $\mathsf{CC}_{\hat{c}c}^h$ directly based on the observation that nodes in $\mathsf{CC}_{\hat{c}c}^h$ tend to have higher h-clique degrees. The main defect of CoreApp is that it needs to compute the $\mathsf{CC}_{\hat{c}c}^h$ on the entire graph and doubles the size of candidate nodes set between different iterations from $V(G)$, which is very costly. Since a large number of nodes actually contribute nothing to the $\mathsf{CC}_{\hat{c}c}^h$, considering those unpromising nodes in the computational procedure causes much unnecessary time consumption.

Sampling-based approximation algorithm: Sun et al. proposed a sampling algorithm, called SeqSamp, which can obtain an approximation of the h-clique densest subgraph [17]. The general idea of the SeqSamp algorithm is as follows. First, the algorithm maintains a variable $r(u)$ for each node u, and assigns every h-clique to the node v with the minimum r value among the nodes in the h-clique and increases $r(v)$ by 1. Then, SeqSamp sorts nodes of V in an increasing order according to their r values. After that SeqSamp computes the h-clique density of the subgraph induced by the first s nodes for each $s \in [n]$, and returns the subgraph which achieves maximum h-clique density among all n subgraphs. To save memory usage, the algorithm stores each h-clique into main memory independently with probability $p = \frac{\sigma}{\sum_{u \in H} d_u^G(\boxtimes^h)}$, where σ is a parameter which represents the approximate number of h-cliques to be sampled. The main limitation of SeqSamp is that it suffers from a loose upper bound even after a large number of iterations. Thus, to obtain a good approximation, such an algorithm often needs a long time as confirmed in our experiments.

However, SeqSamp is not scalable for large k values and large-scale graphs because it requires repeated enumeration of all k-cliques from the graph in each iteration. To address this issue, He et al. [26] propose SCTL, which builds an index structure to expedite the k-clique enumeration process by leveraging the succinct clique tree from pivoter. SCTL constructs a tree that maintains a unique representation of all k-cliques, occupying significantly less space than storing all k-cliques directly, thus allowing the tree to remain in memory. With the help of the succinct clique tree and various optimization techniques, SCTL achieves greater efficiency compared to SeqSamp. Nonetheless, SCTL still requires enumerating all k-cliques in the worst case to update vertex weights during each iteration, making its running time proportional to the number of k-cliques present in the graph.

Additionally, Fang et al. [27] propose a SOTA approach by employing a simple yet effective Frank-Wolfe-based framework that utilizes k-clique counting instead of k-clique enumeration. The Frank-Wolfe algorithm operates iteratively, initially assigning a weight $r(v)$ to every vertex v in the graph, with this weight set to the number of k-cliques containing v divided by k. In each iteration, the algorithm identifies the vertex v with the minimum weight for every k-clique and updates its weight $r(v)$. They discover that the change in $r(v)$ for the vertex with the minimum weight can be determined using the number of k-cliques that include v. Building on this observation, they develop a novel framework based on a k-clique counting algorithm. By integrating these techniques, they create an efficient $(1+\epsilon)$ k-clique counting-based approximation (KCCA) algorithm, where $\epsilon > 0$.

3) The Colorful h-Star Core Based Algorithm: The $\mathsf{CC}_{\hat{c}c}^h$ achieves a $\frac{1}{h}$-approximation to the H-CLIQUE-DS-PROBLEM, but computing the $\mathsf{CC}_{\hat{c}c}^h$ on the graph is a time-consuming task. In this subsection, we present an effective pruning strategy to reduce the graph G based on our colorful h-star core model without sacrificing accuracy. Note that our graph reduction technique can be considered as a preprocessing approach which can be used to speed up the computation of $\mathsf{CC}_{\hat{c}c}^h$ based approximate h-clique densest subgraph algorithm.

Observation: Assume that there is a clique of size w in G, and then the w-clique must be contained in an h-clique θ-core, here $\theta = \binom{w\text{-}1}{h\text{-}1}$. This is because each node in this w-clique participates in at least θ h-cliques in G, which provides a nontrivial lower bound for the maximum h-clique core number $\geq \theta$. Obviously, the larger a clique we can obtain is, the tighter our lower bound will be. Thus, our goal is to find a clique as large as possible. Since finding the maximum clique is NP-hard, we can use a linear-time greedy maximum clique algorithm proposed by Rossi et al. [28] to find a large clique.

The basic pruning rule: Suppose that we have a large clique with size w computed by the greedy algorithm [28]. Then, we have the following result.

Theorem 7: Given a graph G, and an h-clique, the h-clique θ-core CC_θ^h is contained in the $(w\text{-}1)$-core, where w is the size of a large clique of G and $\theta = \binom{w\text{-}1}{h\text{-}1}$.

Proof: For each node $v \in \mathsf{CC}_\theta^h$, the following inequalities hold that $\binom{w\text{-}1}{h\text{-}1} \leq d_v^{\mathsf{CC}_\theta^h}(\boxtimes^h) \leq d_v^{\mathsf{CC}_\theta^h}(h\text{-}star) = \binom{d_v^{\mathsf{CC}_\theta^h}}{h\text{-}1}$.

It provides a lower bound $d_v^{\mathsf{CC}_\theta^h} \geq w\text{-}1$ of the degree, indicating that CC_θ^h must be contained in the $(w\text{-}1)$-core. □

The above analysis inspires us to first reduce the input graph G to a $(w\text{-}1)$-core based on the fact that the $\mathsf{CC}_{\hat{c}c}^h$ is a subgraph of the $(w\text{-}1)$-core. Although shrinking the graph G to a $(w\text{-}1)$-core eliminates a number of nodes with core number less than w-1, the remaining graph may still be very large, because the $(w\text{-}1)$-core is often far from the final h-clique densest subgraph. Therefore, we strive to seek a more effective reduction technique on G to further prune the $(w\text{-}1)$-core. We can achieve this by applying our colorful h-star core decomposition to further remove the unnecessary nodes for computing the $\mathsf{CC}_{\hat{c}c}^h$.

The advanced pruning rule: Reconsider the colored graph G and the h-clique θ-core CC_θ^h. Clearly, each node of CC_θ^h participates in at least θ cliques, thus it is also contained in at least θ colorful h-stars. This is because nodes in a clique must have different colors, suggesting that CC_θ^h must be contained in the colorful h-star θ-core CSC_θ^h. Therefore, we can use the colorful h-star θ-core for pruning unpromising nodes. Moreover, the following theorem shows that the colorful h-star θ-core pruning technique is more effective than the $(w\text{-}1)$-core pruning strategy.

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

Algorithm 5: h-CDS by Colorful h-Star Core Reduction.

Input: A graph G and an integer h
Output: The $(\theta, \star^h)$-core
1 $\boxtimes^h \leftarrow$ compute a large clique using a greedy algorithm proposed in [28] ;
2 $\omega \leftarrow |V_{\boxtimes^h}|, \theta \leftarrow \binom{\omega-1}{h-1}$;
3 $\mathsf{C}_{w\text{-}1} \leftarrow$ compute the $(\omega$-1)-core using the peeling algorithm in [23];
4 $\mathsf{CSC}_k^h \leftarrow \mathsf{ColorfulStarCore}(\mathsf{C}_{w\text{-}1}, h, \theta)$;
5 **return** CSC_k^h ;
6 **Procedure** ColorfulStarCore (H, h, k)
7 **for** $u = 1$ **to** $|V_H|$ **do**
8 $\quad d_u^H(\star^h) \leftarrow \mathsf{Counting}(u)$;
9 Let Q be an empty queue;
10 **for** *each* $v \in H$ **do**
11 $\quad$ **if** $d_u^H(\star^h) < k$ **then**
12 $\quad\quad$ Push v to Q;
13 **while** $Q \neq \emptyset$ **do**
14 $\quad$ Pop a node u from Q;
15 $\quad$ **for** *each* $v \in N_u^H$ **do**
16 $\quad\quad d_v^{H\setminus u}(\star^h) \leftarrow \mathsf{Updating}(\mathcal{DP}, v)$;
17 $\quad\quad$ **if** $d_v^{H\setminus u}(\star^h) < k$ **then**
18 $\quad\quad\quad$ Push v to Q;
19 $\quad$ Delete u from H;
20 **return** H;

Theorem 8: Given a graph G, and an integer h, the colorful h-star θ-core CSC_θ^h is contained in the $(w$-1)-core of G, where w is the size of a large clique of G, and $\theta = \binom{w\text{-}1}{h\text{-}1}$.

Proof: Similar to the proof of Theorem 7, for each node $v \in \mathsf{CSC}_\theta^h$, the following inequalities hold that $\binom{w\text{-}1}{h\text{-}1} \leq d_v^{\mathsf{CSC}_\theta^h}(\star^h) \leq d_v^{\mathsf{CSC}_\theta^h}(h\text{-}star) = \binom{d_v^{\mathsf{CSC}_\theta^h}}{h\text{-}1}$.

As a consequence, we have $d_v^{\mathsf{CSC}_\theta^h} \geq w$-1, thus CSC_θ^h is an induced subgraph of $\mathsf{C}_{w\text{-}1}$. □

Based on the above analysis, to approximate the h-clique densest subgraph, we can progressively eliminate the unpromising nodes and shrink the input graph G to a smaller and smaller subgraph based on the following core-reduction order

$$G \supseteq \mathsf{C}_{w\text{-}1} \supseteq \mathsf{CSC}_\theta^h \supseteq \mathsf{CC}_{\hat{c}c}^h.$$

Our colorful h-star core based graph reduction technique is shown in Algorithm 5. Algorithm 5 first applies the $(w$-1)-core reduction (lines 1-3), and then performs the colorful h-star core pruning on the $(w$-1)-core (lines 4-20). Note that the algorithm uses a similar peeling procedure to iteratively remove the nodes with colorful h-star degree smaller than θ to compute the colorful h-star θ-core (lines 6-20). It is easy to show that the time complexity of Algorithm 5 is $O(hm)$. This is because the lines 1-3 takes $O(m)$ time, and the computing of the colorful h-star θ-core uses $O(hm)$ time.

A heuristic approach: Since a colorful h-star is a good relaxation of an h-clique, the colorful h-star core can also be used to approximate the h-clique core. The h-clique $\hat{c}c$-core has been proved to be a $\frac{1}{h}$-approximation of H-CLIQUE-DS-PROBLEM.

TABLE II
DATASETS

Dataset	$n = \|V\|$	$m = \|E\|$	χ	$d_{\max}$
Nasasrb	54,870	1,311,227	38	275
Pkustk	87,804	2,565,054	54	131
Buzznet	101,163	2,763,066	62	64,289
Pwtk	217,891	5,653,221	42	179
DBLP	317,080	1,049,866	114	343
MsDoor	404,785	9,378,650	42	76
Digg	770,799	5,907,132	66	17,643
LDoor	909,537	20,770,807	42	76
Skitter	1,694,616	11,094,209	71	35,455
Orkut	2,997,166	106,349,209	79	27,466
LiveJournal	4,847,572	42,851,237	324	20,333

Therefore, the colorful h-star $c\hat{s}c$-core should also provide a good approximation solution. That is to say, we can obtain a heuristic algorithm by just computing the the colorful h-star $c\hat{s}c$-core on G using Algorithm 3 as an approximation of the h-clique densest subgraph (as defined in Section IV-A1, $c\hat{s}c$ is the maximum value of colorful h-star core numbers among all nodes).

V. EXPERIMENTS

A. Experimental Setup

Datasets: We collect 11 large real-world graphs from two different sources, including (1) Stanford Network Analysis Project (SNAP) (http://snap.stanford.edu/data/), and (2) Network Repository (https://networkrepository.com/). These datasets cover various domains, such as online social networks (e.g., Buzznet, Digg, Orkut and LiveJournal), collaboration networks (e.g., DBLP), internet topology graphs (e.g., Skitter) and scientific computing networks (e.g., Nasasrb, Pkustk, Pwtk, MsDoor and LDoor). The detailed statistics of the datasets are summarized in Table II. In Table II, χ and $d_{\max}$ denote the number of colors obtained by a greedy coloring algorithm and the maximum degree of the graph respectively.

Algorithms: Due to the efficient enumeration and updating capabilities of colorful h-stars through the use of a novel dynamic programming (DP) algorithm, we demonstrate the advantages of DP, particularly the update algorithm, by applying it to the decomposition of colorful h-star core and colorful h-star truss.

For the colorful h-star core and truss decomposition, we have incorporated two baselines: k-core and k-truss, to serve as reference tests. Additionally, we implemented four algorithms proposed in this manuscript:

- KCore [29] serves as a baseline that outputs the traditional k-core. We implement it using a SOTA peeling algorithm based on bin-sorting, which has a time complexity of $O(m)$.
- CSC-DP is an algorithm to compute colorful h-star core which recomputes the colorful h-star degrees of its neighbors using the proposed DP algorithm after removing a node.

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

- CSC-DP+ is plus version of CSC-DP which uses the proposed updating technique to update the colorful h-star degrees.
- KTruss [20] serves as a baseline that outputs the traditional k-truss. We implement it using a SOTA peeling algorithm, which has a time complexity of $O(m^{1.5})$.
- CST-DP is an algorithm to compute colorful h-star truss which recomputes the h-colorful support the edges using the proposed DP algorithm.
- CST-DP+ is plus version of CST-DP which uses a updating technique to update the h-colorful support of the edges.

For the h-clique densest subgraph problem, we implement our two algorithms:

- h-CDS first performs our colorful h-star core based pruning on G, and then calls CoreApp [14] on the reduced graph.
- h-CDS+ is the heuristic approach using the colorful h-star $c\hat{s}c$-core as an approximation solution.

and four state-of-the-art algorithms:

- CoreApp [14] is a h-clique core based approximation algorithm which can obtain a $1/h$ approximation (we slightly modify the origin method with some graph reduction to be more efficient).
- Exact is an exact solution [17] using the max-flow technique.
- SeqSamp [17] is a version of the sampling algorithm SeqSamp with the number of iterations $T = 1$.
- SeqSamp128 [17] is a version of SeqSamp with the number of iterations $T = 128$.
- SCTL [26] is an approximation algorithm that counts the number of k-cliques in the approximate solution by selectively traversing an index with the number of iterations $T = 1$
- KCCA [27] is an approximation algorithm that based on an efficient $(1 + \epsilon)$ k-clique counting-based approach with $T = 1$.

Parameters: We have only one parameter h in our experiments. Unless otherwise specified, we evaluate all algorithms with a varying h from 3 to 9. In all our experiments, we set time limit to 24 hours for each algorithm, and "INF" for the running time of any algorithm which exceeds 24 hours. We implement all our algorithms in C++. All experiments are conducted on a Linux machine equipped with a 2.9 GHz AMD Ryzen 3990X CPU and 256 GB RAM running CentOS 7.9.2 (64-bit).

B. Results of the Colorful h-Star k-Core/Truss Problem

Exp-1: Runtime of different algorithms: Fig. 8 presents the running time results for KCore, CSC-DP+, CSC-DP, Ktruss, CST-DP, and CST-DP+ across different datasets for h = 6. We also tested other values of h and found that the results remain consistent. It is evident that CSC-DP+ is 2 to 7 times faster than CSC-DP across all datasets. For instance, on the *livejournal* dataset, the total running time of CSC-DP is 115.1 seconds, whereas CSC-DP+, enhanced by the proposed updating technique, only takes 28.2 seconds, making it approximately 4 times faster than CSC-DP. However, both CSC-DP+ and

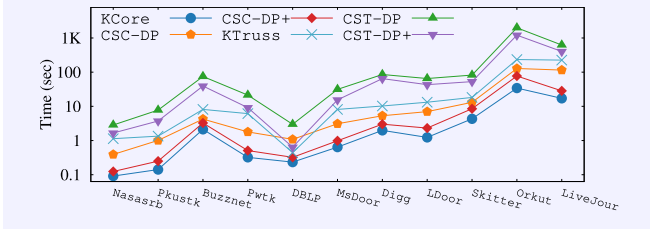
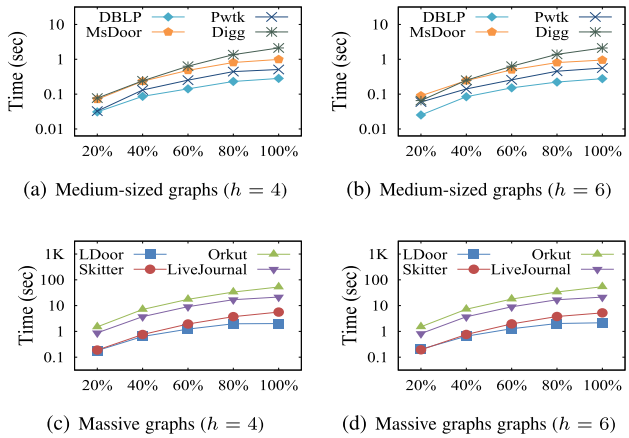


Fig. 8. Running time of KCore, CSC-DP+, CSC-DP, Ktruss, CST-DP, and CST-DP+ ($h = 6$).



(a) Medium-sized graphs ($h = 4$) (b) Medium-sized graphs ($h = 6$)

(c) Massive graphs ($h = 4$) (d) Massive graphs graphs ($h = 6$)

Fig. 9. Scalability of CSC-DP+.

CSC-DP perform slightly worse than KCore, as KCore does not consider higher-order attributes, making it the fastest baseline. Similarly, CST-DP+ is 2 to 3 times faster than CST-DP across all datasets. The difference between CST-DP+ and CST-DP is smaller compared to CSC-DP+ and CSC-DP, primarily because truss-based algorithms require more time for peeling. Compared to the baseline KTruss, the performance gap of CST-DP+ follows a linear pattern, keeping it within an acceptable range for practical applications. Additionally, it can be observed that CST-DP and CST-DP+ are slower than CSC-DP+ and CSC-DP. On the LiveJournal dataset, CST-DP requires 403.21 seconds, which is slower than CSC-DP+. This is because CST-DP theoretically requires $O(hm^{1.5})$ time, whereas CST-DP only requires $O(hm)$ time. Moreover, across all datasets, CST-DP is consistently slower than CST-DP+. These results confirm our theoretical analysis shown in Sections III and IV.

Exp-2: Scalability: Fig. 9 shows the scalability of CSC-DP+ on 8 different datasets. Due to the fact that CST-DP+ employs a similar DP update algorithm, its scalability results are analogous to those of CSC-DP+. Owing to space constraints, the scalability tests for the CST-DP+ algorithm are omitted here. Those 8 datasets can be divided into two sets according to their number of edges: four medium-sized graphs, which are DBLP, Pwtk, Digg, and MsDoor and four massive graphs, including LDoor, Orkut, Skitter, and LiveJournal. For each dataset, we generate four subgraphs by randomly sampling nodes from 20% to 100%. Fig. 9 shows the results of $h = 4$ and $h = 6$, and similar results can also be observed with other parameters. As can be seen, the

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

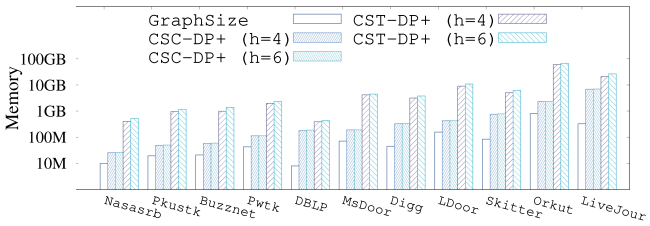


Fig. 10. Memory overhead of algorithms on different datasets ($h = 4, 6$).

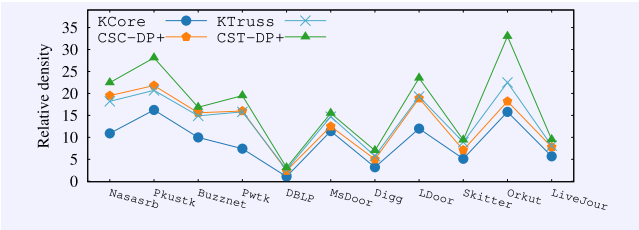


Fig. 11. Effectiveness of KCore, CSC-DP+, Ktruss, CST-DP+ ($h = 6$).

running time of CSC-DP+ increases smoothly as the graph size increases on both medium-sized and massive graphs. Moreover, all the curves are nearly linear, indicating that our algorithm scales very well in practice. These results also confirm the near-linear time complexity of our CSC-DP+ algorithm (i.e., its complexity is $O(hm)$).

Exp-3: Memory overhead: Fig. 10 shows the memory usage of our CSC-DP+ and CST-DP+ algorithms on various datasets for $h = 4$ and $h = 6$ respectively. The results are shown in Fig. 10. Similar results can also be observed for the other values of h. From Fig. 10, we can see that the memory cost of CSC-DP+ is insensitive w.r.t. h on all datasets. This is because the space complexity of CSC-DP+ is $O(hn + m)$, and $O(hn)$ in the space complexity is usually much smaller than $O(m)$ (see Table II) on real-world graphs. For the graphs with a relatively large color number, such as DBLP and LiveJournal, the space consumption of CSC-DP+ is around 10 times higher than the graph size, while for the other datasets, the space usage of CSC-DP+ is only slightly larger than the graph size. Additionally, it is noticeable that the space usage of CST-DP+ increases modestly with the growth of h. This is despite the space complexity of CST-DP+ being $O(hm)$. As h increases, the number of $(h-2)$-colorful node sets within the intersection of $N(u)$ and $N(v)$ for edge e_{uv} is reduced. Although the theoretical space complexity indicates a linear increase, in reality, the h-colorful support for many edges diminishes as h rises. Therefore, the spatial scalability of CST-DP+ remains quite good.

Exp-4: Effectiveness of different models: Fig. 10 and 11 presents the relative density of the KCore, CSC-DP+, Ktruss, and CST-DP+ models when h = 6. Since the results for CSC-DP+ and CSC-DP, as well as for CST-DP+ and CST-DP, are consistent, we omit the results for CST-DP and CSC-DP in this analysis. It is evident that the CST-DP+ model exhibits the highest density, as it utilizes stars for truss clustering, resulting in a higher degree of aggregation by definition. Generally, truss models tend to have higher density compared to core models. However, CSC-DP+ shows higher density than Ktruss in some datasets (for example, the Nasasrb and Pkustk datasets). It indicates that higher-order models typically demonstrate greater density than traditional models. Overall, these results confirm that our proposed colorful star-based higher-order clustering models are indeed more effective.

C. Results of the h-Clique-DS-Problem

In this subsection, we carry out a set of experiments to evaluate the performance of 8 different algorithms for solving the h-Clique-DS-Problem.

Exp-5: Running time of different algorithms for h-Clique-DS-Problem . Fig. 12 shows the running time achieved by the eight competing algorithms as h varies from 3 to 9 across the larger eight datasets. We observe the following results: **(i)** As expected, the running time of all eight approximation algorithms increases as h increases. This is because the number of h-cliques grows as h increases. We can clearly see that both the colorful h-star core-based algorithm (h-CDS) and the heuristic approach (h-CDS+) are significantly faster than the other competitors, with Exact taking the longest time on almost all datasets. There is one exception with DBLP, where the SeqSamp128 algorithm, after 128 iterations, takes more time for smaller k values compared to the exact algorithm (which, to be fair, also uses a powerful pruning method and is not so naive). This shows that, in extreme cases, when sampling algorithms use too many iterations, they can consume more time than exact algorithms. **(ii)** h-CDS+ outperforms h-CDS on the majority of datasets. h-CDS+ is slightly faster than all other algorithms (especially on LDoor and Orkut), but a little slower than h-CDS on LiveJournal with $h = 3, 4$. The reason is that h-CDS+ needs to compute the entire colorful h-star core decomposition, so for smaller h values, h-CDS may be slightly faster. On LDoor, CoreApp is already very fast for a small h, so in this case, it cannot be significantly accelerated by our algorithms. **(iii)** Among the compared state-of-the-art methods, the latest KCCA is the best, but our h-CDS+ algorithm outperforms KCCA across all datasets. When comparing SeqSamp, SCTL, and KCCA after just one iteration, their performance is similar most of the time, though KCCA tends to be more efficient. Despite the high efficiency of KCCA after one iteration, h-CDS+ outperforms KCCA across all datasets. This is because our algorithm only requires linear time to compute the colorful core, validating our theoretical analysis. Besides, on some datasets, SeqSamp, SeqSamp128 and Exact fail to terminate within 24 hours for large h values. The reason could be that the sampling based algorithms involve a time-consuming procedure which runs a clique-counting subroutine twice to first count the number of h-cliques and then sample the h-cliques to be stored into memory. The exact solution runs a more time-consuming max-flow algorithm, thus they can not deal with large h values. These results confirm that the proposed technique is very effective in speeding up the approximate h-clique densest subgraph computation.

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

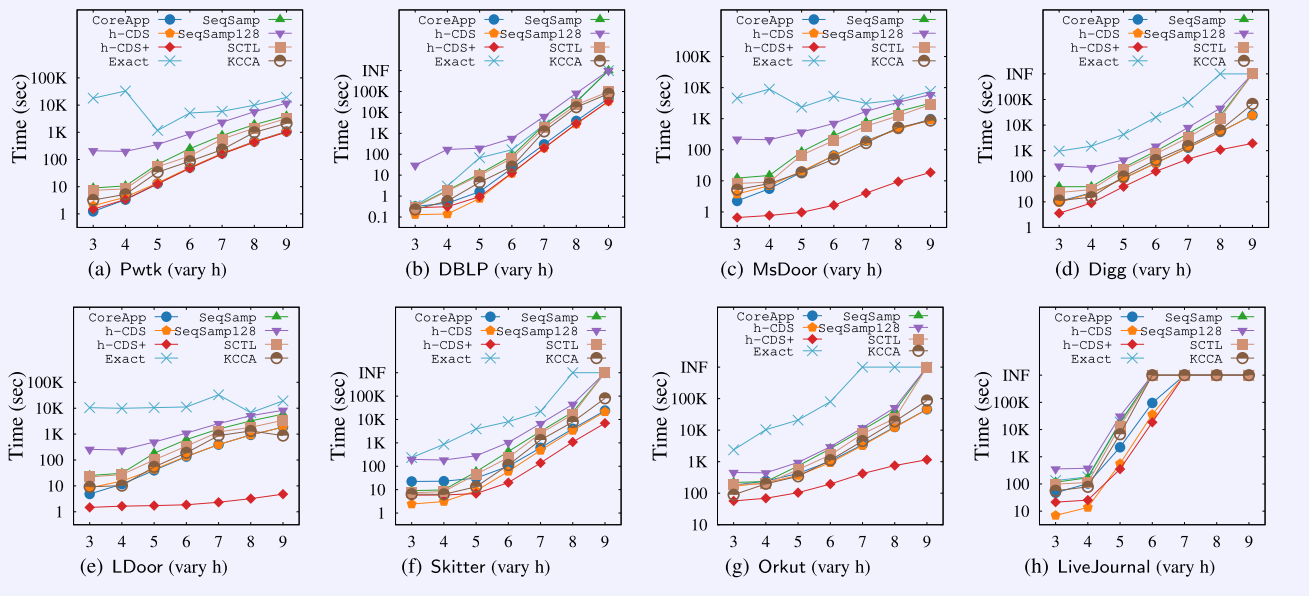


Fig. 12. Running time of different algorithms for H-CLIQUE-DS-PROBLEM with varying h from 3 to 9.

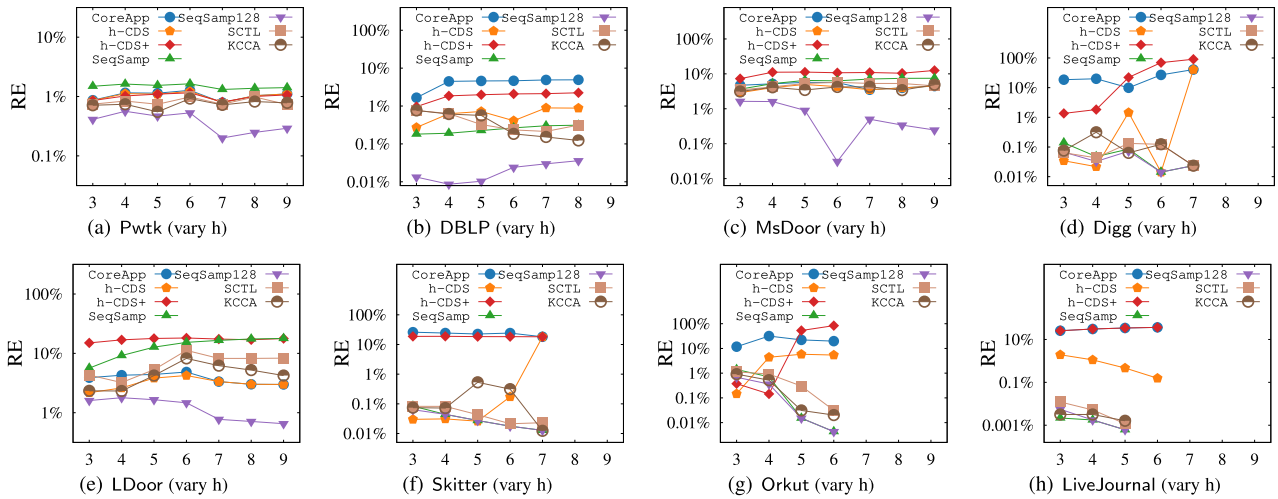


Fig. 13. Relative error (RE) of different algorithms with varying h from 3 to 9 (Points are missing where Exact runs out of 24 hours.).

Exp-6: Approximation performance. We use the relative error (RE) as a metric to evaluate the approximation performance of different algorithms which is defined as $|\sigma_h^* - \sigma_h|/\sigma_h^*$, where σ_h^* is the h-clique density of the h-clique densest subgraph obtained by Exact and σ_h is an estimated value. To compute the relative error, we run each approximation algorithm 100 times and then take the average value over the 100 runs as the final result.

Fig. 13 shows the relative errors of various algorithms on the larger eight datasets. Some points are missing on large h values where the exact algorithm cannot get a solution with 24 hours. Note that CoreApp and h-CDS achieve the same relative error because they both return the $\mathrm{CC}^h_{\hat{c}c}$ as an approximation. We can see that the compared algorithms, SeqSamp, SeqSamp128, KCCA, and SCTL, all achieve good approximations for the h-clique densest subgraph, as these algorithms generally require counting the number of cliques. However, in terms of density, our proposed algorithms, h-CDS and h-CDS+, also show strong performance in terms of error rates. Although they may not always lead in relative comparisons, when considering absolute values, across all datasets, the relative error (RE) of our proposed algorithms is consistently below 10%. Coupled with the running time data from Exp-4, this demonstrates that the colorful h-star core-based algorithms, h-CDS and h-CDS+, produce high-quality results using significantly less time. Among all datasets, SeqSamp128 yields the best error performance, which is expected given that this algorithm undergoes 128 iterations, while SeqSamp, KCCA, and SCTL only use one iteration. This highlights that sample-based algorithms benefit from increased iterations, leading to better outcomes. However, although SeqSamp128 can achieve a lower relative error, it is

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

TABLE III
THE POWER OF PRUNING TECHNIQUES USED IN h-CDS ($h = 6$, 1K=1,000, 1M=1,000,000, 1G=1,000,000,000)

Dataset	$n = \|V\|$			$m = \|E\|$			#Density=m/n			Δ	σ_6: 6-clique density		
	G	CSC^h_θ	$\mathsf{CC}^h_{\hat{cc}}$	G	CSC^h_θ	$\mathsf{CC}^h_{\hat{cc}}$	G	CSC^h_θ	$\mathsf{CC}^h_{\hat{cc}}$	CSC^h_θ	G	CSC^h_θ	$\mathsf{CC}^h_{\hat{cc}}$
Nasasrb	54.9K	52.1K	1.6K	1.3M	1.3M	28.4K	23.90	24.13	17.50	5.13%	12.36K	12.99K	11.14K
Pkustk	87.8K	41.3K	396	2.6M	1.4M	9.1K	29.21	32.75	23.05	53.01%	25.90K	29.71K	95.19K
Buzznet	101K	33.8K	275	2.8M	2.2M	21.5K	27.31	65.04	78.23	66.63%	63.68K	191K	4.17M
DBLP	317K	114	114	1.0M	6.4K	6.4K	3.31	56.50	56.50	99.96%	13.31K	23.39M	23.39M
Digg	771K	23.4K	153	5.9M	2.9M	9.5K	7.66	125.26	62.29	96.96%	20.01K	658K	10.25M
Skitter	1.7M	3.0K	180	11.1M	222K	11.9K	6.55	73.87	66.24	99.82%	5.76K	2.66M	9.53M
Orkut	3.0M	693K	132	106M	50.4M	7.5K	35.48	72.70	56.77	76.87%	15.75K	64.80K	7.57M
LiveJournal	4.8M	483	385	42.9M	108K	73.7K	8.84	224.41	191.31	99.99%	1.70M	16.86G	10.72G

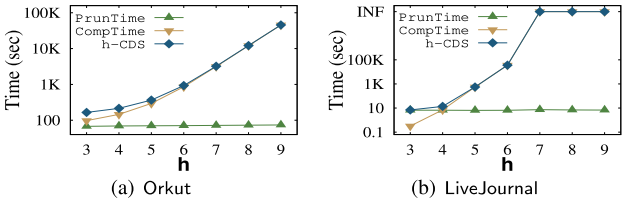


Fig. 14. The running time distribution of h-CDS.

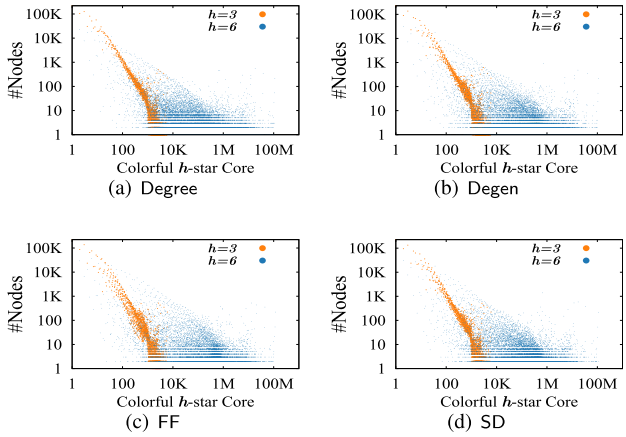


Fig. 15. Nodes' distributions of colorful h -star cores based on different graph colorings (Skitter).

very costly and cannot provide an approximation for larger h values, such as $h = 6$ on LiveJournal and $h = 9$ on Skitter and Orkut.

Exp-7: The performance of h-CDS. Here we evaluate h-CDS in terms of the running time distribution and the graph size reduction for $h = 6$ on Orkut and LiveJournal. Recall that h-CDS first prunes the graph using the colorful h-star θ-core CSC^h_θ, and then calls CoreApp to compute the h-clique $\hat{cc}$-core $\mathsf{CC}^h_{\hat{cc}}$ to achieve a $\frac{1}{h}$-approximation solution. The runtime of h-CDS includes the pruning time and the time spent on computing $\mathsf{CC}^h_{\hat{cc}}$, denoted by PrunTime and CompTime respectively. Fig. 14 shows the time distribution of h-CDS on different datasets. As can be seen, PrunTime is stable with an increasing h from 3 to 9 on all graphs, while CompTime increases significantly as h increases. For a small h, PrunTime and CompTime are comparable, while for a large h, PrunTime is dominated by CompTime. These results indicate that the cost of h-CDS is mainly dominated by computing $\mathsf{CC}^h_{\hat{cc}}$, and the pruning procedure is very efficient.

Table III shows the statistics of G, CSC^h_θ and $\mathsf{CC}^h_{\hat{cc}}$. In Table III, $\Delta = (n_1 - n_2)/n_1$ and σ_6 is 6-clique density where $n_1 = |V_G|$, $n_2 = |V_{\mathsf{CSC}^h_\theta}|$ which can be used to measure the effectiveness of the pruning rule in h-CDS. From Table III, we can see that our pruning strategy is very effective; it can largely prune the nodes that are definitely not contained in CSC^h_θ, especially on LiveJournal, DBLP, Skitter and Digg where Δ can achieve nearly 99.99%. Taking the LiveJournal dataset as an example, after removing a large number of nodes from the original graph ($|V| = 4,847,572$), our pruning technique returns CSC^h_θ with only 483 nodes, which is quite close to the target subgraph $\mathsf{CC}^h_{\hat{cc}}$ (385 nodes). In addition, CSC^h_θ also improves the density over the original graph by up to 17 times (DBLP, Digg and LiveJournal). On all datasets except Buzznet and DBLP, the densities of $C^{\mathcal{S}}_\theta$ are higher than those of $\mathsf{CC}^h_{\hat{cc}}$, suggesting that our colorful h-star core is indeed very cohesive. Also, we can see that CSC^h_θ can significantly increase the clique density. On LiveJournal and Nasasrb, CSC^h_θ even achieves a higher clique density than $\mathsf{CC}^h_{\hat{cc}}$. These results indicate that the colorful h-star core can provide a very good approximation for the h-clique densest subgraph.

D. The Effects of Graph Colorings

In this subsection, we study how graph colorings impact the performance of the colorful h-star core decomposition and qualities of the colorful h-star maximal core. Among all graph coloring techniques, greedy coloring algorithms using node-ordering heuristics to reduce the number of colors, turn out to be the most efficient algorithms. Here are four popular ordering heuristics studied in the recent literature [12]:

- *Degree:* Coloring the nodes following a non-increasing ordering of degree (break ties by node ID) [30].
- *Degen:* Coloring the nodes following an inverse degeneracy ordering [12]. Note that such an inverse degeneracy ordering can be easily obtained by reversing the node-deletion ordering of the core-decomposition algorithm [23].
- *FF:* Coloring the nodes in the order they appear in the input graph representation [30].
- *SD:* Coloring an uncolored node whose colored neighbor nodes use the largest number of distinct colors [31].

Exp-8: The nodes' distribution. Fig. 15 shows the nodes' distribution of various colorful h-star k-cores on Skitter for

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
PERFORMANCE OF DIFFERENT COLOR ALGORITHMS (DATASET: Skitter, H IS THE COLORFUL 6-STAR $c\hat{s}c$-CORE)

	$n = \|V_H\|$	$m = \|E_H\|$	χ	$\sigma_6(H)$
Degree	212	15,503	71	10.27M
Degen	213	15,609	75	10.28M
FF	233	15,145	101	10.08M
SD	213	15,600	68	10.31M

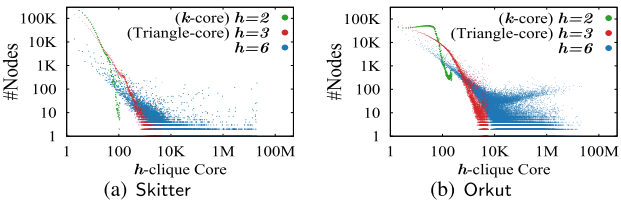


Fig. 16. Nodes' distributions of h-clique cores.

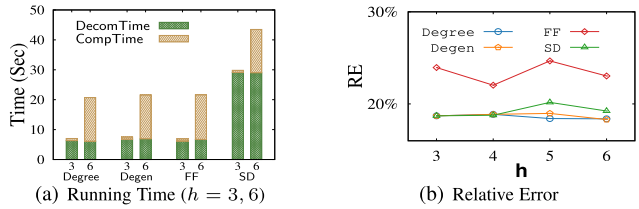


Fig. 17. The qualities of h-CDS+ with different graph colorings.

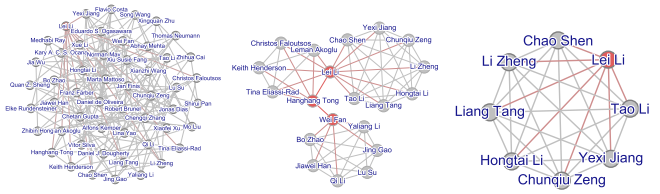


Fig. 18. The cohesive subgraphs found in DBLP, based on k-core, colorful 3-star k-core and 3-clique densest subgraph.

$h = 3$ and 6 when using different coloring algorithms. Note that, both x-axis and y-axis are in log scale. Thus, the distribution of colorful h-star core numbers generally follows a power-law distribution when $h = 3$. As can be seen in Fig. 15, most nodes have smaller colorful h-star degree and the maximum colorful h-star core number gets larger with varying h from 3 to 6. The nodes' distributions are very similar in the settings of four different coloring algorithms. A slight difference is that nodes in Fig. 15(c) are more scattered. A possible reason is that the FF heuristic uses the larger number of colors to color nodes (see Table IV), which makes a h-star more likely to be colorful and each node participates in more colorful h-stars. In our experiments, we set Degree as the default coloring algorithm for h-CDS and h-CDS+, because Degree runs very faster than Degen (it needs to compute k-core decomposition) and SD (it picks an uncolored node dynamically).

Fig. 16 shows the nodes' distribution of different h-clique k-cores on Skitter and Orkut for $h = 2$, 3 and 6. Again, the distribution of h-clique core number follows a power-law distribution, thus most of nodes participate in less h-cliques and are contained in a h-clique core with smaller core number. In addition, we can also observe that 3-clique cores (see Fig. 16(a)) and colorful 3-star cores (see Fig. 15(a)) have the similar nodes' distribution. However, the distributions for $h = 6$ show slightly different situations: there are more nodes with small 6-clique core number compared to nodes with small colorful 6-star core number in Fig. 15. Even so, the two kinds of distributions share the same trend when h values get larger. Therefore, our colorful h-star core model can still be seen as a good approximation of h -clique core.

Exp-9: The qualities of h-CDS+ *with different graph colorings*. Fig. 17 shows the performance of h-CDS+ in terms of the time consumption and relative errors on Skitter. In Fig. 17(a), DecomTime and CompTime indicate the time cost on the colorful h-star core decomposition and computing h-clique density of the colorful h-star $c\hat{s}c$-core whose statistics are shown in Table IV. As can be seen, SD consumes more time than the other three coloring techniques. This is because Degree, Degen and FF color nodes following a fixed order, but SD selects an uncolored node dynamically based on the number of distinct colors of their neighbors, which can be very costly though it produces higher quality colorings (using smaller number of colors in Table IV). For a large h value, CompTime increases significantly since computing h-clique density involves counting the number of h-cliques which runs very slow. In Fig. 17(b), we can see that h-CDS+ equipped with FF has a larger relative error. The results suggest that the estimating precision of h-CDS+ seems to match the number of colors used in greedy coloring algorithms shown in Table IV, and the reason might be that when less colors are used in coloring algorithms, a colorful h-star has the high probability to be a h-clique, thus it can be a good approximation of a clique.

E. Case Studies

Exp-10: We extract a subgraph, namely DBLPs, from DBLP for case studies. DBLPs, containing 3,545 nodes and 5,076 edges, is a collaboration network of authors who published at least two papers in the database (DB) and data mining (DM) related conferences between 2010 and 2020. Here we perform three queries for a given author's name (in this case study, "Lei Li" is used as an illustrative example) to compute the k-core, the colorful 3-star core and the 3-clique densest subgraph that contain this author on DBLPs (Fig. 18), respectively. As depicted in Fig. 18(a), Dr. Lei Li is located in a 4-core. In this subgraph, every author collaborated with at least four researchers over the period of ten years. The result for the colorful 3-star k-core is quite different from the result of k-core (Fig. 18(b)), and it is also more compact and cohesive than k-core. The authors marked by red (i.e., Dr. Lei Li, Hanghang Tong and Wei Fan) are group leaders or senior researchers. They basically act as the "bridges" connecting their group with others, which play an important role in joint work with other authors. The triangle densest subgraph that contains Dr. Lei Li shown in Fig. 18(c) exhibits a different

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

semantic. The result turns out to be a clique, which is also a subgraph of the colorful 3-star k-core. Researchers involved in this clique closely collaborated with each other. However, such a clique structure cannot reveal the collaboration relationships with the other groups. Thus, compared to the k-core and the h-clique densest subgraph, our colorful h-star core will be better to reveal the close collaboration between different research groups.

VI. Related Work

Cohesive subgraph models: A large number of cohesive subgraph models have been proposed based on different cohesiveness measures [32]. Notable examples include k-core [19], k-truss [20], maximal k-edge connected subgraph [21], and maximal cliques [33], [34], [35]. *k-core* is a maximal subgraph among all subgraphs in G, in which the degree of each node is at least k [23], [36]. Recently, such a concept was extended to uncertain graph [37], [38], attributed graphs [39], [40], distance generalized cores [41], [42], [43] and so on [44], [45], [46], [47], [48]. A *k-truss* is a maximal subgraph where each edge participates in at least $k-2$ triangles [20], [49], [50], [51], [52], [53], [54]. A maximal k-edge connected subgraph is a maximal subgraph such that any two nodes in that subgraph have at least k edge disjoint paths connecting them [21], [55], [56]. All the k-core, k-truss, and maximal k-edge connected subgraph can be computed in polynomial time by a peeling-style algorithm. A maximal clique is a maximal complete subgraph. Finding all maximal cliques in a graph is a classic NP-hard problem. Practical algorithms for maximal clique enumeration are often based on the classic Bron-Kerbosch algorithm [22]. The concept of maximal clique was also extended to the context of uncertain graphs [57], [58]. In addition, there also exist some other cohesive subgraph models including *query-biased density* [59], [60], k-clique cores [14], and k-(r,s) nucleus [61], [62], [63], [64] (a generalization of k-core and k-truss). Different from all the above models, our *colorful h-star core* model can be considered as a relaxation of the k-clique core model. Moreover, unlike the k-clique core, our model can be computed in near-linear time.

The densest subgraph problem: The dense subgraphs problem has been widely studied [65], [66], [67]. Given a pattern (also called motif), such a problem is to extract a subgraph which maximizes the pattern density, i.e., the average number of patterns on nodes. The most commonly studied density is average degree, defined as $\frac{m}{n}$. Finding a subgraph that maximizes the average degree was referred to as the densest subgraph problem, which can be solved using a parametric maximum-flow algorithm in polynomial time [65], [68]. Epasto et al. studied the densest subgraph computation in evolving graphs [66]. Qin et al. proposed an algorithm to find the top-k locally densest subgraphs [69]. The concept of h-clique densest subgraph (h-Clique-DS) was first introduced by Tsourakakis [15] by extending the concept of the densest subgraph (DS) and the triangle densest subgraph (TDS), which are special cases for $h=2$ and $h=3$ respectively [15], [67]. The h-clique densest subgraph problem (h-Clique-DS-Problem) aims to find h-Clique-DS among all subgraphs of G. Tsourakakis et al. generalized the greedy $\frac{1}{2}$-approximation algorithm to a $\frac{1}{h}$-approximation algorithm for h-Clique-DS-Problem. Raman et al. [16] investigated a higher-order variant of locally dense subgraph [69] based on triangle, called top-k local triangle-densest subgraph discovery. Mitzenmacher et al. proposed a randomized algorithm to identify an h-clique dense subgraph [67]. Later in [14], the authors developed an approximation solution based on the h-clique core. Recently, Sun et al. introduced an alternative approach by sampling h-cliques to save space [17]. He et al. [26] propose SCTL, which builds an index structure to expedite the k-clique enumeration process, and then compute the h-clique densest subgraph. Fang et al. [27] propose a SOTA approach of h-clique densest subgraph problem by employing a effective Frank-Wolfe-based framework that utilizes k-clique counting instead of k-clique enumeration. Besides, there are a lot of research on densest subgraph recently [70], [71], [72], [73], [74], [75], [76], [77], [78]. In this manuscript, we propose a new graph reduction technique based on our colourful h-star core which is dramatically different from all the previous algorithms.

The colorful motif and star counting: For the colorful motif, research has focused on colorful cycles [10], [79], colorful paths [80], [81], and colorful independent sets [82]. The studies most relevant to our work include the following: Rubert et al. [83], [84] present SIMBio, a tool designed for inferring and searching colorful motifs in biological networks, which demonstrates improved performance over existing tools like MOTUS and Torque in identifying relevant motifs and biological patterns. Italiano et al. [85] investigate the NP-hard problem of finding a maximum colorful clique in vertex-colored graphs, providing complexity results and developing efficient algorithms for specific graph classes, including XP parameterized and polynomial-time solutions. Additionally, there has been considerable recent work on star counting [86], [87], [88], [89]. Notably, Finocchi et al. [89] introduce efficient sequential and parallel algorithms for listing and counting k-diamonds, which are relaxed cliques with one missing edge, showcasing optimal performance in dense graphs and improved runtime over k-clique algorithms for sparse graphs. Meanwhile, Yu et al. [87] establish that counting $O(1)$-stars is the optimal method among constant degree polynomial tests for strongly distinguishing a graph instance $G(n,p)$ from the union of a random copy of graph H with another instance of $G(n,p)$, thus generalizing and extending previous results on the inference capabilities of $O(1)$-degree polynomials. However, we are the first to systematically analyze the definition, algorithms, and applications of colorful h-stars.

VII. Conclusion

In this manuscript, we present the novel concept of the colorful h-star within colorful graphs and propose two sophisticated models for cohesive subgraphs: the colorful h-star core and the colorful h-star truss. We illustrate that these colorful h-stars can be efficiently enumerated and updated using an innovative dynamic programming (DP) algorithm. Building upon this DP algorithm, we have devised a decomposition algorithm for the colorful h-star core with a time complexity of $O(hm)$ and a space complexity of $O(hn+m)$. Furthermore, we introduce a

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

decomposition algorithm for the colorful h-star truss, characterized by a time complexity of $O(hm^{1.5})$ and a space complexity of $O(hm)$, where m is the number of edges and n is the number of nodes in the graph. Based on our proposed model, we propose a graph reduction technique to speed up an approximate k-clique densest subgraph mining algorithm. Moreover, we show that the colorful h-star core with the maximum core number is also a very good approximation of the k-clique densest subgraph. We conduct extensive experiments on 11 large real-world graphs, and the results demonstrate the efficiency, scalability and effectiveness of the proposed solutions.

REFERENCES

[1] P. Ribeiro, P. Paredes, M. E. P. Silva, D. Aparício, and F. M. A. Silva, "A survey on subgraph counting: Concepts, algorithms, and applications to network motifs and graphlets," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 28:1–28:36, 2022.

[2] Z. Gao, C. Cheng, Y. Yu, L. Cao, C. Huang, and J. Dong, "Scalable motif counting for large-scale temporal graphs," in *Proc. IEEE Int. Conf. Data Eng.*, 2022, pp. 2656–2668.

[3] M. Bressan, S. Leucci, and A. Panconesi, "Faster motif counting via succinct color coding and adaptive sampling," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 6, pp. 96:1–96:27, 2021.

[4] J. Wang, Y. Wang, W. Jiang, Y. Li, and K. Tan, "Efficient sampling algorithms for approximate temporal motif counting," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 1505–1514.

[5] H. Peng, J. Li, Q. Gong, Y. Ning, S. Wang, and L. He, "Motif-matching based subgraph-level attentional convolutional network for graph classification," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5387–5394.

[6] L. P. Nguyen, J. Mille, D. H. Li, D. Conte, and N. Ragot, "Efficient dynamic texture classification with probabilistic motifs," in *Proc. Int. Conf. Pattern Recognit.*, 2022, pp. 564–570.

[7] F. Xia, S. Yu, C. Liu, J. Li, and I. Lee, "CHIEF: Clustering with higher-order motifs in big networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 3, pp. 990–1005, May/Jun. 2022.

[8] P. Li, G. J. Puleo, and O. Milenkovic, "Motif and hypergraph correlation clustering," *IEEE Trans. Inf. Theory*, vol. 66, no. 5, pp. 3065–3078, May 2020.

[9] P. Aboulker, M. Bonamy, N. Bousquet, and L. Esperet, "Distributed coloring in sparse graphs with fewer colors," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2018, pp. 419–425.

[10] D. Chakraborti, A. M. Frieze, and M. Hasabnis, "Colorful hamilton cycles in random graphs," *SIAM J. Discret. Math.*, vol. 37, no. 1, pp. 51–64, 2023.

[11] Z. Zhang and J. Guo, "Colorful graph coloring," in *Proc. Int. Workshop Front. Algorithmics*, 2022, pp. 141–161.

[12] W. Hasenplaugh, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Ordering heuristics for parallel graph coloring," in *Proc. ACM Symp. Parallelism Algorithms Architectures*, 2014, pp. 166–177.

[13] L. Yuan, L. Qin, X. Lin, L. Chang, and W. Zhang, "Effective and efficient dynamic graph coloring," *Proc. VLDB Endowment*, vol. 11, no. 3, pp. 338–351, 2017.

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

[15] C. E. Tsourakakis, "The k-clique densest subgraph problem," in *Proc. Int. Conf. World Wide Web*, 2015, pp. 1122–1132.

[16] R. Samusevich, M. Danisch, and M. Sozio, "Local triangle-densest subgraphs," in *Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Mining*, 2016, pp. 33–40.

[17] B. Sun, M. Danisch, T. H. Chan, and M. Sozio, "KClist: A simple algorithm for finding k-clique densest subgraphs in large graphs," *Proc. VLDB Endowment*, vol. 13, no. 10, pp. 1628–1640, 2020.

[18] B. Balasundaram and S. Butenko, "Graph domination, coloring and cliques in telecommunications," in *Handbook of Optimization in Telecommunications*, Berlin, Germany: Springer, 2006, pp. 865–890.

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

[20] J. Wang and J. Cheng, "Truss decomposition in massive networks," *Proc. VLDB Endowment*, vol. 5, no. 9, pp. 812–823, 2012.

[21] R. Zhou, C. Liu, J. X. Yu, W. Liang, B. Chen, and J. Li, "Finding maximal k-edge-connected subgraphs from a large graph," in *Proc. Int. Conf. Extending Database Technol.*, 2012, pp. 480–491.

[22] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," *ACM J. Exp. Algorithmics*, vol. 18, 2013, Art. no. 3.1.

[23] V. Batagelj and M. Zaversnik, "An O(m) algorithm for cores decomposition of networks," 2003, *arXiv:0310049*.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. Cambridge, MA, USA: The MIT Press and McGraw-Hill Book Company, 2001.

[25] M. Latapy, "Main-memory triangle computations for very large (sparse (power-law)) graphs," *Theor. Comput. Sci.*, vol. 407, no. 1/3, pp. 458–473, 2008.

[26] Y. He, K. Wang, W. Zhang, X. Lin, and Y. Zhang, "Scaling up K-clique densest subgraph detection," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 69:1–69:26, 2023.

[27] Y. Zhou, Q. Guo, Y. Fang, and C. Ma, "A counting-based approach for efficient K-clique densest subgraph discovery," *Proc. ACM Manag. Data*, vol. 2, no. 3, 2024, Art. no. 119.

[28] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, "Parallel maximum clique algorithms with applications to network analysis," *SIAM J. Sci. Comput.*, vol. 37, no. 5, pp. C589–C616, 2015.

[29] V. Batagelj and M. Zaversnik, "Fast algorithms for determining (generalized) core groups in social networks," *Adv. Data Anal. Classification*, vol. 5, no. 2, pp. 129–145, 2011.

[30] D. J. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *Comput. J.*, vol. 10, no. 1, pp. 85–86, 1967.

[31] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[32] L. Chang and L. Qin, "Cohesive subgraph computation over large sparse graphs," in *Proc. IEEE Int. Conf. Data Eng.*, 2019, pp. 2068–2071.

[33] T. Yu et al., "Incremental maximal clique enumeration for hybrid edge changes in large dynamic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 4, pp. 1650–1666, Apr. 2024.

[34] D. Yu, L. Zhang, Q. Luo, X. Cheng, and Z. Cai, "Maximal clique search in weighted graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9421–9432, Sep. 2023.

[35] K. Yu and C. Long, "Fast maximal quasi-clique enumeration: A pruning and branching co-design approach," *Proc. ACM Manag. Data*, vol. 1, no. 3, pp. 211:1–211:26, 2023.

[36] R.-H. Li, J. Yu, and R. Mao, "Efficient core maintenance in large dynamic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2453–2465, Oct. 2014.

[37] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich, "Core decomposition of uncertain graphs," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 1316–1325.

[38] B. Yang, D. Wen, L. Qin, Y. Zhang, L. Chang, and R. Li, "Index-based optimal algorithm for computing k-cores in large uncertain graphs," in *Proc. IEEE Int. Conf. Data Eng.*, 2019, pp. 64–75.

[39] R. Li, L. Qin, J. X. Yu, and R. Mao, "Influential community search in large networks," *Proc. VLDB Endowment*, vol. 8, no. 5, pp. 509–520, 2015.

[40] R. Li et al., "Skyline community search in multi-valued networks," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 457–472.

[41] F. Bonchi, A. Khan, and L. Severini, "Distance-generalized core decomposition," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1006–1023.

[42] Q. Dai et al., "Scaling up distance-generalized core decomposition," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 312–321.

[43] Q. Liu, X. Zhu, X. Huang, and J. Xu, "Local algorithms for distance-generalized core decomposition over large dynamic graphs," *Proc. VLDB Endowment*, vol. 14, no. 9, pp. 1531–1543, 2021.

[44] W. Bai, Y. Jiang, Y. Tang, and Y. Li, "Parallel core maintenance of dynamic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 8919–8933, Sep. 2023.

[45] J. Liu, C. Xu, C. Yin, W. Wu, and Y. Song, "K-core based temporal graph convolutional network for dynamic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3841–3853, Aug. 2022.

[46] L. Sun, X. Huang, R. Li, B. Choi, and J. Xu, "Index-based intimate-core community search in large weighted graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4313–4327, Sep. 2022.

[47] X. Liao, Q. Liu, J. Jiang, X. Huang, J. Xu, and B. Choi, "Distributed d-core decomposition over large directed graphs," *Proc. VLDB Endowment*, vol. 15, no. 8, pp. 1546–1558, 2022.

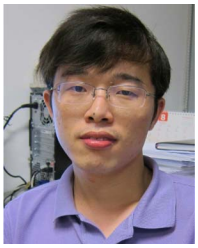
Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.

[48] X. Sun, X. Huang, and D. Jin, "Fast algorithms for core maximization on large graphs," *Proc. VLDB Endowment*, vol. 15, no. 7, pp. 1350–1362, 2022.
[49] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," in *Proc. Int. Conf. Manage. Data*, 2014, pp. 1311–1322.
[50] X. Huang, L. V. S. Lakshmanan, J. X. Yu, and H. Cheng, "Approximate closest community search in networks," *Proc. VLDB Endowment*, vol. 9, no. 4, pp. 276–287, 2015.
[51] Z. Sun, X. Huang, Q. Liu, and J. Xu, "Efficient star-based truss maintenance on dynamic graphs," *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 133:1–133:26, 2023.
[52] A. Tian, A. Zhou, Y. Wang, and L. Chen, "Maximal d-truss search in dynamic directed graphs," *Proc. VLDB Endowment*, vol. 16, no. 9, pp. 2199–2211, 2023.
[53] Q. Luo, D. Yu, X. Cheng, H. Sheng, and W. Lyu, "Exploring truss maintenance in fully dynamic graphs: A mixed structure-based approach," *IEEE Trans. Comput.*, vol. 72, no. 3, pp. 707–718, Mar. 2023.
[54] Z. Chen, L. Yuan, L. Han, and Z. Qian, "Higher-order truss decomposition in graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3966–3978, Apr. 2023.
[55] T. Akiba, Y. Iwata, and Y. Yoshida, "Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 909–918.
[56] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang, "Efficiently computing k-edge connected components via graph decomposition," in *Proc. Int. Conf. Manage. Data*, 2013, pp. 205–216.
[57] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Mining maximal cliques from an uncertain graph," in *Proc. IEEE Int. Conf. Data Eng.*, 2015, pp. 243–254.
[58] R. Li, Q. Dai, G. Wang, Z. Ming, L. Qin, and J. X. Yu, "Improved algorithms for maximal clique search in uncertain networks," in *Proc. IEEE Int. Conf. Data Eng.*, 2019, pp. 1178–1189.
[59] C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. A. Tsiarli, "Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 104–112.
[60] J. Abello, M. G. C. Resende, and S. Sudarsky, "Massive quasi-clique detection," in *Proc. Latin Amer. Symp. Theor. Inform.*, 2002, pp. 598–612.
[61] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek, "Finding the hierarchy of dense subgraphs using nucleus decompositions," in *Proc. Int. Conf. World Wide Web*, 2015, pp. 927–937.
[62] A. E. Sariyuce and A. Pinar, "Fast hierarchy construction for dense subgraphs," *Proc. VLDB Endowment*, vol. 10, no. 3, pp. 97–108, 2016.
[63] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek, "Nucleus decompositions for identifying hierarchy of dense subgraphs," *ACM Trans. Web*, vol. 11, no. 3, pp. 16:1–16:27, 2017.
[64] A. E. Sariyuce, C. Seshadhri, and A. Pinar, "Local algorithms for hierarchical dense subgraph discovery," *Proc. VLDB Endowment*, vol. 12, no. 1, pp. 43–56, 2018.
[65] A. V. Goldberg, *Finding a Maximum Density Subgraph*. Berkeley, CA, USA: Univ. California Berkeley, 1984.
[66] A. Epasto, S. Lattanzi, and M. Sozio, "Efficient densest subgraph computation in evolving graphs," in *Proc. Int. Conf. World Wide Web*, 2015, pp. 300–310.
[67] M. Mitzenmacher, J. Pachocki, R. Peng, C. E. Tsourakakis, and S. C. Xu, "Scalable large near-clique detection in large-scale networks via sampling," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 815–824.
[68] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, "A fast parametric maximum flow algorithm and applications," *SIAM J. Comput.*, vol. 18, no. 1, pp. 30–55, 1989.
[69] L. Qin, R. Li, L. Chang, and C. Zhang, "Locally densest subgraph discovery," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 965–974.
[70] C. Ma, Y. Fang, R. Cheng, L. V. S. Lakshmanan, X. Han, and X. Li, "Accelerating directed densest subgraph queries with software and hardware approaches," *VLDB J.*, vol. 33, no. 1, pp. 207–230, 2024.
[71] D. Chugh, H. Mittal, A. Saxena, R. Chauhan, E. Yafi, and M. Prasad, "Augmentation of densest subgraph finding unsupervised feature selection using shared nearest neighbor clustering," *Algorithms*, vol. 16, no. 1, p. 28, 2023. [Online]. Available: www.mdpi.com/about/announcements/784
[72] T. Hanaka, "Computing densest K-subgraph with structural parameters," *J. Comb. Optim.*, vol. 45, no. 1, 2023, Art. no. 39.
[73] Y. Xu, C. Ma, Y. Fang, and Z. Bao, "Efficient and effective algorithms for generalized densest subgraph discovery," *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 169:1–169:27, 2023.
[74] L. Chen, C. Liu, R. Zhou, K. Liao, J. Xu, and J. Li, "Densest multipartite subgraph search in heterogeneous information networks," *Proc. VLDB Endowment*, vol. 17, no. 4, pp. 699–711, 2023.
[75] J. Ding and H. Du, "Detection threshold for correlated erds-rényi graphs via densest subgraph," *IEEE Trans. Inf. Theory*, vol. 69, no. 8, pp. 5289–5298, Aug. 2023.
[76] A. Saha, X. Ke, A. Khan, and C. Long, "Most probable densest subgraphs," in *Proc. IEEE Int. Conf. Data Eng.*, 2023, pp. 1447–1460.
[77] T. B. Trung, L. Chang, N. T. Long, K. Yao, and H. T. T. Binh, "Verification-free approaches to efficient locally densest subgraph discovery," in *Proc. IEEE Int. Conf. Data Eng.*, 2023, pp. 1–13.
[78] A. Miyauchi, T. Chen, K. Sotiropoulos, and C. E. Tsourakakis, "Densest diverse subgraphs: How to plan a successful cocktail party with diversity," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 1710–1721.
[79] G. F. Italiano, Y. Manoussakis, K. T. Nguyen, and H. P. Pham, "Maximum colorful cycles in vertex-colored graphs," in *Proc. Int. Comput. Sci. Symp. Russia*, 2018, pp. 106–117.
[80] R. Dondi and M. M. Hosseinzadeh, "Finding colorful paths in temporal graphs," in *Proc. Int. Conf. Complex Netw. Appl.*, 2021, pp. 553–565.
[81] A. Gyárfás and G. N. Sárközy, "Induced colorful trees and paths in large chromatic graphs," *Electron. J. Comb.*, vol. 23, no. 4, 2016, Art. no. P4.46. [Online]. Available: www.combinatorics.org/ojs/index.php/eljc/article/view/v23i4p46
[82] Y. Manoussakis and H. P. Pham, "Maximum colorful independent sets in vertex-colored graphs," *Electron. Notes Discret. Math.*, vol. 68, pp. 251–256, 2018.
[83] D. P. Rubert, E. Araujo, and M. A. Stefanes, "SIMBio: Searching and inferring colorful motifs in biological networks," in *Proc. IEEE Int. Conf. Bioinf. Bioeng.*, 2015, pp. 1–6.
[84] D. P. Rubert, E. Araujo, M. A. Stefanes, J. Stoye, and F. V. Martinez, "Searching and inferring colorful topological motifs in vertex-colored graphs," *J. Comb. Optim.*, vol. 40, no. 2, pp. 379–411, 2020.
[85] G. F. Italiano, Y. Manoussakis, K. T. Nguyen, and H. P. Pham, "Maximum colorful cliques in vertex-colored graphs," in *Proc. Int. Comput. Combinatorics Conf.*, 2018, pp. 480–491.
[86] M. S. Tahaei and S. N. Hashemi, "Graph characterization by counting sink star subgraphs," *J. Math. Imag. Vis.*, vol. 57, no. 3, pp. 439–454, 2017.
[87] X. Yu, I. Zadik, and P. Zhang, "Counting stars is constant-degree optimal for detecting any planted subgraph: Extended abstract," in *Proc. Conf. Learn. Theory*, 2024, pp. 5163–5165.
[88] M. Aliakbarpour, A. S. Biswas, T. Gouleakis, J. Peebles, R. Rubinfeld, and A. Yodpinyanee, "Sublinear-time algorithms for counting star subgraphs via edge sampling," *Algorithmica*, vol. 80, no. 2, pp. 668–697, 2018.
[89] I. Finocchi, R. L. Garcia, and B. Sinaimeri, "From stars to diamonds: Counting and listing almost complete subgraphs in large networks," *Comput. J.*, vol. 67, no. 6, pp. 2151–2161, 2024.

Hongchao Qin received the BS degree in mathematics, and the ME and PhD degrees in computer science from Northeastern University, China, in 2013, 2015 and 2020, respectively. He is currently an assistant professor with the Beijing Institute of Technology, China. His current research interests include social network analysis and data-driven graph mining.

Gao Sen received the BSc and MSc degrees in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, in 2019 and 2022, respectively. Currently, he is working toward the PhD degree in computer science with the School of Computing, National University of Singapore. His research interests include distributed graph system, graph data management and graph privacy preserving.

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.



Rong-Hua Li received the PhD degree from the Chinese University of Hong Kong, in 2013. He is currently a professor with the Beijing Institute of Technology, Beijing, China. His research interests include graph data management and mining, social network analysis, graph computation systems, and graph-based machine learning.

Hongzhi Chen received the PhD degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, in 2020. He is currently a senior RD with ByteDance Infrastructure Team, Beijing, China, working on graph related storage, processing and training systems. His research interests cover the broad area of distributed systems and databases, with special emphasis on graph systems and machine learning/deep learning systems.

Ye Yuan received the BS, MS, and PhD degrees in computer science from Northeastern University, in 2004, 2007, and 2011, respectively. He is now a professor with the Department of Computer Science, Northeastern University, China. His research interests include graph databases, probabilistic databases, and social network analysis.

Guoren Wang received the BSc, MSc, and PhD degrees from the Department of Computer Science, Northeastern University, China, in 1988, 1991 and 1996, respectively. Currently, he is a professor with the Department of Computer Science, Beijing Institute of Technology, Beijing, China. His research interests include XML data management, query processing and optimization, bioinformatics, high dimensional indexing, parallel database systems, and cloud data management. He has published more than 100 research papers.

Authorized licensed use limited to: National University of Singapore. Downloaded on February 27,2026 at 04:16:15 UTC from IEEE Xplore. Restrictions apply.