Using Large Cliques for Hierarchical Dense Subgraph Discovery

Md Moniruzzaman Monir and Ahmet Erdem Sarıyüce(✉)

University at Buffalo, Buffalo, NY 14260, USA
{mdmoniru,erdem}@buffalo.edu

Abstract. Understanding the structure of dense regions in real-world networks is an important research area with myriad practical applications. Using higher-order structures (motifs), such as triangles, had been shown to be effective to locate the dense subgraphs. However, going beyond the triangle structure is computationally demanding and mostly overlooked in the past. In this work, we investigate the use of large cliques (up to 10 nodes) for dense subgraph discovery. Relying on the nucleus decomposition framework that finds hierarchical dense subgraphs, we introduce efficient implementations to instantiate the framework up to 10-cliques. We analyze various real-world networks and discuss the density pointers, dense subgraph distributions, and also the hierarchical relationships. We investigate the clique count distributions per vertex and report surprising behaviors that are not observed in the degree distributions. Our analysis shows that utilizing larger cliques can yield denser structures with more interesting hierarchical relations in several networks.

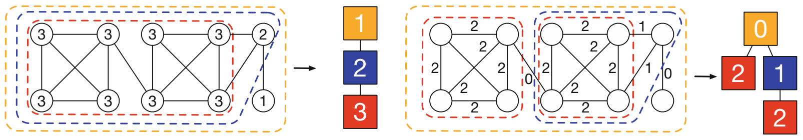
1 Introduction

Real-world networks have a sparse structure in the global level and contain dense regions in local neighborhoods [12]. Dense subgraphs are indicators for unusual behaviors and functional units. There are various applications, such as identifying the news stories from microblogging streams in real-time [2], finding price value motifs in the financial networks [8], detecting DNA motifs in biological networks [10], and locating spam link farms in web [7,11,14]. Dense regions are also used for visualization of complex graph structure [1,31].

Higher-order structures (or motifs) capture the network dynamics by considering a small set of nodes together. Triangle is the smallest non-trivial structure and has been heavily used in several models for network analysis [12]. However, considering larger structures is computationally expensive and thus mostly overlooked in the past. Previous works mostly focused on counting such structures by efficient heuristics [21] and sampling methods [13]. There are also some studies that finds a single optimum subgraph with respect to a given k-clique [28].

In this work, we analyze the impact of large cliques (up to 10 nodes) on the dense subgraph structure of networks. We use nucleus decomposition

S. Chellappan et al. (Eds.): CSoNet 2020, LNCS 12575, pp. 179–192, 2020.
https://doi.org/10.1007/978-3-030-66046-8_15



(a) k-cores **(b)** k-trusses

Fig. 1. Left depicts the k-core decomposition where core numbers are shown for each vertex with a hierarchy tree of different k-cores. Orange, blue, and red regions show the 1-, 2-, and 3-cores which are nested subgraphs, thus form the hierarchy by containment (1-core $\supseteq$ 2-core $\supseteq$ 3-core). Right shows the k-truss decomposition where truss numbers are shown for each edge with a hierarchy tree of different k-trusses. Entire graph is a 0-truss while five vertices in blue region form a 1-truss. There are also two 2-trusses and one of them is a subset of the 1-truss. (Color figure online)

[24,25], generalization of k-core [19,26] and k-truss decompositions [6,23,29,30], to find many subgraphs, of moderate density and with hierarchical relations, with respect to the large cliques. Informally, k-(r, s) nucleus, for fixed positive integers $r < s$, is a maximal subgraph where every r-clique participate in many s-cliques. Due to the computational challenges, existing implementations are bounded by (3, 4)-nucleus decomposition. We introduce practical algorithms for higher-order nucleus decompositions using large cliques and evaluate our algorithms through several experiments on a wide variety of real-world networks. We analyze various real-world networks and discuss the density pointers, dense subgraph distributions, and also the hierarchical relationships in a comparative way. Our statistical analyses of the density pointers and clique counts for the large (r, s) nuclei yield some interesting patterns. We observe that finer dense structures that are lost in larger structures can be identified by our algorithms, and there is an upper bound for higher-order nucleus decompositions where the results stop improving.

We consider a simple undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. We define k-clique as a complete graph among k vertices for $k > 0$, i.e., every vertex is connected to all other vertices. We first discuss k-core, k-truss, and nucleus decompositions.

k-core and k-truss Decompositions. k-core decomposition is a threshold-based hierarchical approach to decompose a network into nested subgraphs where the threshold k is set on the degree of a vertex to exploit the vertex-edge relationships. The idea of k-core was first introduced by Erdős and Hajnal [9] and rediscovered numerous times in different contexts [17,26]. k-core of G is a **maximal** and **connected** subgraph of G in which all nodes have degree at least k. The lowest possible value of k is 1 for any network because of the connectivity constraint. Core number of a node is the highest value of k such that it belongs to a k-core but not to any $(k + 1)$-core. Figure 1a shows an example for the core numbers of vertices and the hierarchy in k-core decomposition.

k-truss decomposition extends the idea of k-core decomposition by changing the focus from vertex-edge relationship to edge-triangle relationship. k-truss of

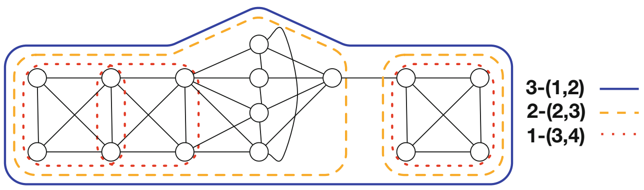


Fig. 2. Generalization of core and truss decompositions by nucleus decomposition. k-(1, 2) and k-(2, 3) nucleus decomposition represents k-core and k-truss, respectively. (1, 2) cannot detect a dense structure and reports the entire graph as a 3-(1, 2) nuclei while (2, 3) separates the 4-clique on the right and gives two 2-(2, 3) nuclei. However, (3, 4) can distinguish all the 4-cliques and report each as a 1-(3, 4) nuclei.

G is a **maximal** and **connected** subgraph of G in which every edge participates in at least k triangless [6,23,29,30]. Truss number of an edge is the highest value of k such that it belongs to a k-truss but not to any $(k+1)$-truss. Figure 1b shows an example for the truss numbers and the hierarchy in k-truss decomposition.

Nucleus Decomposition. k-core and k-truss decompositions are generalized by nucleus decomposition. It unifies the vertex-edge and edge-triangle relationships into r-clique and s-clique relationships where r, s are positive integers such that $r < s$. Informally, k-(r, s) nucleus of G is a **maximal** and **s-connected** subgraph of the r-cliques where each r-clique takes part in at least k s-cliques. We quote the definition of nucleus decomposition from [25].

Definition 1. *Let $r < s$ be positive integers.*

- *$R(G)$ and $S(G)$ are the set of r-cliques and s-cliques in G, respectively.*
- *s-degree of $R \in R(G)$ is the number of $S \in S(G)$ such that S contains R $(R \subset S)$.*
- *Two r-cliques R, R' are **s-connected** if there exists a sequence $R = R_1, R_2, ..., R_k = R'$ in $R(G)$ such that for each i, some $S \in S(G)$ contains $R_i \cup R_{i+1}$.*
- *Let k, r, and s be positive integers such that $r < s$. A k-(r, s) **nucleus** is a subgraph G' which contains the edges in the maximal union $S(G)$ of s-cliques such that*
  - *s-degree of any r-clique $R \in R(G')$ is at least k.*
  - *Any r-clique pair $R, R' \in R(G')$ are s-connected.*

As in the core and truss number definitions, we define the nucleus number of an r-clique, denoted k_s, as the largest k_s value such that the r-clique is a part of a k_s-(r, s) nucleus. Note that the values $r = 1$, $s = 2$ corresponds to the k-core and $r = 2$, $s = 3$ corresponds to the k-truss definition. Figure 2 shows a comparison of core, truss, and (3,4) nucleus subgraphs in a toy graph.

2 Algorithms and Implementation Details

In this section, we present the algorithms to generate nucleus decomposition using large cliques. Our implementation has two parts. First, we enumerate all k-cliques ($3 \leq k \leq 10$) of a graph for higher-order nucleus decompositions and save those cliques in disk. In the second part, we implement (r, s)-nucleus decomposition where $3 \leq r \leq 9$ and $s = r + 1$. As our main algorithmic goal is to construct the forest of nuclei using large cliques, we extend the previous framework of (3, 4)-nucleus decomposition introduced in [25]. But an extension of this framework for higher-order nucleus decompositions is non-trivial as the number of cliques grows exponentially for large values and managing those efficiently requires a careful implementation. Implementations for higher-order decompositions up to $(9, 10)$ nucleus available at http://sariyuce.com/largeND.zip.

2.1 Large Clique Enumeration

In Algorithm 1, we generate all k-cliques of a graph. We use **MACE** (MAximal Clique Enumerator) [18] code to enumerate all maximal cliques. This code takes O($|V||E|$) time for each maximal clique, where $|V|$ and $|E|$ are the number of vertices, and the number of edges in the input graph. In practice the code finds about 100,000 maximal cliques per second in sparse graphs, and the computation time increases almost linearly with the density of the graph. Finding all k-cliques from the list of maximal cliques is not trivial as there are many overlapping k-cliques in those maximal cliques. To solve this issue we initialize a hash table H (line 1 of Algorithm 1) where the key is generated by the id's of participating vertices in a k-clique, and the value is a boolean to show the existence of that k-clique. So, it will store all the enumerated k-cliques. In line 6, we create a bitmask of length n with k bits where n is the number of vertices in the maximal clique M. Instead of generating all permutations of the actual vertex id's, we generate all permutations of this bit mask to reduce computation time leveraging bitwise operations. In line 8, we initialize an empty array C for every permutation of the bitmask. We add a vertex from the maximal clique M into C if the corresponding bit value is 1. If C is not in the hash table H (enumerated for the first time), then we save C in H with value 1. Thus we prevent duplicates of k-cliques.

2.2 Higher-Order Nucleus Decompositions:

The framework of (r, s)-nucleus decomposition adopted a hypergraph version of classic Matula-Beck [19]. We extend this framework to design a generic (r, s)-nucleus decomposition. But as the number of cliques increases exponentially for large cliques in most of the datasets and maintaining these large lists is memory inefficient and intractable, we implement the algorithm case-by-case up to (9, 10)-nucleus decomposition to improve the runtime performance. We do not implement (r, s)-nucleus decomposition where $s > 10$ for practical purposes. We also ignore the nucleus decompositions where $s - r > 1$ since it has been shown that $s - r = 1$ is the best combination for the options with equal s [25].

We first get the list of all r-cliques and s-cliques by using Algorithm 1. Note that both r-cliques and s-cliques can be enumerated at the same time in Algorithm 1 by simply changing the inner part of the outer-most loop. This way we reduce the clique enumeration time which has a huge impact on the runtime when the number of cliques is very large. Also, we only store the r-cliques in a dynamic array for better scalability in terms of the memory space. The list of s-cliques is used only for faster s-clique counting (s-degree) for each r-clique. To save the s-degree of an r-clique, we create an index by using the id's of participating vertices in the r-clique. We use a hash table to store the index and s-degree of an r-clique in order to enable faster lookups. For utilizing the bucket data structure from the framework of (r, s)-nucleus decomposition we create another hash table to store the index of an r-clique and its position in the dynamic array where every r-clique is stored. So, in the bucket, we store the r-clique's position in the dynamic array and its s-degree. The bucket data structure uses linked lists for storing bucket contents and hash maps for finding the link list entry of any given r-clique. During the peeling process, we use the bucket sort for updating the s-degrees of r-cliques. In every iteration (line 6 of Algorithm 2), we choose the unprocessed r-clique with lowest s-degree and assign this value as its k_s-value. Then we find all s-cliques which contain this r-clique, check whether the neighbor r-cliques in those s-cliques are processed or not. If not processed and s-degrees of those neighbor r-cliques are greater than the s-degree of the current r-clique, then s-degrees of the neighbor r-cliques are decremented.

Algorithm 1: k-CliqueGen(G, k)

```
    input  : $G$: graph, $k$: positive integer
    output: list of all $k$-cliques
 1  Initialize a hash table H
 2  Enumerate all maximal cliques using MACE [18]
 3  foreach maximal clique $M$ do
        // $M$ is an array of vertices
 4      $n \leftarrow$ number of vertices in $M$
 5      if $n \geq k$ then
 6          $b \leftarrow$ bitmask of $k$ leading 1's and $(n-k)$ trailing 0's
 7          for every permutation of $b$ do
 8              $C \leftarrow$ an empty array
                // C stores the clique vertices
 9              for $i = 0$ to $n$ do
10                  if $b[i] = 1$ then add M[i] to $C$
11              if $C$ is not in the hash table $H$ then
12                  save $C$ in the hash table
```

Algorithm 2: set-k(G, r, s)

```
input  : $G$: graph, $r < s$: positive integers
output: $k_s(.)$ : array of $k_s$ indices for $r$-cliques
// fast clique enumeration by Algorithm 1
1  $r$-cliquesList, $s$-cliquesList ← list of $r$-cliques and $s$-cliques by k-CliqueGen($G$, $k$)
   // initialization
2  foreach $r$-clique $R \in r$-cliquesList do
       // using $s$-cliquesList for fast counting
3      $d_s(R) \leftarrow$ number of $s$-cliques containing $R$
4      Mark $R$ as unprocessed
5      Save $R$ in a hash table with $d_s(R)$ value
   // peeling
6  foreach unprocessed $r$-clique $R$ with min. $d_s(R)$ do
7      $k_s(R) = d_s(R)$                    // nucleus number assigned
8      Find set $C$ of $s$-cliques containing $R$
9      foreach $s$-clique $S \in C$ do
10         if any $r$-clique $R' \subset S$ is processed then
11             Continue
12         foreach $r$-clique $R' \in S$ do
13             if $d_s(R') > d_s(R)$ and $R' \neq R$ then
14                 $d_s(R') = d_s(R') - 1$
15     Mark $R$ as processed
```

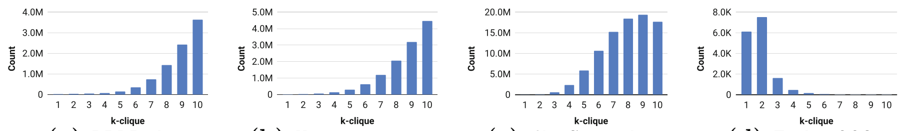
3 Experiments

We evaluate our algorithms on various types of undirected simple real-world networks from different domains such as social networks (Hamsterter, fb-Reed, fb-Simmons, fb-Caltech36, fb-Haverford76, fb-Swarthmore42, fb-USFCA72), collaboration networks (Jazz, Erdos992, DBLP_ds, DBLP_pp, DBLP_dm), interaction networks (PGP, Drug), internet networks (Caida), and infrastructure networks (PowerGrid). All datasets are collected from SNAP [16], Konect [15], ICON [5], Network Repository [22], and Facebook100 dataset [27]. Key statistics of our datasets are summarized in Table 1. All experiments are performed on a Linux operating system (v. Linux 3.10.0-1127) running on a machine with Intel(R) Xeon(R) Gold 6130 CPU processor at 2.10 GHz with 192 GB memory.

We find three types of patterns for k-clique distribution in our datasets. The first pattern is shown in Figs. 3a and 3b where k-clique ($3 \leq k \leq 10$) count increases exponentially. This pattern is the most common, observed in 9 networks. In Fig. 3c, the second pattern is shown where the clique count shows some left-skew. In this pattern, the highest clique count is in the range between 1-clique to 10-clique. We observe this pattern in 5 networks (Caida, fb-Reed, fb-Caltech36, fb-Simmons, and fb-Swarthmore42). The last pattern, shown in Fig. 3d, is not so common as we find it in only 2 networks (Erdos992 and PowerGrid). This pattern shows right-skew, and the clique count starts decreas-

Table 1. k-clique ($k \leq 10$) counts for the real-world graphs used in our experiments. Bold numbers indicate the largest k-clique counts of a dataset.

Dataset	$\|V\|$	$\|E\|$	$\|\triangle\|$	$\|K_4\|$	$\|K_5\|$	$\|K_6\|$	$\|K_7\|$	$\|K_8\|$	$\|K_9\|$	$\|K_{10}\|$
Jazz	198	2.7K	17.9K	78.4K	273.7K	846K	2.4M	6.3M	14.8M	**30.5M**
fb-Caltech36	769	16.7K	119.6K	460K	1.3M	2.7M	4.8M	6.9M	**8.2M**	7.9M
fb-Reed	962	18.8K	97.1K	233.2K	349.5K	**398.9K**	392.6K	349K	274.7K	181.9K
fb-Haverford76	1.4K	59.6K	627.9K	3.1M	9.5M	21.1M	37.4M	56.2M	74.2M	**87.5M**
Drug	1.5K	48.5K	569.5K	4.2M	21.8M	86.5M	268.3M	289.7M	288.7M	**408.6M**
fb-Simmons	1.5K	33K	168.6K	456.9K	962.1K	1.7M	2.6M	3.3M	**3.4M**	2.8M
fb-Swarthmore42	1.7K	61.1K	552.7K	2.3M	5.9M	10.6M	15.2M	18.4M	**19.4M**	17.7M
Hamsterster	2.4K	16.6K	53.3K	132.9K	298.1K	619.1K	1.2M	2M	3.2M	**4.5M**
fb-USFCA72	2.7K	65.3K	371.7K	1.2M	3M	7.6M	19.4M	48.2M	112.0M	**233.6M**
PowerGrid	4.9K	**6.6K**	651	90	15	2	0	0	0	0
Erdos992	6.1K	**7.5K**	1.6K	450	168	55	11	1	0	0
DBLP_ds	8.1K	23K	47.2K	148.4K	610.2K	2.5M	9.8M	33.3M	99.9M	**263.9M**
DBLP_PP	8.4K	22.9K	74.2K	436.1K	2.7M	15M	71.6M	300.8M	1.12B	**3.76B**
PGP	10.7K	24.3K	54.8K	238.6K	1M	3.8M	11.4M	27.9M	56.4M	**95.2M**
DBLP_dm	16.4K	33.9K	39.5K	63.5K	145.1K	342.5K	740.3K	1.4M	2.4M	**3.6M**
Caida	26.5K	53.4K	36.4K	53.9K	82.2K	102.1K	**104.1K**	87.5K	60.3K	33.9K



(a) DBLP_dm (b) Hamsterster (c) fb-Swarth. (d) Erdos992

Fig. 3. k-clique ($1 \leq k \leq 10$) distribution for DBLP_dm, Hamsterster, fb-Swarthmore42 and Erdos992 networks. x-axis is the value of k and y-axis is the count of k-cliques with that k value. Clique count increases exponentially in Fig. 3a and 3b which is the most common pattern observed in 9 networks. In Fig. 3c, we observe left-skewed distribution, which is found in 5 networks. The right-skewed behavior in Fig. 3d is not common, observed only in 2 networks.

ing after 2-cliques. Also, it is worth mentioning that there is no correlation between the clique distribution pattern and the size of a network (i.e., number of vertices and edges).

In the following, we present our analysis of higher-order nucleus decompositions with respect to various aspects. We first analyze the s-degree and k_s-value distributions and explain different behaviors observed in real-world networks. Then we investigate how the densest parts found by each higher-order nucleus decomposition change. Last, we look at the hierarchical relationships among nuclei in higher-order decompositions.

3.1 Analysis of s-degrees and k_s-values

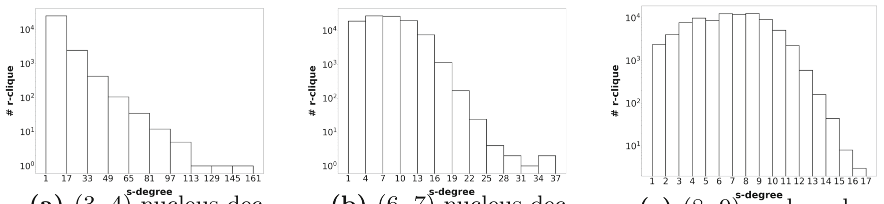
Here we discuss the empirical patterns related to s-degrees and k_s-values that we find in our diverse datasets for higher-order nucleus decompositions. We analyze the large (r, s) nuclei and give comparisons for different r, s values.

s-degree Distributions. One of the most common macroscopic structural properties that are found in real-world networks is the heavy-tailed degree distribution [3,20]. There are very few highly connected nodes coexisting with a large number of lowly connected nodes. According to the clique definition, vertex is 1-clique and edge is 2-clique. So, vertex-edge relationships can be represented by (1, 2)-nucleus, and degree distribution of vertices can be thought of 2-clique frequency distribution of 1-cliques. Similarly, in (r, s)-nucleus decomposition, s-clique frequency of an r-clique is defined as the s-degree, i.e., the number of s-cliques containing the r-clique. We explore s-degree distribution of r-cliques from (3, 4) to (9, 10)-nucleus decompositions.

One interesting finding is that we do not observe any heavy-tailed s-degree distribution in the collaboration networks for higher-order nucleus decompositions (Figs. 4 and 5). In all other networks (except `PGP`), the heavy-tail pattern is observed up to (5, 6)-nucleus decompositions. After that, we observe Poisson distribution for most of the networks. We present `Caida` in Fig. 4 and `DBLP-dm` in Fig. 5 as a representative sample. In both figures, x-axis (binned) and y-axis represent s-degrees and the number of r-cliques, respectively. Other higher-order nucleus decompositions are not shown due to space constraints.

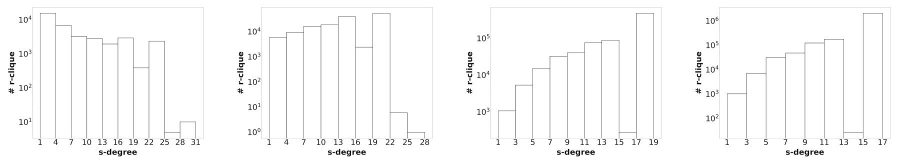
k_s-value Distributions. In (r, s)-nucleus decomposition, k_s-value is an important property of an r-clique encoding local structural information. k_s-values provide a more regular structure and distribution than the s-degrees. We analyze the k_s-values for all higher-order nucleus decompositions. At first, we explore the distribution of k_s-values concerning r-cliques from (3, 4) to (9, 10)-nucleus decomposition. Figure 6 presents the distributions for `Drug` network. Although the k_s-value distribution does not reveal any potential structural information, we notice that for most of the networks the range of k_s-values decreases gradually with larger r, s values, and the mass of the histogram is shifted to the right in the nucleus decompositions with larger r and s values. This indicates that most r-cliques have a large k_s-value in higher-order nucleus decompositions.

We also perform a vertex-centric analysis. The number of r-cliques that a vertex is a part of is an important measure. Figures 7 presents the results for `PGP` network. We discover that there are very few vertices with large r-clique



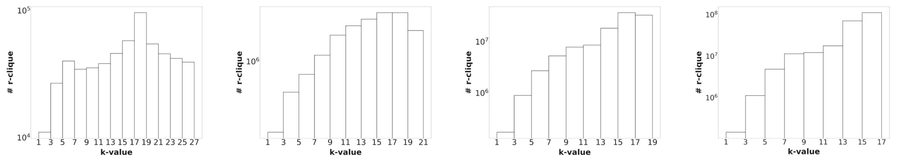
(a) (3, 4) nucleus dec. **(b)** (6, 7) nucleus dec. **(c)** (8, 9) nucleus dec.

Fig. 4. s-degree distribution of r-cliques for `caida` network, for r, s as (3, 4), (6, 7) and (8, 9). x-axis shows the s-degrees and y-axis denotes the counts of r-cliques in log scale. The histogram shows that there is a heavy-tail pattern up to (5, 6)-nucleus decomposition.



(a) (3, 4) nucleus d. **(b)** (5, 6) nucleus d. **(c)** (7, 8) nucleus d. **(d)** (9, 10) nucleus d.

Fig. 5. s-degree distribution of r-cliques for `DBLP_dm` network, for r, s as (3, 4), (5, 6), (7, 8) and (9, 10). x-axis shows the s-degrees and y-axis denotes the counts of r-cliques in log scale. The histogram shows that there is no heavy-tail pattern (similar behaviors observed for other collaboration networks).



(a) (3, 4) nucleus d. **(b)** (5, 6) nucleus d. **(c)** (7, 8) nucleus d. **(d)** (9, 10) nucleus d.

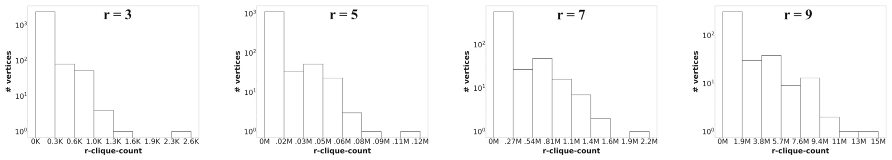
Fig. 6. k_s-value distribution of r-cliques for `Drug` network. The x-axis (binned) is k_s-value and the y-axis is the number of r-cliques. We use logarithmic scale in y-axis to handle the skewness. The mass of the histogram is shifted to right in higher-order nucleus decompositions.

counts in nine networks. These vertices reflect the dense structure in the networks and number of such vertices remains constant in higher-order decompositions. But there is a huge decrease in the total number of vertices which are part of r-cliques in higher-order because of vertices that do not participate in any r-clique. Another interesting pattern that we uncover is the two distinct groups of the vertices in higher-order. One group has very high r-clique counts while the other group has very low r-clique counts. This indicates the existence of core-periphery structure in `DBLP_dm`, `DBLP_ds`, `Drug`, `Erdos992`, `Hamsterster`, and `Jazz` networks [4].

Being part of many r-cliques, a vertex has many k_s-values from those r-cliques. The k_s-value of a vertex in higher-order nucleus decompositions can be defined by the average, max, or min values of those k_s-values. We examine the distribution of those measure but there is no common pattern overall, neither for datasets nor for the varying r, s values.

3.2 Degeneracy Core

In the classic k-core decomposition, every vertex gets a k_s-value (core number) which reflects its significance in the network. The degeneracy of a graph is the maximum k-value such that the graph has a non-empty k-core. The core subgraph with maximum k-value is called the degeneracy core. As degeneracy core



(a) # vertices: 2530 **(b)** # vertices: 1223 **(c)** # vertices: 673 **(d)** # vertices: 403

Fig. 7. Frequency distribution of r-cliques for vertices in `PGP` network. A few vertices have large r-clique counts and it remains constant in higher orders. There is a significant decrease in the number of vertices that are part of an r-clique in higher orders.

is a significant dense subgraph, we analyze it for higher-order nucleus decompositions. In (r, s)-nucleus decomposition, the k_s-value of an r-clique denotes that its s-degree is at least k_s. The r-cliques with the maximum k_s-value form the degeneracy core. We locate those cores by taking the induced subgraph of vertices that are part of those r-cliques. We uncover some interesting structural patterns in the degeneracy cores of higher-order nucleus decompositions. The expected behavior is that the size (vertex count) of the degeneracy core will decrease, and the density (fraction of internal edges with respect to the total possible edges) will increase with higher orders as fine-grained subgraphs will be found in a higher order which are lost in the larger structure. Surprisingly, the degeneracy core with the highest density is found in (3, 4)-nucleus decomposition for eight networks in our dataset. The size and density of the degeneracy core remain unchanged in higher orders for these networks. Table 2 presents the statistics of the densest degeneracy cores for all nucleus decompositions in each network. We only show the nucleus decompositions where going to larger r, s values yields a denser degeneracy core, i.e., the higher-order decompositions that do not results in denser degeneracy cores are not shown. We get the best degeneracy core in (4, 5)-nucleus decomposition for 5 facebook networks. Figure 8 presents the density and size behavior of all the k-cores in `fb-Haverford76`. **In several cases, it is possible to find denser structures with higher-order nucleus decompositions where r, s is larger than 3, 4.** In `fb-Reed` network, for instance, the size of the degeneracy core reduces from 33 to 19 and density increases from 0.6326 to 0.9766 when we go from (3, 4) to (4, 5)-nucleus decomposition. In `PGP` and `fb-USFCA72` networks, the best result is found in (5, 6)-nucleus decompositions. In `Drug` network, it is possible to find denser structures until the (8, 9)-nucleus decomposition.

3.3 Hierarchy Analysis

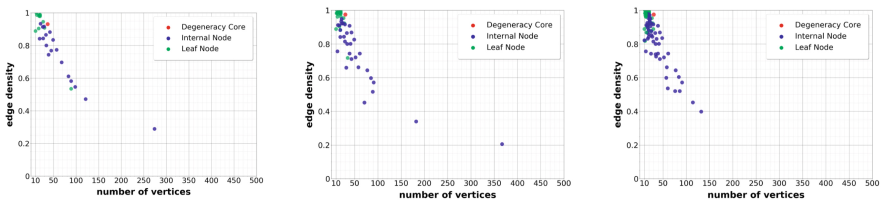
Nucleus decomposition summarizes the network as a forest of nuclei that shows a global hierarchical snapshot of dense subgraphs with containment relations. We analyze the forest of nuclei in higher-order nucleus decompositions to understand the hierarchical structure of a network for large r, s values. We use circle packing diagrams to visualize the forest of nuclei where each circle denotes a

Table 2. Vital statistics of the degeneracy cores. The quality of the degeneracy core is measured both in terms of size and density. For a network, size and density of the degeneracy core in higher-order decompositions are shown until the best degeneracy core is found. The smallest and densest degeneracy core is found in (3, 4)-nucleus decomposition for eight real-world networks.

Dataset	ND	Size	Density
`Caida`	(3, 4)	17	0.9926
`DBLP_ds`	(3, 4)	36	1.0
`DBLP_pp`	(3, 4)	45	1.0
`DBLP_dm`	(3, 4)	25	1.0
`Jazz`	(3, 4)	30	1.0
`Hamster.`	(3, 4)	25	1.0
`Erdos992`	(3, 4)	8	1.0
`fb-USF.`	(3, 4)	35	0.9832
	(4, 5)	35	0.9832
	(5, 6)	32	0.9940

Dataset	ND	Size	Density
`fb-Simmons`	(3, 4)	25	0.9633
	(4, 5)	24	0.9710
`fb-Haverf.`	(3, 4)	37	0.9309
	(4, 5)	30	0.9747
`fb-Caltech36`	(3, 4)	26	0.9661
	(4, 5)	23	0.9841
`fb-Reed`	(3, 4)	33	0.6326
	(4, 5)	19	0.9766
`fb-Swarth.`	(3, 4)	78	0.3876
	(4, 5)	51	0.5152

Dataset	ND	Size	Density
`PGP`	(3, 4)	35	0.9546
	(4, 5)	62	0.5124
	(5, 6)	29	0.9877
`Drug`	(3, 4)	59	0.8843
	(4, 5)	78	0.6417
	(5, 6)	42	0.9268
	(6, 7)	42	0.9268
	(7, 8)	42	0.9268
	(8, 9)	38	0.9445
`PowerG.`	(3, 4)	12	0.5455



(a) (3, 4), # nuclei: 46 **(b)** (4, 5), # nuclei: 82 **(c)** (5, 6), # nuclei: 137

Fig. 8. Size vs density plot of the nuclei (having more than 10 vertices) for `fb-Haverford76` network, for (3, 4), (4, 5) and (5, 6)-nucleus decompositions. In addition to the degeneracy core, we also show the leaf subgraphs and internal non-leaf ones in the nuclei hierarchy. The size of the degeneracy core reduces from 37 to 30 and its density increases from 0.9309 to 0.9747 for moving from (3, 4) to (4, 5)-nucleus decomposition. The degeneracy core remains unchanged for larger higher-order nucleus decompositions.

nucleus subgraph in the forest and the containment relationships are captured with nested circles. The size of the circle is proportional to the number of vertices in the corresponding nucleus, and the color of the circle represents the density (blue and red show density 0 and 1, respectively). Containment within each circle represents a level in the hierarchy. Any nucleus with less than 10 vertices are ignored. For most networks, we observe that denser structures can be obtained in higher-order decompositions with larger r, s values. However, this comes at the cost of losing hierarchical relationships. Figure 9 presents the visualizations for `fb-Simmons` network for all higher-order decompositions. Forest of (r, s)-nuclei can present the hierarchical structure at a finer granularity up to a certain higher-order decomposition. For most of the networks in our experiments, the hierarchical structure disappears after $(7, 8)$-nucleus decomposition.

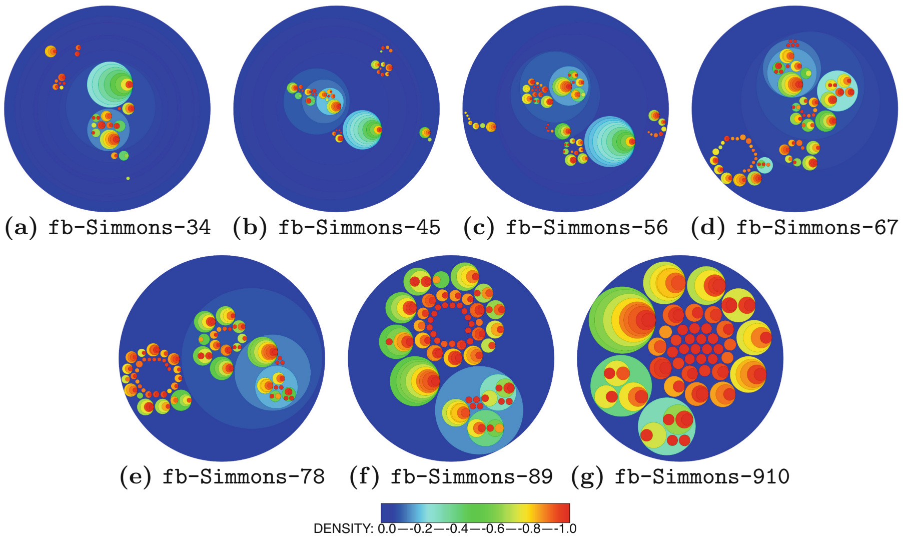


Fig. 9. (r, s)-nuclei forest in higher-order nucleus decompositions for `fb-Simmons`. The density is color coded. Each circle denotes a nucleus subgraph in the forest and the containment relationships are captured with nested circles. Any nucleus with less than 10 vertices are ignored. The leaves are mostly red with high densities. Finer dense structures that are lost in larger structures become visible in higher-order decompositions.

4 Conclusion

In this paper, we introduced an analysis for using large cliques in dense subgraph discovery. Relying on the nucleus decomposition, we implemented higher-order decompositions that can use up to 10-cliques and analyzed the resulting dense subgraphs and hierarchical relations for various real-world networks. Our analysis suggests that utilizing larger cliques can yield denser structures with more interesting hierarchical relations. For future work, we plan to investigate faster algorithms for large (r, s) nucleus decompositions.

Acknowledgments. Sariyuce was supported by NSF-1910063. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

References

1. Alvarez-Hamelin, J.I., Barrat, A., Vespignani, A.: Large scale networks fingerprinting and visualization using the k-core decomposition. In: NIPS, pp. 41–50 (2006)
2. Angel, A., Koudas, N., Sarkas, N., Srivastava, D., Svendsen, M., Tirthapura, S.: Dense subgraph maintenance under streaming edge weight updates for real-time story identification. VLDB J. 1–25 (2013). https://doi.org/10.1007/s00778-013-0340-z

3. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
4. Borgatti, S.P., Everett, M.G.: Models of core/periphery structures. Soc. Netw. **21**(4), 375–395 (2000)
5. Clauset, A., Tucker, E., Sainz, M.: The colorado index of complex networks (2016). https://icon.colorado.edu
6. Cohen, J.: Trusses: cohesive subgraphs for social network analysis. Technical report, National Security Agency Technical Report, Fort Meade, MD (2008)
7. Dourisboure, Y., Geraci, F., Pellegrini, M.: Extraction and classification of dense communities in the web. In: WWW, pp. 461–470 (2007)
8. Du, X., Jin, R., Ding, L., Lee, V.E., Jr., J.H.T.: Migration motif: a spatial-temporal pattern mining approach for financial markets. In: SIGKDD, pp. 1135–1144 (2009)
9. Erdős, P., Hajnal, A.: On chromatic number of graphs and set-systems. Acta Math. Hung. **17**(1–2), 61–99 (1966)
10. Fratkin, E., Naughton, B., Brutlag, D., Batzoglou, S.: Motifcut: regulatory motifs finding with maximum density subgraphs. Bioinformatics **22**(14), e150–e157 (2006)
11. Gibson, D., Kumar, R., Tomkins, A.: Discovering large dense subgraphs in massive graphs. In: VLDB, pp. 721–732 (2005)
12. Gleich, D.F., Seshadhri, C.: Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In: SIGKDD, pp. 597–605 (2012)
13. Jain, S., Seshadhri, C.: A fast and provable method for estimating clique counts using turán's theorem. In: WWW, pp. 441–449 (2017)
14. Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: Trawling the web for emerging cyber-communities. In: WWW, pp. 1481–1493 (1999)
15. Kunegis, J.: Konect: the Koblenz network collection. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1343–1350. ACM (2013)
16. Leskovec, J., Krevl, A.: SNAP Datasets: stanford large network dataset collection (2014). http://snap.stanford.edu/data
17. Lick, D.R., White, A.T.: k-degenerate graphs. Can. J.Math. **22**(5), 1082–1096 (1970)
18. Makino, K., Uno, T.: New algorithms for enumerating all maximal cliques. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 260–272. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27810-8_23
19. Matula, D.W., Beck, L.L.: Smallest-last ordering and clustering and graph coloring algorithms. J. ACM **30**(3), 417–427 (1983)
20. Newman, M.E.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
21. Pinar, A., Seshadhri, C., Vishal, V.: Escape: efficiently counting all 5-vertex subgraphs. In: WWW, pp. 1431–1440 (2017)
22. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI, pp. 4292–4293 (2015)
23. Saito, K., Yamada, T.: Extracting communities from complex networks by the k-dense method. In: IEEE ICDM Workshops, pp. 300–304 (2006)
24. Sarıyüce, A.E., Seshadhri, C., Pınar, A., Çatalyürek, Ü.V.: Finding the hierarchy of dense subgraphs using nucleus decompositions. In: WWW, pp. 927–937 (2015)
25. Sarıyüce, A.E., Seshadhri, C., Pınar, A., Çatalyürek, Ü.: Nucleus decompositions for identifying hierarchy of dense subgraphs. ACM Trans. Web **11**(3), 1–27 (2017)
26. Seidman, S.B.: Network structure and minimum degree. Social Netw. **5**(3), 269–287 (1983)
27. Traud, A.L., Mucha, P.J., Porter, M.A.: Social structure of facebook networks. Phys. A **391**(16), 4165–4180 (2012)

28. Tsourakakis, C.: The k-clique densest subgraph problem. In: International Conference on World Wide Web, WWW, pp. 1122–1132 (2015)
29. Verma, A., Butenko, S.: Network clustering via clique relaxations: A community based. Graph Partitioning and Graph Clustering **588**, 129 (2013)
30. Zhang, Y., Parthasarathy, S.: Extracting analyzing and visualizing triangle k-core motifs within networks. In: IEEE ICDE, pp. 1049–1060 (2012)
31. Zhao, F., Tung, A.: Large scale cohesive subgraphs discovery for social network visual analysis. In: PVLDB, pp. 85–96 (2013)