Colorful h-star Core Decomposition

Sen Gao†, Rong-Hua Li†, Hongchao Qin†, Hongzhi Chen‡, Ye Yuan†, Guoren Wang†
†*Beijing Institute of Technology, China;* ‡*ByteDance.*
Gawssin@gmail.com; lironghuabit@126.com; qhc.neu@gmail.com;
chenhongzhi@bytedance.com; yuan-ye@bit.edu.cn; wanggrbit@126.com

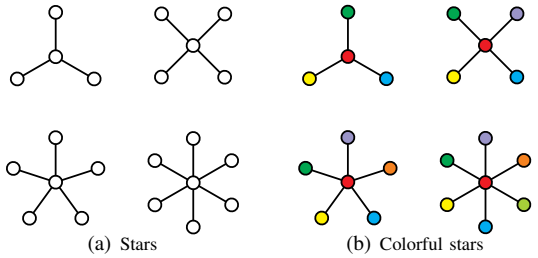
***Abstract*—The h-clique based higher-order cohesive subgraph mining is an important operator in graph analysis. The h-clique core and h-clique densest subgraph are two representative higher-order cohesive subgraph models which have been widely used in many practical applications. However, computing these two models on large graphs is often very costly due to the hardness of counting the h-cliques. In this paper, we propose a relaxed higher-order cohesive subgraph model, called colorful h-star core, based on counting the number of colorful h-stars. Unlike the h-cliques, 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 an efficient colorful h-star core decomposition algorithm which takes $O(h \times m)$ time and uses $O(h \times n + m)$ space, where m and n denote the number of edges and nodes of the graph respectively. In addition, we also propose a graph reduction technique based on our colorful h-star core model to accelerate the computation of the state-of-the-art approximation algorithm for h-clique densest subgraph mining. Moreover, we show that the colorful h-star core can also provide a very good approximation of the h-clique densest subgraph. The results of comprehensive experiments on 11 large real-world datasets demonstrate the efficiency, scalability and effectiveness of the proposed algorithms.**

I. Introduction

Real-world graphs, such as social networks, biological networks, and communication networks often consist of cohesive subgraph structures. Mining cohesive subgraphs from a graph is a fundamental operator in many graph analysis tasks [1], which has attracted much attention in the database and data mining communities due to a large number of applications, such as community search [2]–[4], locating influential nodes [5], [6], keyword extraction from text [7], [8], and real-time story identification [9], [10].

There exist many different cohesive subgraph models, including k-core [11], [12], k-truss [13], [14], k-plex [15], h-clique [16]–[18], k-edge connected subgraph [19], [20], densest subgraph [2], [3] and so on. The cohesiveness of a subgraph is usually measured by the minimum degree, the average degree, or edge connectivity [10]. Mining cohesive subgraphs based on different cohesiveness measures leads to the resulting subgraphs with different properties, and also often requires different levels of computational effort [10], [21].

Recent literature [22]–[25] on h-clique based higher-order cohesive subgraph mining has attracted much attention. The higher-order variants of k-core and densest subgraph have been proposed to identify more cohesive subgraphs of the graph. Specifically, a higher-order k-core model, called h-clique k core, is defined as a maximal subgraph, in which each node participates in at least k h-cliques (h-clique is a subgraph with h nodes such that each pair of nodes is connected with an edge) [16], [22] . The h-clique densest subgraph, first introduced by Tsourakakis [23], is a higher-order variant of the traditional densest subgraph [23]–[25]. Instead of maximizing the average degree, 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. Although those higher-order cohesive subgraph models have been well studied and also have successfully been applied in many different domains [22]–[25], the computation of those models is often very costly on large graphs for a relatively large h (e.g., $h = 6$). This is because all those h-clique based higher-order cohesive subgraph models need to count the number of h-cliques, which is often costly for a large h due to combinatorial explosions.



(a) Stars (b) Colorful stars

Fig. 1. Illustration of stars and colorful stars.

To address this issue, 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 [26], [27] such that any two adjacent nodes in the graph have different colors. An h-star $\mathcal{R}$ is a tree with a central node connected to the other $h-1$ nodes (Fig 1(a)). A colorful h-star, denoted by $\mathcal{S}$, 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 traditional k-core, we can also obtain a colorful h-star core decomposition of the graph by determining all the colorful h-star k cores.

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 [22]. To summarize, the main contributions of this paper are as follows.

New model. We propose a new higher-order cohesive subgraph model, called colorful h-star core, based on a newly-

introduced concept of colorful h-star. Our model can be considered as a relaxation of the h-clique core model. Unlike the h-clique core model, a striking feature of our model is that it can be computed in near-linear time w.r.t. the graph size.

Novel algorithms. We propose a DP algorithm to compute the number of colorful h-stars for each node. The novel DP-based updating technique 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. Since h is often very small (less than 10), our work provides a near-linear time solution for higher-order graph analysis applications. We also present a colorful h-star core based graph reduction technique to accelerate the h-clique core based approximate h-clique densest subgraph mining algorithm without sacrificing approximation performance.

Extensive 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 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 [22]. (4) Graph coloring techniques can affect the performance of our algorithms, but the distributions of corlorful h-star core numbers are generally similar w.r.t. different graph coloring techniques. 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.

Reproducibility. The source code of this paper is released at Github: https://github.com/Gawssin/ColorfulStarCore for reproducibility purposes.

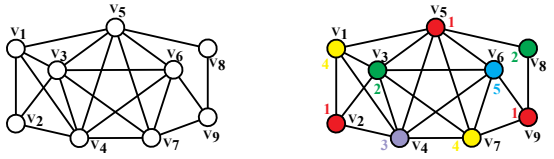
II. Preliminaries

Let $G = (V, E)$ be an undirected graph, where V ($|V| = n$) and E ($|E| = m$) denote the set of nodes and edges respectively. We denote with $N_u(G)$ 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 $H = (V_H, E_H)$ is called an induced subgraph of G if $V_H \subseteq V$ and $E_H = \{(u, v)|(u, v) \in E, u \in V_H, v \in V_H\}$.

An h-star $\mathcal{R}$ is a tree, with one internal or central node having degree $h-1$ and the other $h-1$ nodes having degree 1. Clearly, in a graph G, a node u participates in $\binom{d_u(G)}{h-1}$ h-stars which are centered on u if $d_u(G) \geq h-1$. Below, we first introduce the concept of graph coloring, and then define our colorful h-star model.

TABLE I
Notations and meanings

Notation	Meaning
$G = (V, E)$	a graph with nodes set V and edge set E
n, m	$n = \|V\|, m = \|E\|$
$N_u(G)$	$N_u(G)$ is the set of neighbors of u in G
$d_u(G)$	$d_u(G) = \|N_u(G)\|$
χ	the number of colors used in graph coloring algorithms
$d_u(G, \mathcal{H})$	pattern degree of u in G w.r.t. $\mathcal{H}$
C_k	the classical k-core
C_u	the core number of u in G
δ	the maximum core number of G
$C_k^{\mathcal{S}}$	the colorful h-star k core, also called $(k, \mathcal{S})$-core
$c_u(G, \mathcal{S})$	the colorful h-star core number of u in G
$\delta^{\mathcal{S}}$	the maximum colorful h-star core number of G
C_k^{Ψ}	the h-clique k core, also called (k, Ψ)-core
$c_h(G)$	the number of h-cliques in G
$\sigma_h(G)$	the h-clique density of G
δ^{Ψ}	the maximum h-clique core number of G
w	the number of nodes of a large clique
θ	a lower bound of h-clique density of G, $\theta = \binom{w-1}{h-1}$



(a) An undirected graph (b) The colored graph

Fig. 2. Illustration of the graph coloring technique.

Graph coloring is a procedure that assigns an integer color value taken from $[1, \cdots, \chi]$ to each node u in G, denoted by $\mathsf{color}(u)$, so that no two adjacent nodes have the same color value. Since the minimum coloring problem (χ is minimum) is NP-hard [28], we make use of a linear-time greedy coloring algorithm [26], [27] to obtain a valid coloring. The following example illustrates the graph coloring procedure.

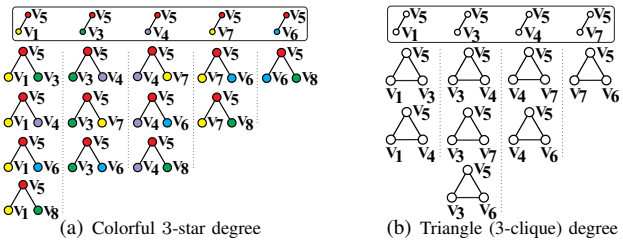
Example 1. *Consider a graph G in Fig. 2(a). We can color G based on an inverse degree ordering as used in [26], [27]. Clearly, in G, $(v_5, v_3, v_4, v_7, v_6, v_1, v_2, v_9, v_8)$ is an inverse degree ordering. Following this ordering, we first color v_5 with the smallest color 1, and then color v_3 with a color 2, and the other nodes are iteratively colored in a similar way. Fig. 2(b) shows the results of this coloring procedure.*

Based on a valid coloring, we define a concept called colorful h-star as follows.

Definition 1 (Colorful h-star)**.** *Given a colored graph $G = (V, E)$ and an integer $h \geq 2$, an h-star $\mathcal{S}$ in G is colorful if any pair of nodes $u, v \in \mathcal{S}$ have different color values.*

By Definition 1, 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 colors. However, by greedy coloring, a subgraph induced by the nodes of a colorful h-star may have many edges and has the potential to become a clique, as the greedy coloring algorithm ensures that two adjacent nodes have different 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.

Definition 2 (Pattern degree)**.** *Given a colored graph $G =$*



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

Fig. 3. Illustration of the pattern degree of v_5.

(V, E) and an integer h. Let $\mathcal{R}$, $\mathcal{S}$, Ψ be an h-star, a colorful h-star and an h-clique respectively. The h-star degree of a node u in G, denoted by $d_u(G, \mathcal{R})$, is the number of h-stars centered on u; the colorful h-star degree of u, denoted by $d_u(G, \mathcal{S})$, is the number of colorful h-stars centered on u; and the h-clique degree of u, denoted by $d_u(G, \Psi)$, is the number of h-cliques that u participates in.

Example 2 illustrates the definition of pattern degree.

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

For higher-order graph analysis, it often needs to compute the pattern degree of a node. For example, the h-clique based higher-order graph analysis applications [23]–[25] 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. We detail these challenges as follows.

Challenge. Unlike the h-star degree, which can be derived by a combinatorial formula, 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. Colorful h-star counting and updating

In this section, we first propose a dynamic programming (DP) algorithm to compute the colorful h-star degree of any given node. Then, we develop an efficient updating algorithm to update the colorful h-star degree of a node when one of its neighbors is removed.

Algorithm 1: The DP-based Counting Algorithm

Input: A graph G and a node u
Output: The colorful h-star degree $d_u(G, \mathcal{S})$

1 $\mathsf{color}[1, \cdots, n] \leftarrow \mathsf{GreedyColoring}(G)$;
2 **for** $i = 1$ **to** χ **do**
3 $\quad \mathsf{Group}(i) \leftarrow \{v | v \in N_u(G), \mathsf{color}(v) = i\}$;
4 $\quad \mathsf{cnt}(i) \leftarrow |\mathsf{Group}(i)|$;
5 $d_u(G, \mathcal{S}) \leftarrow \mathsf{DP}(\chi, h-1)$;
6 **return** $d_u(G, \mathcal{S})$;
7 **Procedure** $\mathsf{GreedyColoring}(G)$
8 Let π' be any ordering on nodes;
9 $flag(i) \leftarrow -1$ for $i = 1, \cdots, \chi$;
10 **for** *each node* $v \in \pi'$ *in order* **do**
11 $\quad$ **for** $u \in N_v(G)$ **do**
12 $\quad\quad flag(\mathsf{color}(u)) \leftarrow v$;
13 $\quad c \leftarrow \min\{i | i > 0, flag(i) \neq v\}$;
14 $\quad \mathsf{color}(v) \leftarrow c$;
15 **return** $\mathsf{color}(v)$ for all $v \in G$;
16 **Procedure** $\mathsf{DP}(c, \bar{h})$
17 **for** $i = 0$ **to** c **do**
18 $\quad$ **for** $j = 0$ **to** $\bar{h}$ **do**
19 $\quad\quad$ **if** $j = 0$ **then** $\mathsf{dp}(i, j) \leftarrow 1$;
20 $\quad\quad$ **else if** $i < j$ **then** $\mathsf{dp}(i, j) \leftarrow 0$;
21 $\quad\quad$ **else** $\mathsf{dp}(i, j) \leftarrow \mathsf{dp}(i-1, j-1) \times \mathsf{cnt}(i) + \mathsf{dp}(i-1, j)$;
22 **return** $\mathsf{dp}(c, \bar{h})$;

A. The DP 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 [26], [27] 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 $d_u(G, \mathcal{S})$ 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. $\mathsf{Group}(i) = \{v | v \in N_G(u), \mathsf{color}(v) = i\}$. The size of each color group is denoted by $\mathsf{cnt}(i) = |\mathsf{Group}(i)|$. Let $\mathsf{DP}(i, j)$ be 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 $i \in [1, \cdots, \chi], j \in [1, \cdots, h], i \geq j$.

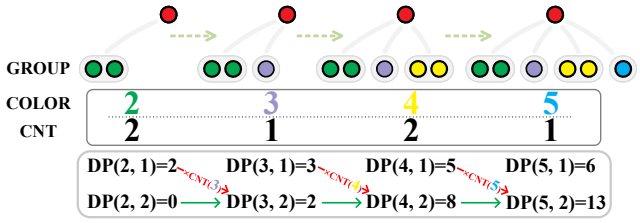


Fig. 4. Illustration of the computation for v_5's colorful 3-star degree.

The base cases can be set as follows:

$$\begin{cases} \mathsf{DP}(i,0)=1, & \text{for all } i \in [0,\cdots,\chi] \\ \mathsf{DP}(i,j)=0, & \text{for all } i \in [0,\cdots,\chi], j \in [0,\cdots,h], i<j, \end{cases} \tag{2}$$

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

Example 3. *Fig. 4 shows the computational procedure of v_5's colorful 3-star degree. The DP algorithm computes $d_{v_5}(G,\mathcal{S})$ 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". 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 $d_{v_5}(G,\mathcal{S})=\mathsf{DP}(5,3-1)=13$ can be computed in the similar method following an increasing order of color values as shown in Fig. 4.*

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; and slightly modify the DP recursive equation as:

$$\mathsf{DP}(j)=\mathsf{DP}(j-1)\times\mathsf{cnt}(i)+\mathsf{DP}(j), \tag{3}$$

for all $i\in[0..\chi], j\in[0..h], i\geq j$. Note that the two $\mathsf{DP}(j)$ on the left and right hand side of the equation have different meanings. The $\mathsf{DP}(j)$ on the left hand side indicates $\mathsf{DP}(i,j)$ of Eq. (1) ; the $\mathsf{DP}(j-1)$ and $\mathsf{DP}(j)$ on the right hand side are old values, which indicate $\mathsf{DP}(i-1,j-1)$ and $\mathsf{DP}(i-1,j)$ respectively. After χ-round iterative calculation, we can obtain $\mathsf{DP}(h-1)$ which is equal to $d_u(G,\mathcal{S})$. Note that 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 Eq. (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 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 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 $\mathsf{cnt}(\mathsf{color}(v))$, 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,

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

Here $\mathcal{DP}(i)$ denotes 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*

$$\begin{cases} \mathcal{G}(i)\leftarrow\mathcal{F}(i-1)\times\mathsf{cnt}(\mathsf{color}(v)) \\ \mathcal{DP}(i)\leftarrow\mathcal{F}(i)+\mathcal{G}(i) \end{cases} \tag{4}$$

for all $i\in[1,\cdots,h]$. The updated $d_u(G\backslash v,\mathcal{S})$ is equal to $\mathcal{DP}(h-1)$.

Proof. First, we set χ' as $\mathsf{color}(v)$. Apparently, $\mathcal{G}(i)$ can be derived by $\mathcal{F}(i-1)\times\mathsf{cnt}(\mathsf{color}(v))$, because the current

Algorithm 2: The Counting and Updating Algorithm

Input: A graph G, a node u and its neighbor v
Output: The updated colorful h-star degree $d_u(G \backslash v, \mathcal{S})$

1 $d_u(G, \mathcal{S}) \leftarrow \mathsf{Counting}(u)$;
2 Delete v from G;
3 $d_u(G \backslash v, \mathcal{S}) \leftarrow \mathsf{Updating}(\mathcal{DP}, v)$;

4 **Procedure** $\mathsf{Counting}(u)$
5 $\mathcal{F}(\cdot) \leftarrow 0; \mathcal{F}(0) \leftarrow 1$;
6 $\chi' \leftarrow 1$;
7 **for** $i = 2$ **to** χ **do**
8 **for** $j = h$ **to** 1 **do**
9 $\mathcal{F}(j) \leftarrow \mathcal{F}(j-1) \times \mathsf{cnt}(i) + \mathcal{F}(j)$;
10 **for** $j = 1$ **to** h **do**
11 $\mathcal{G}(j) \leftarrow \mathcal{F}(j-1) \times \mathsf{cnt}(1)$;
12 $\mathcal{DP}(j) \leftarrow \mathcal{F}(j) + \mathcal{G}(j)$;
13 **return** $\mathcal{DP}(h-1)$;

14 **Procedure** $\mathsf{Updating}(\mathcal{DP}, v)$
15 $\chi' \leftarrow \mathsf{color}(v), \mathcal{F}(0) \leftarrow 1$;
16 **for** $i = 1$ **to** h **do**
17 $\mathcal{G}(i) \leftarrow \mathcal{F}(i-1) \times \mathsf{cnt}(\mathsf{color}(v))$;
18 $\mathcal{F}(i) \leftarrow \mathcal{DP}(i) - \mathcal{G}(i)$;
19 $\mathsf{cnt}(\mathsf{color}(v)) \leftarrow \mathsf{cnt}(\mathsf{color}(v)) - 1$;
20 **for** $i = 1$ **to** h **do**
21 $\mathcal{G}(i) \leftarrow \mathcal{F}(i-1) \times \mathsf{cnt}(\mathsf{color}(v))$;
22 $\mathcal{DP}(i) \leftarrow \mathcal{F}(i) + \mathcal{G}(i)$;
23 **return** $\mathcal{DP}(h-1)$;

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 $\mathcal{DP}(i) = \mathcal{F}(i) + \mathcal{G}(i)$. □

Our algorithm is outlined in Algorithm 2. First, in the counting phase, we re-design a DP algorithm to compute the colorful h-star degree of u by using the arrays $\mathcal{F}$, $\mathcal{G}$, and DP as defined in Eq. (4). 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 8-9). After that, based on the $\mathcal{F}$ array, we can compute $\mathcal{G}$ and $\mathcal{DP}$ by Eq. (4) (lines 10-12). Note that because the color value χ' can be determined arbitrarily, we fix $\chi' = 1$ for the sake of simplicity (lines 6-7). It is easy to see that both the time and space complexity of the re-designed DP procedure are the same as those of Algorithm 1.

Second, 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 Eq. (4) to restore $\mathcal{G}$ and $\mathcal{F}$ by $\mathcal{DP}$. A difference compared to the counting stage is that χ' must be set to $\mathsf{color}(v)$ (line 15), because only the number of colorful h-stars colored by $\mathsf{color}(v)$ will reduce. The proposed updating technique consists of three steps, restoring (lines 16-18), updating (line 19) and regenerating (lines 20-22).

S1. In the restoring step, our algorithm restores $\mathcal{F}(i)$ and $\mathcal{G}(i)$ on the basis of $\mathcal{DP}(i)$ and $\mathsf{cnt}(\mathsf{color}(v))$.

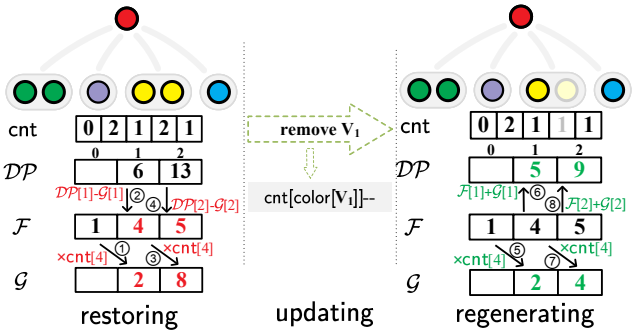


Fig. 5. Illustration of updating v_5's colorful 3-star degree.

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 4. *For the graph G in Fig. 2(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. 5 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 3-star degree $d_{v_5}(G, \mathcal{S}) = \mathcal{DP}(2)$ will be updated to 9 when the algorithm terminated.*

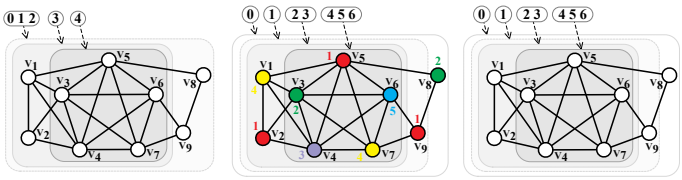
The correctness of Algorithm 2 can be guaranteed by Theorem 2. A nice feature of our update technique is that both the restoring and recomputing steps consume $O(h)$ time, which is much more efficient than the straightforward re-computation based updating algorithm. The following theorem shows the time 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 $O(\min\{\chi, d_u(G)\} + h)$ 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 $O(\min\{\chi, d_u(G)\} + h)$ space for each node. This is because the size of the array cnt is bounded by $O(\min\{\chi, d_u(G)\}$, while $\mathcal{DP}$ consumes $O(h)$ space. □

IV. Colorful h-star Core Decomposition

In this section, we propose a novel higher-order cohesive subgraph model, called colorful h-star k core, also referred to as $(k, \mathcal{S})$-core, based on the concept of colorful h-star. Similar to the classical core decomposition on graph, we also develop a near-linear peeling algorithm for colorful h-star core decomposition based on the proposed colorful h-star counting and updating techniques.



(a) k-cores (b) $(k, \mathcal{S})$-cores (c) (k, Ψ)-cores

Fig. 6. k-core, $(k, \mathcal{S})$-core and (k, Ψ)-core (Ψ: 3-clique; $\mathcal{S}$: colorful 3-star).

A. The colorful h-star core model

We start by reviewing the definition of the classical k-core [21], and then introduce the concept of colorful h-star core. Given a graph G and an integer k, a k-core (or core of order k), 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., $d_u(C_k) \geq k$ for every $u \in C_k$ [21]. 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 [21]. The maximum core number of a graph G, denoted by δ, is the maximum value of core numbers among all nodes in G. The maximum core number δ is also referred to as the degeneracy of G [29].

Example 5. *Fig. 6(a) depicts k-cores of the graph in Fig. 2(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.*

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) $d_u \geq c_u$; (d) the k-core is unique and the core decomposition can be computed in $O(m + n)$ time [30].

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

Definition 3 (Colorful h-star k core)**.** *Given a colored graph G, an integer k and the size h of a colorful star $\mathcal{S}$, a colorful h-star k core, or $(k, \mathcal{S})$-core of G, denoted by C^s_k, is a maximal subgraph H such that $\forall u \in V_H$, $d_u(H, \mathcal{S}) \geq k$.*

Based on Definition 3, the colorful h-star core number of u, denoted by $c_u(G, \mathcal{S})$, is the largest k such that there exists a colorful h-star k core containing u. The maximum colorful h-star core number of a graph G, denoted by $\delta^{\mathcal{S}}$, is the maximum value of colorful h-star core numbers among all nodes.

Example 6. *Let $\mathcal{S}$ be the colorful 3-star. Fig. 6(b) shows all $(k, \mathcal{S})$-cores of the graph. The number k in each rectangle indicates the $(k, \mathcal{S})$-core contained in that rectangle. For example, the subgraph induced by $\{v_3, v_4, v_5, v_6, v_7\}$ is the $(6, \mathcal{S})$-core since each node in the 5-clique participates in 6 colorful 3-stars. We can see that k-cores and $(k, \mathcal{S})$-cores are the same for $k = 3, 4$, but obviously different when $k = 1, 2$. Note that the entire graph is a $(0, \mathcal{S})$-core.*

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

Algorithm 3: The Colorful h-Star Core Decomposition

Input: A graph G and an integer h
Output: $c_v(G, \mathcal{S})$ for each node $v \in V$
1 **for** $u = 1$ **to** $|V|$ **do**
2 $d_u(G, \mathcal{S}) \leftarrow$ Counting(u); // Algorithm 2
3 Sort nodes of G in a non-decreasing order of their colorful h-star degrees;
4 $H \leftarrow G$; max_core $\leftarrow 0$;
5 **for** $i = 1$ **to** $|V|$ **do**
6 $u \leftarrow \arg\min_{v \in V_H} d_v(H, \mathcal{S})$;
7 **if** $d_u(H, \mathcal{S}) >$ max_core **then**
8 max_core $\leftarrow d_u(H, \mathcal{S})$;
9 $c_u(G, \mathcal{S}) \leftarrow$ max_core;
10 **for** *each* $w \in N_u(H)$ **do**
11 $d_w(H \backslash u, \mathcal{S}) \leftarrow$ Updating$(\mathcal{DP}, w)$; // Algorithm 2
12 Delete u from H;
13 Resort the nodes of H;
14 **return** $c_u(G, \mathcal{S})$ for each node $v \in V$;

B. The colorful h-star core decomposition algorithm

Based on Definition 3, we define the colorful h-star core decomposition problem as follows.

Problem 1 (Colorful h-star Core Decomposition)**.** *Given a graph G and an integer h, the colorful h-star core decomposition of G is a problem of computing the colorful h-star core numbers for all nodes in G.*

It is easy to derive that a $(k, \mathcal{S})$-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 (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 $c_u(G, \mathcal{S})$ for each node $v \in V$ (line 14). Obviously, δ^s 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 $O(h \times m)$ time using $O(hn + m)$ space.*

Proof. The graph G can be colored in linear time by using the greedy coloring algorithm [26], [27]. For each node u, we need to compute colorful h-star degree of u, i.e., $d_u(G, \mathcal{S})$, which takes $O(h \times \chi)$ 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 $O(h \times \min\{d_u, \chi\})$. Thus, the total time overhead for calculating the colorful h-star degrees for all nodes is bounded by $O(h \times m)$. Similar to the linear-time core decomposition algorithm, we can employ a bucket sort technique [31] 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 $O(h \times m)$. 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 $d_u(G, \mathcal{S}) = d_u(G)$. In this case, the colorful 2-star core decomposition turns into the classical core decomposition. The core decomposition takes $O(m)$ time using $O(m + n)$, which is consistent with our results.

V. Speed 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 [23], this problem may be NP-hard for a large h. Because the number of h-cliques in a graph is exponential in the size h, and counting h-clique is often infeasible when h is large. Thus, in this paper, we focus mainly on the approximation algorithms. Below, we will discuss two state-of-the-art approximation algorithms (Section V-B) to solve this problem. Then, 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 V-C).

A. The h-clique densest subgraph

Given a graph G, an h-clique Ψ is a subgraph with h nodes such that each pair of nodes is connected with an edge. The h-clique number of G, denoted by $c_h(G)$, is the number of h-cliques in G. We also denote $d_u(G, \Psi)$ to the number of h-cliques that u participates in.

Definition 4 (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 $\sigma_h(H) = \frac{c_h(H)}{|V_H|}$.*

Problem 2 (h-Clique-DS-Problem)**.** *Given a graph G and an integer h, find a subgraph H^* that achieves the largest h-clique density among all subgraphs of G, and let $\sigma_h^* = \sigma_h(H^*)$.*

Fang et al. [22] introduced a concept of h-clique core, which can help achieve a good approximation to the h-Clique-DS-Problem.

Definition 5 (h-clique Core)**.** *([22]) Given a graph G, an integer k, and an h-clique Ψ, the h-clique k core, or (k, Ψ)-core of G, denoted by C_k^Ψ, is a maximal subgraph such that $\forall u \in C_k^\Psi$, $d_u(C_k^\Psi, \Psi) \geq k$.*

We denote the h-clique core number of a node $u \in V$ by $c_u(G, \Psi)$, 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 δ^Ψ, is the maximum value of h-core numbers among all nodes.

Example 7. *Let Ψ be the 3-clique. Fig. 6(c) shows all (k, Ψ)-cores of the graph. The number k in each rectangle indicates the (k, Ψ)-core contained in that rectangle. For example, the subgraph induced by $\{v_3, v_4, v_5, v_6, v_7\}$ is the $(6, \Psi)$-core, since each node in the 5-clique participates in six 3-cliques. Also, $c_{v_1}(G, \Psi) = 3$ as v_1 is contained in the $(3, \Psi)$-core and is not a member of $(4, \Psi)$-core.*

The following theorem shows that the (δ^Ψ, Ψ)-core is a good approximation of the h-clique densest subgraph [22].

Theorem 5. *([22]) Given a graph G and an h-clique Ψ, the (δ^Ψ, Ψ)-core $C_{\delta^\Psi}^\Psi$ is a $\frac{1}{h}$-approximation solution to h-clique densest subgraph problem, such that $\sigma_h(C_{\delta^\Psi}^\Psi) \geq \frac{1}{h} \times \sigma_h^*$.*

B. Existing approximation algorithms and their limitations

Core-based approximation algorithm. Fang et al. established lower and upper bounds on the h-clique density for each (k, Ψ)-core [22]. Based on these bounds, they computed the upper and lower bounds of σ_h^*. They also found that the h-clique densest subgraph is located in some specific (k, Ψ)-core and the (δ^Ψ, Ψ)-core is a $\frac{1}{h}$-approximation solution. Therefore, to obtain the (δ^Ψ, Ψ)-core, 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 (δ^Ψ, Ψ)-core directly based on the observation that nodes in (δ^Ψ, Ψ)-core tend to have higher h-clique degrees. The main defect of CoreApp is that it needs to compute the (δ^Ψ, Ψ)-core 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 (δ^Ψ, Ψ)-core, considering those unpromising nodes in the computational procedure causes much unnecessary time consumption.

Sampling-based approximation algorithm. Sun et al. proposed a sampling based algorithm, called SeqSamp++, which can obtain an approximation of the h-clique densest subgraph [25]. 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 $p = \frac{\sigma}{c_h(G)}$, 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.

C. The colorful h-star core based algorithm

The (δ^Ψ, Ψ)-core achieves a $\frac{1}{h}$-approximation to the h-Clique-DS-Problem, but computing the (δ^Ψ, Ψ)-core on

Algorithm 4: The Colorful h-Star Core Reduction

Input: A graph G and an integer h
Output: The $(\theta, \mathcal{S})$-core
1 $\Psi \leftarrow$ compute a large clique using a greedy algorithm proposed in [32];
2 $\omega \leftarrow |V_\Psi|, \theta \leftarrow c_h(\Psi) = \binom{\omega-1}{h-1}$;
3 $C_{w-1} \leftarrow$ compute the $(\omega-1)$-core using the peeling algorithm [30];
4 $(\theta, \mathcal{S})$-core $\leftarrow$ ColorfulStarCore(C_{w-1}, h, θ);
5 **return** $(\theta, \mathcal{S})$-core;

6 **Procedure** ColorfulStarCore (H, h, k)
7 **for** $u = 1$ **to** $|V_H|$ **do**
8 $\quad d_u(H, \mathcal{S}) \leftarrow$ Counting(u);
9 Let Q be an empty queue;
10 **for** *each* $v \in H$ **do**
11 $\quad$ **if** $d_u(H, \mathcal{S}) < k$ **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 $\quad\quad d_v(H \backslash u, \mathcal{S}) \leftarrow$ Updating$(\mathcal{DP}, v)$;
17 $\quad\quad$ **if** $d_v(H \backslash u, \mathcal{S}) < k$ **then**
18 $\quad\quad\quad$ Push v to Q;
19 $\quad$ Delete u from H;
20 **return** H;

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 (δ^Ψ, Ψ)-core 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 $\theta = \binom{w-1}{h-1}$. 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 $\delta^\Psi \geq \theta$. 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. [32] to find a large clique.

The basic pruning rule. Suppose that we have a large clique with size w computed by the greedy algorithm [32]. Then, we have the following result.

Theorem 6. *Given a graph G, and an h-clique Ψ, the h-clique θ core C_θ^Ψ is contained in the $(w-1)$-core, where w is the size of a large clique of G and $\theta = \binom{w-1}{h-1}$.*

Proof. For each node $v \in C_\theta^\Psi$, the following inequalities hold

$$\binom{w-1}{h-1} = c_v(C_\theta^\Psi, \Psi) \leq d_v(C_\theta^\Psi, \Psi) \leq d_v(C_\theta^\Psi, \mathcal{R}) = \binom{d_v(C_\theta^\Psi)}{h-1}.$$

This provides a lower bound $d_v(C_\theta^\Psi) \geq w-1$ of the degree, indicating that C_θ^Ψ must be contained in the $(w-1)$-core. □

The above analysis inspires us to firstly reduce the input graph G to a $(w-1)$-core based on the fact that the (δ^Ψ, Ψ)-core is a subgraph of the $(w-1)$-core. Although shrinking the graph G to a $(w-1)$-core eliminates a number of nodes with core number less than $w-1$, the remaining graph may still be very large, because the $(w-1)$-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 $(w-1)$-core. We can achieve this by applying our colorful h-star core decomposition to further remove the unnecessary nodes for computing the (δ^Ψ, Ψ)-core.

The advanced pruning rule. Reconsider the colored graph G and the h-clique θ core C_θ^Ψ. Clearly, each node of C_θ^Ψ 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 C_θ^Ψ must be contained in the colorful h-star θ core $C_\theta^\mathcal{S}$. 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 $(w-1)$-core pruning strategy.

Theorem 7. *Given a graph G, and an integer h, the colorful h-star θ core $C_\theta^\mathcal{S}$ is contained in the $(w-1)$-core of G, where w is the size of a large clique of G, and $\theta = \binom{w-1}{h-1}$.*

Proof. Similar to the proof of Theorem 6, for each node $v \in C_\theta^\mathcal{S}$, the following inequalities hold

$$\binom{w-1}{h-1} = c_v(C_\theta^\mathcal{S}, \mathcal{S}) \leq d_v(C_\theta^\mathcal{S}, \mathcal{S}) \leq d_v(C_\theta^\mathcal{S}, \mathcal{R}) = \binom{d_v(C_\theta^\mathcal{S})}{h-1}.$$

As a consequence, we have $d_v(C_\theta^\mathcal{S}) \geq w-1$, thus $C_\theta^\mathcal{S}$ is an induced subgraph of C_{w-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

$$G \supseteq C_{w-1} \supseteq C_\theta^\mathcal{S} \supseteq C_{\delta^\Psi}^\Psi.$$

Our colorful h-star core based graph reduction technique is shown in Algorithm 4. Algorithm 4 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 4 is $O(h \times m)$. This is because the lines 1-3 takes $O(m)$ time, and the computing of the colorful h-star θ core uses $O(h \times m)$ 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 δ^Ψ core has been proved to be a $\frac{1}{h}$-approximation of H-CLIQUE-DS-PROBLEM. Therefore, the colorful h-star $\delta^\mathcal{S}$ core should also provide a good approximation solution. That is to say, we can obtain a heuristic algorithm by just computing the $(\delta^\mathcal{S}, \mathcal{S})$-core on G using Algorithm 3 as an approximation of the h-clique densest subgraph. Although such a heuristic algorithm is without theoretical guarantees, it can achieve similar and even better approximation performance compared to the state-of-the-art approximation solution, using much less time as confirmed in our experiments.

VI. 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)

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

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 the greedy coloring algorithm and the maximum degree of the graph respectively.

Algorithms. For the colorful h-star core decomposition, we implement two algorithms HStarDP and HStarCD. The only difference between them is that after removing a node, HStarDP recomputes the colorful h-star degrees of its neighbors using the proposed DP algorithm, while HStarCD uses the proposed updating technique to update the colorful h-star degrees. Since there are no other algorithms that can be used to compute the colorful h-star core decomposition, HStarDP is severed as a baseline for comparison. For the h-clique densest subgraph problem, we compare our algorithms with three state-of-the-art approximation algorithms which are (1) SeqSamp [25], (2) SeqSamp128 [25], and (3) CoreApp [22], and an exact solution Exact [25] using the max-flow technique. Since the approximation performance of SeqSamp++ relies on the number of iterations [25], we implement two versions of SeqSamp++ with the number of iterations $T = 1$ and $T = 128$ for comparison, denoted by SeqSamp and SeqSamp128 respectively. CoreApp is the h-clique core based approximation algorithm which can obtain a $1/h$ approximation [22]. For these three baseline algorithms, we use the original C++ implementations in our experiments. We implement our two algorithms: (1) HStarPP which first performs our colorful h-star core based pruning on G, and then calls CoreApp on the reduced graph; and (2) HStarMPP which is the heuristic approach using the $(\delta^{\mathcal{S}}, \mathcal{S})$-core as an approximation solution. Instead of getting the $(\delta^{\mathcal{S}}, \mathcal{S})$-core by performing HStarCD on G, we also implement a subroutine HStarMB which computes the core directly by using binary search. HStarMB tests a possible core value each time to find $\delta^{\mathcal{S}}$, and adjusting its lower bound and upper bound according the results of tests. When the subroutine terminates, it will return $\delta^{\mathcal{S}}$ and a subgraph as the $(\delta^{\mathcal{S}}, \mathcal{S})$-core. We implement all our algorithms in C++. All experiments are conducted on a Linux machine equipped with a 2.9GHz AMD Ryzen 3990X CPU and 256GB RAM running CentOS 7.9.2 (64-bit).

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.

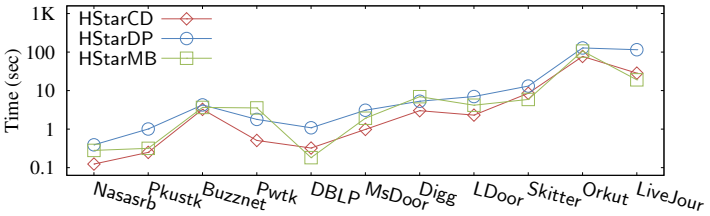
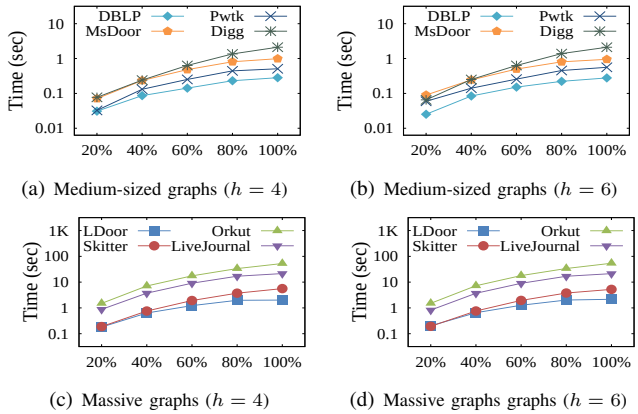


Fig. 7. Running time of HStarCD, HStarDP and HStarMB ($h = 6$)



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

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

Fig. 8. Scalability of HStarCD

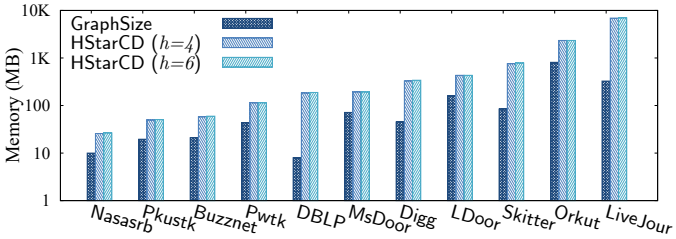


Fig. 9. Memory overhead of HStarCD on different datasets ($h = 4, 6$)

B. Results of the colorful h-star core decomposition

Runtime of different algorithms. In this experiment, we compare the running time of HStarDP, HStarCD and HStarMB on different datasets. Fig. 7 shows the results for $h = 6$. For other h values, the results are consistent. From Fig. 7, we can see that HStarCD is 2 to 7 times faster than HStarDP over all datasets. For example, on LiveJournal, the total running time of HStarDP is 200.3 seconds, while the HStarCD algorithm equipped with the proposed updating technique takes only 53.2 seconds, which is around 4 times faster than HStarDP. These results confirm our theoretical analysis shown in Sections III and IV. Note that the performance of HStarMB is comparable with that of HStarCD, and sometimes even slightly worse than HStarCD. The reasons could be that HStarCD is already very fast as its time complexity is $O(hm)$, while the binary-search method has an additional log factor although it does not need to compute all the core numbers.

Scalability. Here we evaluate the scalability of HStarCD on 8 different datasets. Note that since HStarDP is clearly worse than HStarCD, we do not show the results of HStarDP in all the remaining experiments. 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%.

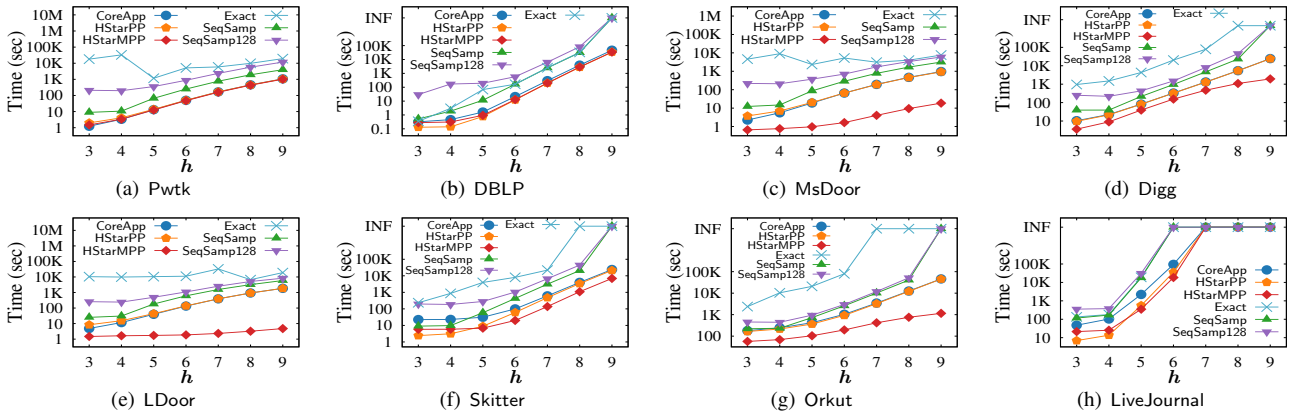


Fig. 10. Running time of different algorithms with varying h from 3 to 9

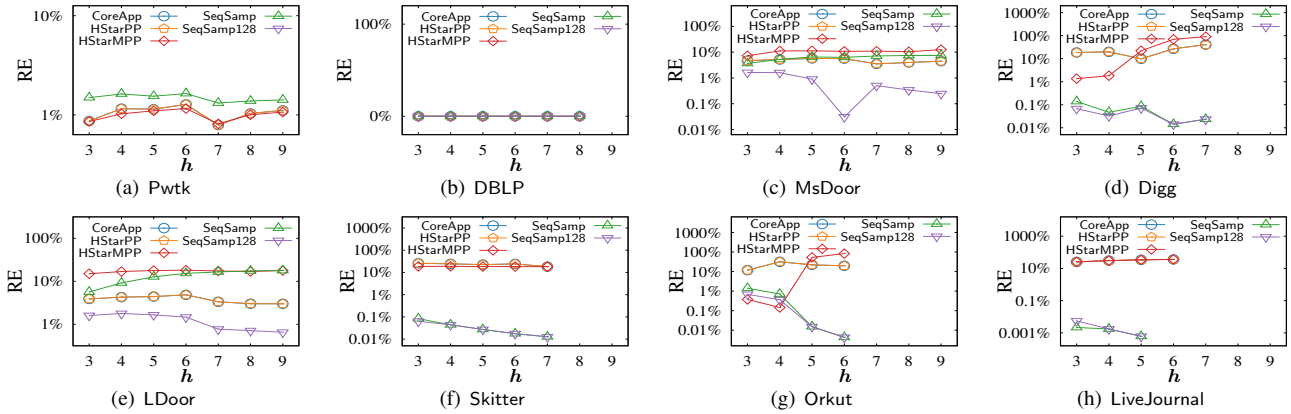


Fig. 11. Relative error (RE) of different algorithms with varying h from 3 to 9 (Points are missing where Exact runs out of 24 hours.)

Fig. 8 shows the results of $h = 4$ and $h = 6$, and similar results can also be observed with other parameters. As can be seen, the running time of HStarCD 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 HStarCD algorithm (i.e., its complexity is $O(h \times m)$).

Memory overhead. We evaluate the memory usage of our HStarCD algorithm on various datasets for $h = 4$ and $h = 6$ respectively. The results are shown in Fig. 9. Similar results can also be observed for the other values of h. From Fig. 9, we can see that the memory cost of our algorithm is insensitive w.r.t. h on all datasets. This is because $O(hn)$ 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 is around 10 times higher than the graph size, while for the other datasets, the space usage of HStarCD is only slightly larger than the graph size. These results confirm the near-linear space complexity of the HStarCD algorithm.

C. *Results of the* H-CLIQUE-DS-PROBLEM

In this subsection, we carry out a set of experiments to evaluate the performance of five different algorithms for solving the H-CLIQUE-DS-PROBLEM.

Running time. We plot the running time achieved by each algorithm as a function of h varying from 3 to 9 on 4 datasets in Fig. 10. The results on the other datasets are consistent. As expected, the running time of all the five approximation algorithms increases as h increases. We can clearly see that both the colorful h-star core based algorithm (HStarPP) and the heuristic approach (HStarMPP) are significantly faster than the other competitors, and Exact takes the most time on almost all datasets. In addition, we can also observe that HStarMPP seems to be slightly faster than all other algorithms (especially on LDoor and Orkut), but a little slower than HStarPP on LiveJournal with $h = 3, 4$. The reason is that HStarMPP needs to compute the entire colorful h-star core decomposition, thus for a small h, HStarPP may be slightly faster than HStarMPP. We can see that on LDoor, CoreApp is already very fast for a small h, thus in this case it cannot be significantly speeded up by our algorithms. Note that on most 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. Taking the Skitter dataset as an example (Fig. 10(f)), SeqSamp, SeqSamp128 and CoreApp consume 58.40, 279.42 and 33.14 seconds respectively, when $h = 5$. However, under the same parameter setting, HStarPP and HStarMPP take only 6.89 and 4.81 seconds respectively. The HStarPP algorithm, for instance, improves the running time over SeqSamp, SeqSamp128 and CoreApp by 8.5×,

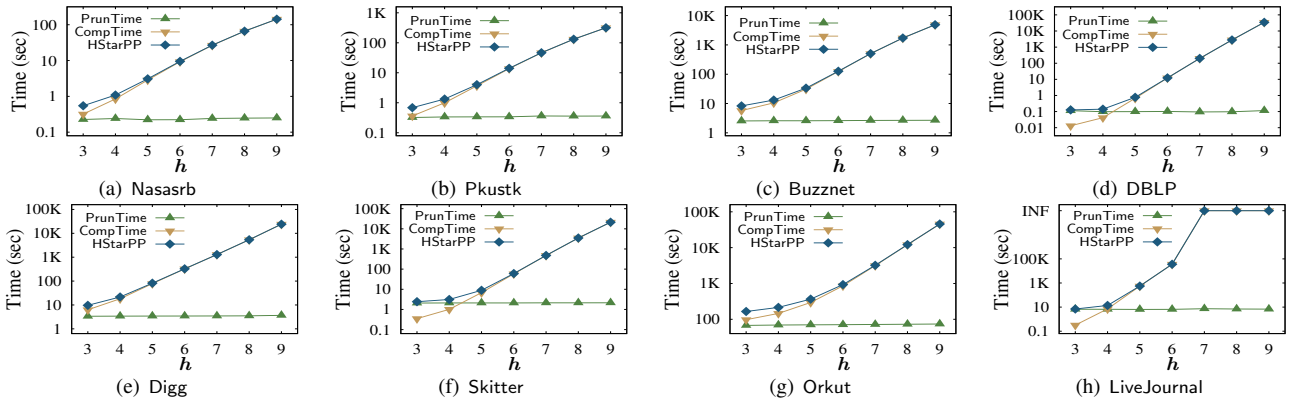


Fig. 12. The running time of HStarPP with varying h

TABLE III
THE POWER OF PRUNING TECHNIQUES USED IN HStarPP ($h = 6$, 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	$C^{\mathcal{S}}_{\theta}$	$C^{\Psi}_{\delta^{\Psi}}$	G	$C^{\mathcal{S}}_{\theta}$	$C^{\Psi}_{\delta^{\Psi}}$	G	$C^{\mathcal{S}}_{\theta}$	$C^{\Psi}_{\delta^{\Psi}}$	$C^{\mathcal{S}}_{\theta}$	G	$C^{\mathcal{S}}_{\theta}$	$C^{\Psi}_{\delta^{\Psi}}$
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

$40.5\times$ and $4.8\times$, respectively. HStarMPP is even one order of magnitude faster than CoreApp and SeqSamp, and nearly two orders of magnitude faster than SeqSamp128. The reason is that our colorful h-star core based techniques are very efficient for a relatively large value of h. These results confirm that the proposed graph reduction technique is very effective in speeding up the approximate h-clique densest subgraph computation.

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. 11 shows the relative errors of various algorithms on 4 different datasets. Similar results can be observed on the other datasets. Some points are missing on large h values where the exact algorithm cannot get a solution with 24 hours. Note that CoreApp and HStarPP achieve the same relative error because they both return the (δ^{Ψ}, Ψ)-core as an approximation. From Fig. 11, we can see that the core-based algorithms CoreApp and HStarPP can be more accurate than SeqSamp on LDoor for all h values. The heuristic algorithm HStarMPP also achieves a good approximation as CoreApp, and it performs even better on Orkut for $h = 3, 4$. Although SeqSamp128 can achieve a lower relative error, it is very costly and cannot provide an approximation for a large h, e.g., $h = 6$ on LiveJournal and $h = 9$ on Skitter and Orkut. These results indicate that (1) SeqSamp128 is very accurate on most datasets, but it is intractable on large graphs for large h values; (2) the colorful h-star core based algorithms HStarPP and HStarMPP produce high-quality results using much less time.

The performance of HStarPP. Here we evaluate HStarPP in terms of the running time distribution and the graph size reduction for $h = 6$ on Orkut and LiveJournal. Recall that HStarPP first prunes the graph using the colorful h-star θ core $C^{\mathcal{S}}_{\theta}$, and then calls CoreApp to compute the h-clique δ^{Ψ} core $C^{\Psi}_{\delta^{\Psi}}$ to achieve a $\frac{1}{h}$-approximation solution. The runtime of HStarPP includes the pruning time and the time spent on computing $C^{\Psi}_{\delta^{\Psi}}$, denoted by PrunTime and CompTime respectively. Fig. 12 shows the time distribution of HStarPP 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 HStarPP is mainly dominated by computing $C^{\Psi}_{\delta^{\Psi}}$, and the pruning procedure is very efficient.

Table III shows the statistics of G , $C^{\mathcal{S}}_{\theta}$ and $C^{\Psi}_{\delta^{\Psi}}$. In Table III, $\Delta = (n_1 - n_2)/n_1$ and σ_6 is 6-clique density where $n_1 = |V(G)|$, $n_2 = |V(C^{\mathcal{S}}_{\theta})|$ which can be used to measure the effectiveness of the pruning rule in HStarPP. 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 $C^{\mathcal{S}}_{\theta}$, 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 $C^{\mathcal{S}}_{\theta}$ with only 483 nodes, which is quite close to the target subgraph $C^{\Psi}_{\delta^{\Psi}}$ (385 nodes). In addition, $C^{\mathcal{S}}_{\theta}$ 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 $C^{\Psi}_{\delta^{\Psi}}$, suggesting that our colorful h-star core is indeed very cohesive. Also, we can see that $C^{\mathcal{S}}_{\theta}$ can significantly increase the clique density. On LiveJournal and Nasasrb, $C^{\mathcal{S}}_{\theta}$

even achieves a higher clique density than $C^{\Psi}_{\delta^{\Psi}}$. These results indicate that our colorful h-star core model can also 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 [26]:

Degree: Coloring the nodes following a non-increasing ordering of degree (break ties by node ID) [33].

Degen: Coloring the nodes following an inverse degeneracy ordering [26]. Note that such an inverse degeneracy ordering can be easily obtained by reversing the node-deletion ordering of the core-decomposition algorithm [30].

FF: Coloring the nodes in the order they appear in the input graph representation [33].

SD: Coloring an uncolored node whose colored neighbor nodes use the largest number of distinct colors [34] .

The nodes' distribution. Fig. 13 shows the nodes' distribution of various colorful h-star k cores on Skitter for $h = 3$ 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. 13, most nodes have smaller colorful h-star degree and the maximum colorful h-star core number $\delta^{\mathcal{S}}$ 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. 13(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 HStarPP and HStarMPP, because Degree is much faster than Degen (it needs to compute k-core decomposition) and SD (it picks an uncolored node dynamically).

Fig. 14 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. 14(a)) and colorful 3-star cores (see Fig. 13(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. 13. 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.

The performance of HStarMPP. We evaluate the performance of HStarMPP in terms of the time consumption and relative errors on Skitter in Fig. 15. In Fig. 15(a), DecomTime and CompTime indicate the time cost taken for computing the colorful h-star core decomposition and computing h-clique density of $(\delta^{\mathcal{S}}, \mathcal{S})$-core, respectively. The detailed statistics

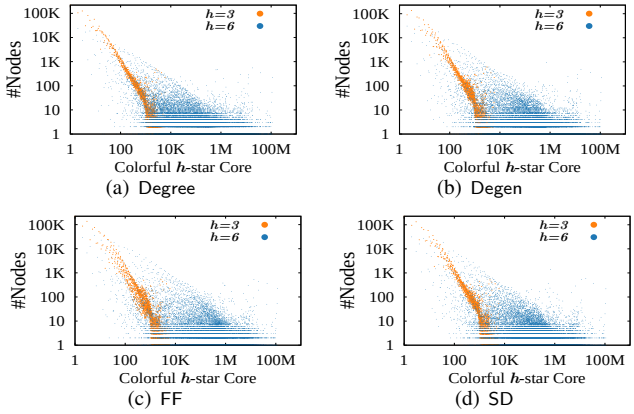


Fig. 13. Nodes' distributions of colorful h-star cores based on different graph colorings (Skitter)

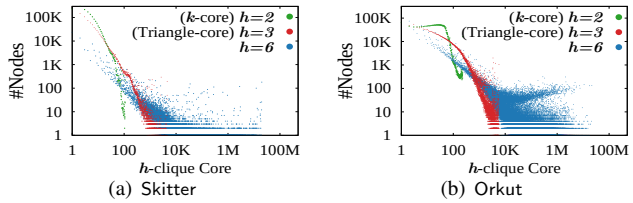


Fig. 14. Nodes' distributions of h-clique cores

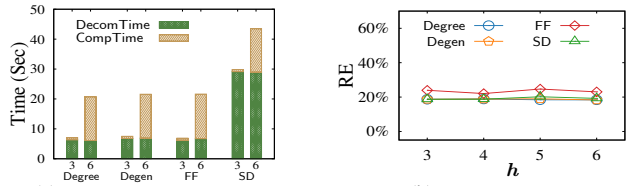
TABLE IV
PERFORMANCE OF DIFFERENT COLOR ALGORITHMS
(Skitter, $h = 6$, H IS THE $(\delta^{\mathcal{S}}, \mathcal{S})$-CORE)

	$n = \lvert V_H \rvert$	$m = \lvert E_H \rvert$	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

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 slightly higher quality colorings (using smaller number of colors in Table IV). For a large h value, CompTime increases significantly since computing the h-clique density involves counting the number of h-cliques which runs very slow. In Fig. 15(b), we can see that HStarMPP equipped with FF has a larger relative error. The results suggest that the estimating precision of HStarMPP 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. Overall, Degree is the best method in terms of both runtime and approximation quality. This is why we use Degree as a default coloring technique in the proposed solutions.

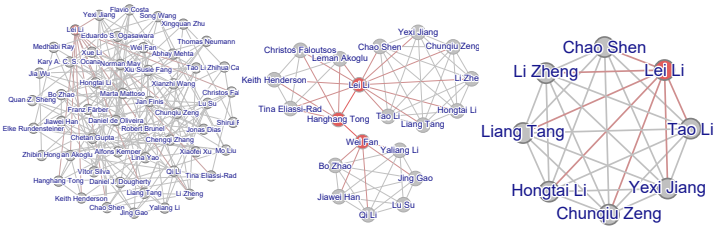
E. Case Studies.

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



(a) Running Time ($h = 3, 6$) (b) Relative Error

Fig. 15. Performance of HStarMPP with various graph coloring tehcniques



(a) k-core (b) $(k, \mathcal{S})$-cores (c) triangle densest subgraph

Fig. 16. The cohesive subgraphs found in DBLPs, based on k-core, $(k, \mathcal{S})$-cores and triangle densest subgraph ($\mathcal{S}$: colorful 3-star).

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. 16), respectively. As depicted in Fig. 16(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 $(k, \mathcal{S})$-core is quite different from the result of k-core (Fig. 16(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. 16(c) exhibits a different semantic. The result turns out to be a clique, which is also a subgraph of the $(k, \mathcal{S})$-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 might be better to reveal the close collaboration between different research groups.

VII. Related Work

Cohesive subgraph models. A large number of cohesive subgraph models have been proposed based on different cohesiveness measures [10]. Notable examples include k-core [21], k-truss [13], maximal k-edge connected subgraph [19], and maximal cliques [35]. *k-core* is a maximal subgraph among all subgraphs in G, in which the degree of each node is at least k [12], [21], [30]. Recently, such a concept was extended to uncertain graph [1], [36], attributed graphs [37], [38], and distance generalized cores [39]–[41]. A *k-truss* is a maximal subgraph where each edge participates in at least $k-2$ triangles [13], [14], [42]. 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 [19], [20], [43]. 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 [35]. Practical algorithms for maximal clique enumeration are often based on the classic Bron-Kerbosch algorithm [29], [35]. The concept of maximal clique was also extended to the context of uncertain graphs [44], [45]. In addition, there also exist some other cohesive subgraph models including *query-biased density* [3], [46], k-clique cores [22], and k-(r, s) nucleus [47]–[50] (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 [51]–[53]. 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 [51], [54]. Epasto et al. studied the densest subgraph computation in evolving graphs [52]. Qin et al. proposed an algorithm to find the top-k locally densest subgraphs [55]. The concept of h-clique densest subgraph (H-CLIQUE-DS) was first introduced by Tsourakakis [23] 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 [23], [53]. 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. [24] investigated a higher-order variant of locally dense subgraph [55] 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 [53]. Later in [22], 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 [25]. In this paper, we also study the h-clique densest subgraph problem. We propose a new graph reduction technique based on our colourful h-star core which is dramatically different from all the previous algorithms.

VIII. Conclusion

In this paper, we propose a new colorful h-star core model to identify cohesive subgraphs in large real-world graphs. To efficiently compute the colorful h-star cores, we develop a novel DP based colorful h-star degree counting and updating algorithms. We show that the colorful h-star core decomposition can be done in $O(hm)$ time using $O(hn+m)$ space. Based on our colorful h-star core, 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] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich, "Core decomposition of uncertain graphs," in *KDD*, 2014.

[2] J. Chen and Y. Saad, "Dense subgraph extraction with application to community detection," *IEEE Transactions on knowledge and data engineering*, vol. 24, no. 7, pp. 1216–1230, 2010.

[3] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli, "Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees," in *KDD*, 2013.

[4] X. Huang, L. V. Lakshmanan, and J. Xu, "Community search over big graphs: Models, algorithms, and opportunities," in *ICDE*, 2017.

[5] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, "Identification of influential spreaders in complex networks," *Nature Physics*, vol. 6, pp. 888–893, 2010.

[6] F. D. Malliaros, M.-E. G. Rossi, and M. Vazirgiannis, "Locating influential nodes in complex networks," *Scientific reports*, vol. 6, no. 1, pp. 1–10, 2016.

[7] M. Vazirgiannis, "Graph of words: Boosting text mining tasks with graphs," in *WWW Companion*, 2017.

[8] A. Tixier, F. Malliaros, and M. Vazirgiannis, "A graph degeneracy-based approach to keyword extraction," in *EMNLP*, 2016.

[9] A. Angel, N. Koudas, N. Sarkas, and D. Srivastava, "Dense subgraph maintenance under streaming edge weight updates for real-time story identification," *PVLDB*, vol. 5, no. 6, pp. 574–585, 2012.

[10] L. Chang and L. Qin, "Cohesive subgraph computation over large sparse graphs," in *ICDE*, 2019.

[11] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *SIGMOD*, 2014.

[12] 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, 2014.

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

[14] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," *SIGMOD*, 2014.

[15] A. Conte, D. Firmani, C. Mordente, M. Patrignani, and R. Torlone, "Fast enumeration of large k-plexes," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 115–124.

[16] M. Danisch, O. D. Balalau, and M. Sozio, "Listing k-cliques in sparse real-world graphs," in *WWW*, 2018.

[17] R. Li, S. Gao, L. Qin, G. Wang, W. Yang, and J. X. Yu, "Ordering heuristics for k-clique listing." *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2536–2548, 2020.

[18] S. Jain and C. Seshadhri, "The power of pivoting for exact clique counting," in *WSDM*, J. Caverlee, X. B. Hu, M. Lalmas, and W. Wang, Eds., 2020.

[19] 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 *EDBT*, 2012.

[20] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang, "Efficiently computing k-edge connected components via graph decomposition," in *SIGMOD*, 2013.

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

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

[23] C. E. Tsourakakis, "The k-clique densest subgraph problem," in *WWW*, 2015.

[24] R. Samusevich, M. Danisch, and M. Sozio, "Local triangle-densest subgraphs," in *ASONAM*, 2016.

[25] B. Sun, M. Danisch, T. Chan, and M. Sozio, "Kclist++: A simple algorithm for finding k-clique densest subgraphs in large graphs," *Proceedings of the VLDB Endowment (PVLDB)*, 2020.

[26] W. Hasenplaugh, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Ordering heuristics for parallel graph coloring," in *SPAA*, 2014.

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

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

[29] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," *ACM Journal of Experimental Algorithmics*, vol. 18, 2013.

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

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

[32] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, "Parallel maximum clique algorithms with applications to network analysis," *SIAM Journal on Scientific Computing*, vol. 37, no. 5, pp. C589–C616, 2015.

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

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

[35] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph (algorithm 457)," *Commun. ACM*, vol. 16, no. 9, pp. 575–576, 1973.

[36] 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 *ICDE*, 2019.

[37] R.-H. Li, L. Qin, J. X. Yu, and R. Mao, "Influential community search in large networks," *PVLDB*, vol. 8, no. 5, pp. 509–520, 2015.

[38] R. Li, L. Qin, F. Ye, J. X. Yu, X. Xiao, N. Xiao, and Z. Zheng, "Skyline community search in multi-valued networks," in *SIGMOD*, 2018.

[39] F. Bonchi, A. Khan, and L. Severini, "Distance-generalized core decomposition," in *SIGMOD*, P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska, Eds., 2019.

[40] Q. Dai, R. Li, L. Qin, G. Wang, W. Yang, Z. Zhang, and Y. Yuan, "Scaling up distance-generalized core decomposition," in *CIKM*, G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, Eds., 2021.

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

[42] X. Huang, L. V. S. Lakshmanan, J. X. Yu, and H. Cheng, "Approximate closest community search in networks," *PVLDB*, vol. 9, no. 4, pp. 276–287, 2015.

[43] T. Akiba, Y. Iwata, and Y. Yoshida, "Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction," in *CIKM*, 2013, pp. 909–918.

[44] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Mining maximal cliques from an uncertain graph," in *ICDE*, 2015.

[45] R. Li, Q. Dai, G. Wang, Z. Ming, L. Qin, and J. X. Yu, "Improved algorithms for maximal clique search in uncertain networks," in *ICDE*, 2019.

[46] J. Abello, M. G. Resende, and S. Sudarsky, "Massive quasi-clique detection," in *Latin American symposium on theoretical informatics*. Springer, 2002, pp. 598–612.

[47] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek, "Finding the hierarchy of dense subgraphs using nucleus decompositions," in *WWW*, 2015.

[48] A. E. Sariyuce and A. Pinar, "Fast hierarchy construction for dense subgraphs," *Proc. VLDB Endow.*, vol. 10, no. 3, pp. 97–108, 2016.

[49] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek, "Nucleus decompositions for identifying hierarchy of dense subgraphs," *TWEB*, vol. 11, no. 3, pp. 16:1–16:27, 2017.

[50] A. E. Sariyuce, C. Seshadhri, and A. Pinar, "Local algorithms for hierarchical dense subgraph discovery," *Proc. VLDB Endow.*, vol. 12, no. 1, pp. 43–56, 2018.

[51] A. V. Goldberg, *Finding a maximum density subgraph*. University of California Berkeley, 1984.

[52] A. Epasto, S. Lattanzi, and M. Sozio, "Efficient densest subgraph computation in evolving graphs," in *WWW*, 2015.

[53] 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 *KDD*, 2015.

[54] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, "A fast parametric maximum flow algorithm and applications," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 30–55, 1989.

[55] L. Qin, R. Li, L. Chang, and C. Zhang, "Locally densest subgraph discovery," in *KDD*, 2015.