Nucleus Decompositions for Identifying Hierarchy of Dense Subgraphs

AHMET ERDEM SARIYÜCE, Sandia National Laboratories
C. SESHADHRI, University of California Santa Cruz
ALI PINAR, Sandia National Laboratories
ÜMIT V. ÇATALYÜREK, Georgia Institute of Technology

Finding dense substructures in a graph is a fundamental graph mining operation, with applications in bioinformatics, social networks, and visualization to name a few. Yet most standard formulations of this problem (like clique, quasi-clique, densest at-least-k subgraph) are NP-hard. Furthermore, the goal is rarely to find the "true optimum" but to identify many (if not all) dense substructures, understand their distribution in the graph, and ideally determine relationships among them. Current dense subgraph finding algorithms usually optimize some objective and only find a few such subgraphs without providing any structural relations.

We define the *nucleus decomposition* of a graph, which represents the graph as a *forest of nuclei*. Each nucleus is a subgraph where smaller cliques are present in many larger cliques. The forest of nuclei is a hierarchy by containment, where the edge density increases as we proceed towards leaf nuclei. Sibling nuclei can have limited intersections, which enables discovering overlapping dense subgraphs. With the right parameters, the nucleus decomposition generalizes the classic notions of k-core and k-truss decompositions.

We present practical algorithms for nucleus decompositions and empirically evaluate their behavior in a variety of real graphs. The tree of nuclei consistently gives a global, hierarchical snapshot of dense substructures and outputs dense subgraphs of comparable quality with the state-of-the-art solutions that are dense and have non-trivial sizes. Our algorithms can process real-world graphs with tens of millions of edges in less than an hour. We demonstrate how proposed algorithms can be utilized on a citation network. Our analysis showed that dense units identified by our algorithms correspond to coherent articles on a specific area. Our experiments also show that we can identify dense structures that are lost within larger structures by other methods and find further finer grain structure within dense groups.

CCS Concepts: • **Mathematics of computing → Graph algorithms;**

Additional Key Words and Phrases: k-core, k-truss, graph decomposition, density hierarchy, overlapping dense subgraphs, dense subgraph discovery

This work was funded by the DARPA GRAPHS program, DOE Applied Mathematics Research Program, and Laboratory Directed Research and Development (LDRD) Program of Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. An earlier version of this work was presented in Sariyüce et al. [2015].

Authors' addresses: A. E. Sariyüce and A. Pinar, Sandia National Laboratories, 7011 East Ave. Livermore, CA 94551; emails: a.erdemsariyuce@gmail.com, apinar@sandia.gov; C. Seshadhri, University of California Santa Cruz, 1156 High Street Baskin School of Engineering, Santa Cruz, CA 95064; email: sesh@ucsc.edu; Ü. V. Çatalyuürek, Georgia Institute of Technology, Klaus Advanced Computing Building 266 Ferst Drive, Atlanta GA 30332; email: umit@gatech.edu.

DOI: http://dx.doi.org/10.1145/3057742

ACM Reference Format:
Ahmet Erdem Sariyüce, C. Seshadhri, Ali Pinar, and Ümit V. Çatalyürek. 2017. Nucleus decompositions for identifying hierarchy of dense subgraphs. ACM Trans. Web 11, 3, Article 16 (July 2017), 27 pages.
DOI: http://dx.doi.org/10.1145/3057742

1. INTRODUCTION

Graphs are widely used to model relationships in a wide variety of domains such as sociology, bioinformatics, infrastructure, and the World Wide Web (WWW), to name a few. One of the key observations is that while real-world graphs are often globally sparse, they are locally dense. In other words, the average degree is often quite small (say, at most 10 in a million vertex graph), but vertex neighborhoods are often dense. The classic notions of transitivity [Wasserman and Faust 1994] and clustering coefficients [Watts and Strogatz 1998] measure these densities and are high for many real-world graphs [Sala et al. 2010; Seshadhri et al. 2014].

Finding dense subgraphs is a critical aspect of graph mining [Lee et al. 2010]. It has been used for finding communities and spam link farms in web graphs [Kumar et al. 1999; Gibson et al. 2005; Dourisboure et al. 2007], graph visualization [Alvarez-Hamelin et al. 2006], real-time story identification [Angel et al. 2012], DNA motif detection in biological networks [Fratkin et al. 2006], finding correlated genes [Zhang and Horvath 2005], epilepsy prediction [Iasemidis et al. 2003], finding price value motifs in financial data [Du et al. 2009], graph compression [Buehrer and Chellapilla 2008], distance query indexing [Jin et al. 2009], and increasing the throughput of social networking site servers [Gionis et al. 2013]. This is closely related to the classic sociological notion of group cohesion [Beal et al. 2003; Forsyth 2010]. There are tangential connections to classic community detection, but the objectives significantly differ [Leskovec et al. 2008]. Community definitions involve some relation of inner versus outer connections, while dense subgraphs purely focus on internal cohesion.

1.1. Challenges in Finding Dense Subgraphs

Our input is a graph $G = (V, E)$. For vertex set S, we use $E(S)$ to denote the set of edges internal to S. The *edge density* of S is $\rho(S) = |E(S)|/\binom{|S|}{2}$, the fraction of edges in S with respect to the total possible. The aim is to find a set S with high density subject to some size constraint. Typically, we are looking for large sets of high density.

In general, one can define numerous formulations that capture the main problem. The maximum clique problem is finding the largest S, where $\rho(S) = 1$. Finding the S with at least k vertices that have the highest average degree is known as the densest at-least-k subgraph problem [Khuller and Saha 2009]. Note that the average degree is regarded as the density metric in that work. Quasi-cliques, as defined recently by Tsourakakis et al. [2013], are sets that are almost cliques, up to some fixed “defect.” Unfortunately, most formulations of finding dense subgraphs are NP-hard, even to approximate [Håstad 1996; Feige 2002; Khot 2006].

For graph analysis, one rarely looks for just a single (or the optimal, for whatever notion) dense subgraph. We want to find many dense subgraphs and understand the relationships among them. Ideally, we would like to see if they nest within each other, if the dense subgraphs are concentrated in some region, and if they occur at various scales of size and density. Our article is motivated by the following questions.

—How do we attain a global, hierarchical representation of many dense subgraphs in a real-world graph?
—Can we define an efficiently solvable objective that directly provides *many* dense subgraphs? We wish to avoid heuristics, as they can be difficult to predict formally.

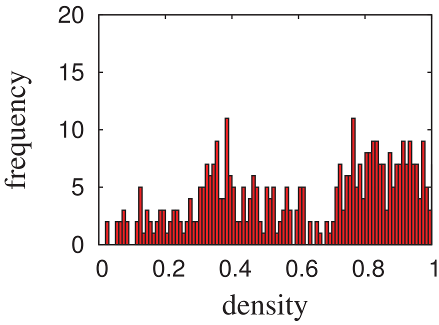


Fig. 1. Density histogram of `facebook` (3, 4)-nuclei; 145 nuclei have a density of at least 0.8 and 359 nuclei have a density of more than 0.25.

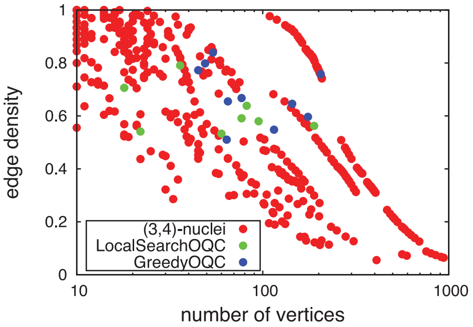


Fig. 2. Size vs. density plot for `facebook` (3, 4)-nuclei; 50 nuclei are larger than 30 vertices with a density of at least 0.8. There are also 138 nuclei larger than 100 vertices with a density of at least 0.25 (best seen in color).

1.2. Our Contributions

Nucleus decompositions: Our primary theoretical contribution is the notion of *nuclei* in a graph. r-clique is a subgraph of r vertices that are all connected to each other. Roughly speaking, an (r, s)-nucleus, for fixed (small) positive integers $r < s$, is a maximal subgraph where every r-clique in it is part of many s-cliques. (The real definition is more technical and involves some connectivity properties.) Moreover, nuclei that do not contain one another cannot share an r-clique. This is inspired by and is a generalization of the classic notion of k-cores and also k-trusses (or triangle cores).

We show that the (r, s)-nuclei (for any $r < s$) form a hierarchical decomposition of a graph. The nuclei are progressively denser as we go towards the leaves in the decomposition. We provide an exact, polynomial algorithm that finds all the nuclei and builds the hierarchical decomposition. In practice, we observe that (3, 4)-nuclei provide the most interesting decomposition. We find the (3, 4)-nuclei for a large variety of more than 20 graphs. Our algorithm is feasible in practice, and we are able to process a 39 million edge graph in less than an hour (using commodity hardware). The source code of our algorithms are available.[1]

Dense subgraphs from (3, 4)-nuclei: The (3, 4)-nuclei provide a large set of dense subgraphs for range of densities and sizes. For example, there are 403 (3, 4)-nuclei (of size at least 10 vertices) in a `facebook` network of 88K edges. We show the density histogram of these nuclei in Figure 1, plotting the number of nuclei with a given density. Observe that we get numerous dense subgraphs, and many with a density fairly close to 1. In Figure 2, we present a scatter plot of vertex size vs. density of the (3, 4)-nuclei. Observe that we obtain dense subgraphs over a wide range of sizes. For comparison, we also plot the output of recent dense subgraph algorithms from Tsourakakis et al. [2013]. (These are arguably the state of the art. More details in next section.) Observe that (3, 4)-nuclei give dense subgraphs of comparable quality. In some cases, the output of Tsourakakis et al. [2013] is very close to a (3, 4)-nucleus.

Representing a graph as a forest of (3, 4)-nuclei: We build the forest of (3, 4)-nuclei for all graphs experimented on. An example output is that of Figure 3, the forest of (3, 4)-nuclei for the `facebook` network. Each node of the forest is a (3, 4)-nucleus, and tree edges indicate containment. More generally, an ancestor nucleus contains all descendant nuclei. By the properties of (3, 4)-nuclei, any two vertices not connected with an edge do not share a triangle. So the branching in the forest represents different

[1] http://bmi.osu.edu/hpc/software/nucleus.

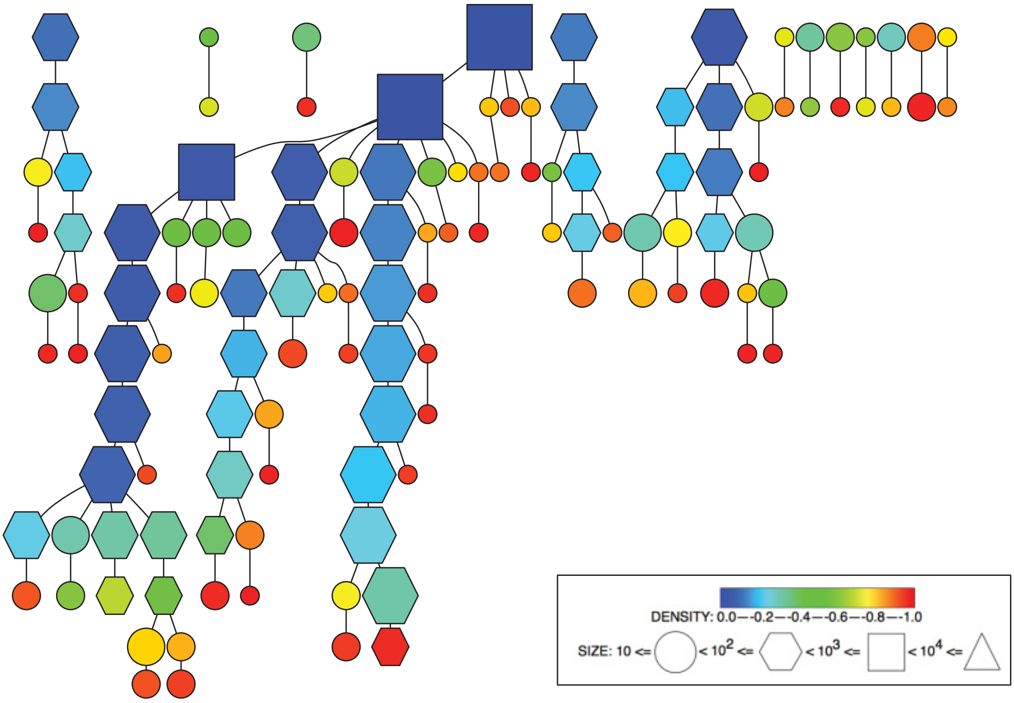

DENSITY: 0.0—-0.2—-0.4—-0.6—-0.8—-1.0
SIZE: 10 <= ○ < 10^2 <= ⬡ < 10^3 <= □ < 10^4 <= △

Fig. 3. (3, 4)-nuclei forest for facebook. Legends for densities and sizes are shown at the top. Long chain paths are contracted to single edges. In the uncontracted forest, there are 47 leaves and 403 nuclei. A parent subgraph contains all the children subgraphs it has. Thus, different branches in the forest depict the different regions in the graph. Thirteen connected components exist in the top level. Sibling nuclei have limited overlaps up to 7 vertices (best seen in color).

regions of the graph. (All nuclei of less than 10 vertices are omitted. For presentation, we contract long chain paths in the tree to single edges, so the forest has less than 403 nodes.)

In the nuclei figures, densities are color-coded, with hotter colors indicating higher density. The log of sizes are coded by shape (circles comprise between 10 and 100 vertices, hexagons between 100 and 1,000 vertices, etc.). For a fixed shape, relative size corresponds to relative size in number of vertices. We immediately see the hierarchy of dense structures. Observe the colors becoming hotter as we go towards to leaves, which are mostly red (density > 0.8). We see numerous hexagons and large circles of color between light blue to green. These indicate the larger parent subgraphs of moderate density (actually density of say 0.25 is fairly high for a subgraph having many hundreds of vertices).

The branching is also significant, and we can group together the dense subgraphs according to the hierarchy. We observe such branching in all our experiments and show more such results later in the article. The (3, 4)-nuclei provide a simple, hierarchical visualization of dense substructures. They are well defined and their exact computation is algorithmically feasible and practical.

We also want to emphasize the overlap between sibling nuclei. While sibling nuclei cannot share triangles, they can share edges, thus vertices. We observe roughly 20 pairs of (3, 4)-nuclei having intersections of four to six vertices. For larger graphs, we observe many more pairs of intersecting nuclei (with larger intersections).

Experimental Analysis: Our experiments had two thrusts. First, we verified that our algorithms could identify many dense structures in real graphs. Our comparisons with state of the art to find the densest subgraph showed that our methods were also competitive and can find denser substructures in many cases. We also showed that higher-order nucleus decompositions could identify denser structures and identify a hierarchical density structure in the graph. Finally, we showed that our algorithms were fast enough to process very large graphs in reasonable times. For instance, our algorithms processed graphs with tens of millions of edges in under an hour.

In the second thrust of our experimental work, we wanted to validate our methods by confirming that structures identified by our algorithms were meaningful. For this purpose, we applied our techniques to a citation network for articles published in American Physical Society (APS) journals. Our results showed that dense units identified by our algorithms corresponded to coherent articles often marked by the same keyword. Our experiments also showed that higher-order decompositions could provide a finer grain view of the dense structures. For instance, (3,4) decomposition revealed a dense group of articles that was lost within a much bigger group when (2,3) decomposition was used. In another example, we observed that a dense structure found by (2,3) decomposition was split into two siblings by (3,4) decomposition (which turned out to be subareas in network science) to provide a more fine-grained view of the density structure.

The rest of the article is organized as follows: Section 2 summarizes the related work, Section 3 introduces the main definitions and the lemma about the nucleus decomposition, Section 4 gives the algorithm to generate a nucleus decomposition and provides a complexity analysis, Section 5 contains the results of extensive experiments we have, and Section 6 concludes the article by discussing the future directions.

2. PREVIOUS WORK

We focus on the dense subgraph discovery problem and aim to find many dense structures and build relations among them. Here we give a summary of the recent literature on dense subgraph discovery methods and explain the *k*-core and *k*-truss decompositions on which we build our algorithms.

Dense subgraph algorithms: As discussed earlier, most formulations of the densest subgraph problem are NP-hard. Some variants, such as maximum average degree [Goldberg 1984; Gallo et al. 1989] and the recently defined triangle-densest subgraph [Tsourakakis 2015], are polynomial time solvable. Linear time approximation algorithms have been provided by Asahiro et al. [2000], Charikar [2000], and Tsourakakis [2015]. There are also dynamic dense subgraph algorithms to handle streaming graph data. Sariyüce et al. [2013] introduced incremental algorithms to maintain *k*-core decomposition. Recently, Epasto et al. [2015] and Bhattacharya et al. [2015] introduced approximation algorithms to maintain densest subgraph, where they use average degree as the density metric. Balalau et al. [2015] adapted the densest subgraph problem for multiple subgraphs with highest total density and introduced heuristics. There are numerous recent practical algorithms for various such objectives: Andersen and Chellapilla's use of cores for dense subgraphs [Andersen and Chellapilla 2009], Rossi et al.'s heuristic for clique [Rossi et al. 2013], and Tsourakakis et al.'s notion of quasi-cliques [Tsourakakis et al. 2013]. These algorithms are extremely efficient and produce excellent output. For comparison's sake, we consider Tsourakakis et al. [2013] as the state of the art, which was compared with previous core-based heuristics and is far superior to prior works. Indeed, their algorithms are elegant, extremely efficient, and provide high-quality output (and are much faster than ours; see Section 5.5 for more discussion). These methods are tailored to finding one (or a few) dense subgraphs and do not give a global/hierarchical view of the structure of dense subgraphs. We believe it would be worthwhile to relate their methods with our notion of nuclei to design even better algorithms.

***k*-cores and *k*-trusses:** The concepts of *k*-cores and *k*-trusses form the inspiration for our work. A *k*-core is a maximal subgraph where each vertex has minimum degree

k, while a k-truss is a subgraph where each edge participates in at least k triangles. The first definition of k-cores was given by Erdős and Hajnal [1966]. It has been rediscovered numerous times in the context of graph orientations and is alternately called the coloring number and degeneracy [Lick and White 1970; Seidman 1983]. The first linear time algorithm for computing k-cores was given by Matula and Beck [1983]. The earliest applications of cores to social networks was given by Seidman [1983], and it is now a standard tool in the analysis of massive networks. The notions of k-truss is first introduced by Saito and Yamada [2006] with a different name: k-dense subgraphs. Later, it was independently proposed by Cohen [2008], Zhang and Parthasarathy [2012], Zhao and Tung [2013], and de Simon et al. [2013] for finding clusters and for network visualization. They all provide efficient algorithms for these decompositions, and Cohen [2008] and Wang and Cheng [2012] explicitly focus on massive scale. Wang et al. [2010] proposed dense-neighborhood (DN) graph, a similar concept to k-truss, where each edge should be involved in k triangles, and adding or removing a vertex from DN-graph breaks this constraint. Putting additional connectedness constraints are investigated in Huang et al. [2014], where they define "k-truss community" as the subgraph where each edge resides in at least k triangles, and each edge pair is triangle-connected (shares a common triangle transitively). Our $(2, 3)$-nucleus definition is the same. Regarding the investigation of hierarchy extracted by k-core decomposition, Adcock et al. [2013] provided a comparison of the tree-decomposition and k-core hierarchies and concluded that they show similarities. Apart from the k-core and k-truss definitions, k-plex and k-club subgraph definitions have drawn a lot of interest as well. In a k-plex subgraph, each vertex is connected to all but at most $k - 1$ other vertices [Seidman and Foster 1978], which complements the k-core definition. In a k-club subgraph, the shortest path from any vertex to other vertex is not more than k [Mokken 1979]. Regarding the generalization of k-core idea, Tatti and Gionis [2015] very recently introduced an adaptation of k-core decomposition in which average degree is increasing with higher k values. Another interesting theoretical work, that we recently discovered, is introduced by Francisco and Oliveira [2011], where they provide a more general definition of k-core in terms of monotone vertex properties and subgraph involvements. All these methods find subgraphs of moderate density and give a global decomposition to visualize a graph.

3. NUCLEUS DECOMPOSITION

Our main theoretical contribution is the notion of nucleus decompositions. We are given an undirected, simple graph G.

Definition 3.1. Let $r < s$ be positive integers and $\mathcal{S}$ be a set of s-cliques in G.

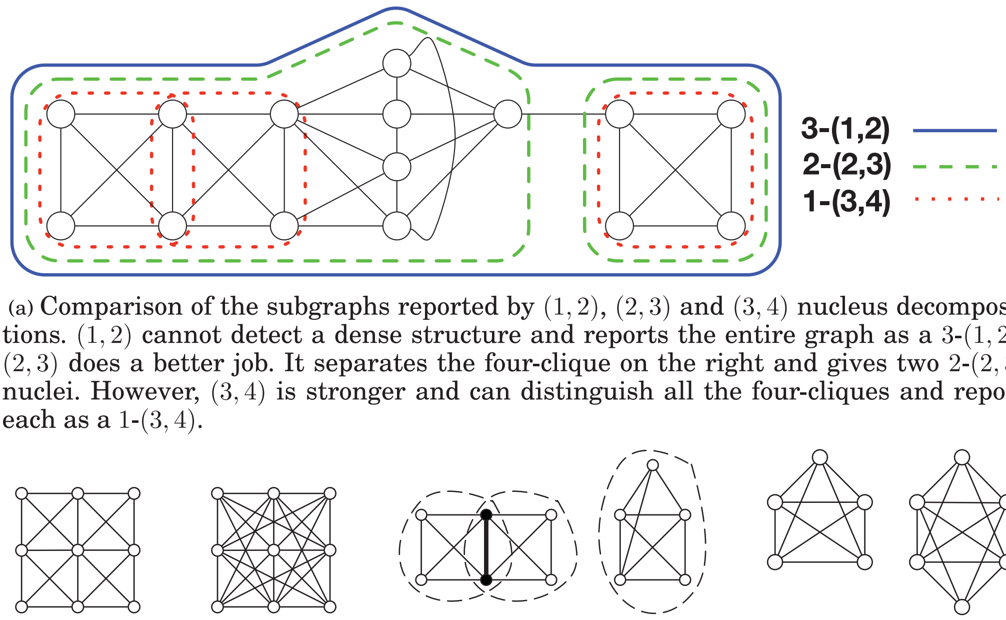
—$K_r(\mathcal{S})$ the set of r-cliques contained in some $S \in \mathcal{S}$.
—The number of $S \in \mathcal{S}$ containing the r-clique $R \in K_r(\mathcal{S})$ is the *$\mathcal{S}$-degree* of that r-clique.
—Two r-cliques R, R' are *$\mathcal{S}$-connected* if there exists a sequence $R = R_1, R_2, \ldots, R_k = R'$ in $K_r(\mathcal{S})$ such that for each i, some $S \in \mathcal{S}$ contains $R_i \cup R_{i+1}$.

These definitions are generalizations of the standard notion of the vertex degree and connectedness. Indeed, setting $r = 1$ and $s = 2$ (so $\mathcal{S}$ is a set of edges) yields exactly that. Our main definition is as follows.

Definition 3.2. Let k, r, and s be positive integers such that $r < s$. A *k-(r, s)-nucleus* is a subgraph that contains the edges in the maximal union $\mathcal{S}$ of s-cliques such that:

—The $\mathcal{S}$-degree of any $R \in K_r(\mathcal{S})$ is at least k.
—Any $R, R' \in K_r(\mathcal{S})$ are $\mathcal{S}$-connected.

We simply refer to (r, s)-nuclei when k is irrelevant in the context. A union of cliques is defined to be the subgraph where the vertex set is the union of vertices of cliques,



(a) Comparison of the subgraphs reported by $(1,2)$, $(2,3)$ and $(3,4)$ nucleus decompositions. $(1,2)$ cannot detect a dense structure and reports the entire graph as a 3-$(1,2)$. $(2,3)$ does a better job. It separates the four-clique on the right and gives two 2-$(2,3)$ nuclei. However, $(3,4)$ is stronger and can distinguish all the four-cliques and report each as a 1-$(3,4)$.

(b) Two graphs have the same number of vertices. 2-$(2,3)$ nucleus, on the left, has 20 edges whereas the 2-$(2,4)$ shown on the right has 28 edges, thus denser.

(c) The left figure shows two $(3,4)$-nuclei overlapping at an edge, since there is no triangle that resides in both $(3,4)$-nuclei. The right figure has only one $(3,4)$-nucleus.

(d) 5-clique on the left is a 2-$(3,4)$. If we add another vertex to the bottom that is connected to the 4 vertices of 5-clique, as shown on the right, it is still a 2-$(3,4)$.

Fig. 4. Illustrative examples.

and the edge set is the union of edges of cliques. Note that we treat nuclei as a union of cliques, although, eventually, we look at this as a subgraph. Our theoretical treatment is more convenient in the former setting, and hence we stick with this definition. In our applications, we simply look at nuclei as subgraphs.

As stated earlier, our definitions are inspired by k-cores and k-trusses. Set $r = 1$, $s = 2$. A k-$(1,2)$-nucleus is a maximal (induced) connected subgraph with minimum vertex degree k. *This is exactly a k-core.* Setting $r = 2$, $s = 3$ gives maximal subgraphs, where every edge participates in at least k triangles, and edges are triangle connected. This is very similar to the definition of k-trusses or triangle-cores, which have more relaxed connectivity constraints.

In Figure 4(a), we compare the $(1,2)$- (k-core), $(2,3)$-, and $(3,4)$-nuclei in the given graph. The smallest subgraph that k-core decomposition can report is the entire graph, which is a 3-core. k-core cannot detect the bridge or any of the 4-cliques. $(2,3)$ decomposition detects the 4-clique on the right as a 2-$(2,3)$-nucleus and separates from the rest of the graph. It basically leverages the fact that a bridge has no triangles. However, it cannot give a finer structure, and 11 vertices on the left are reported as a single 2-$(2,3)$-nucleus. $(3,4)$ decomposition is able to distinguish all three 4-cliques as 1-$(3,4)$-nuclei. $(3,4)$-nuclei can overlap on the edge level, and that is the reason two merged 4-cliques on the left are separated.

Intuitively, a nucleus is a tightly connected cluster of cliques. For large k, we expect the cliques in $\mathcal{S}$ to intersect heavily, creating a dense subgraph. For a fixed k, r and same number of vertices, the density of the nuclei increases as we increase s. Consider the example of Figure 4(b), where there is a 2-(2, 3)-nucleus and a 2-(2, 4)-nucleus on the same number of vertices. Since in the latter case we need every edge to participate in at least two 4-cliques, the resulting density is much higher.

So far, we have only discussed the degree constraint of nuclei. Note that a nucleus is not just connected in the usual (edge) sense but requires the stronger property of being $\mathcal{S}$-connected. The standard definitions of trusses or triangle-cores omit the triangle-connectedness. For us, this is critical. Two cliques of distinct (r, s)-nuclei *can* intersect. For example, when $r > 2$, nuclei can have edge overlaps. This allows for finding even denser subgraphs, as Figure 4(c) shows. In the left figure, cores, trusses, and so on, pick up the entire graph. But there are actually two different 1-(3, 4)-nuclei (each 4-clique) intersecting at an edge. The (3, 4)-nuclei are denser than the graph itself. Note that any edge disjoint decomposition would not find two dense subgraphs.

Critically, the set of (r, s)-nuclei form a laminar family. A laminar family is a set system where all pairwise intersections are trivial (either empty or contains one of the sets).

LEMMA 3.3. *The family of (r, s)-nuclei form a laminar family in terms of r-cliques.*

PROOF. Consider k-(r, s)-nucleus $\mathcal{S}$ and k'-(r, s)-nucleus $\mathcal{S}'$, where $k \leq k'$. Suppose they had a non-empty intersection that includes an s-clique, so some $K_s(S)$ is contained in both $\mathcal{S}$ and $\mathcal{S}'$. Observe that r-cliques in $K_r(\mathcal{S})$ are connected to the r-cliques in $K_s(S)$, which is also connected to the r-cliques in $K_r(\mathcal{S}')$. Furthermore, the $(\mathcal{S} \cup \mathcal{S}')$-degree of a member of $K_r(\mathcal{S} \cup \mathcal{S}')$ is at least k. Hence, $\mathcal{S} \cup \mathcal{S}'$ satisfies the two conditions of being a nucleus, except maximality. By $\mathcal{S}$ is a k-(r, s)-nucleus, so $\mathcal{S} \cup \mathcal{S}' = \mathcal{S}$. So any non-empty intersection is trivial, which is a contradiction. □

Consider two nuclei that are not ancestor descendant. By the above lemma, these two nuclei (considered as subgraphs of G) cannot share an s-clique. Furthermore, they cannot even share an r-clique, since that would connect $\mathcal{S}$ and $\mathcal{S}'$. This is the key disjointness property of nuclei.

Every laminar family is basically a hierarchical set system. Alternately, every laminar family can be represented by a forest of containment. For every nucleus $\mathcal{S}$, any other nucleus intersecting $\mathcal{S}$ is either contained in $\mathcal{S}$ or contains $\mathcal{S}$. Furthermore, all these sets are nested in each other. It makes sense to talk of the smallest sized nucleus containing $\mathcal{S}$. This leads to the main construct we use to represent nuclei.

Definition 3.4. Fix $r < s$. Define the *forest of (r, s)-nuclei* as follows. There is a node for each (r, s) nucleus. The parent of every nucleus is the smallest (by cardinality) other nucleus containing it.

In our figures, we will only show the internal nodes of out degree at least 2 and contract any path of out degree 1 vertices into a single path. This preserves all the branching of the forest.

4. COMPUTING NUCLEUS DECOMPOSITIONS

Our primary algorithmic goal is to construct the tree of nuclei. Our algorithm adopts the classic Matula-Beck result for getting k-cores in linear time [Matula and Beck 1983], but an extension of this approach to generalized nucleus decompositions is far from trivial, and the most intuitive extension does not work. Specifically, it will be wrong to apply core decomposition on a graph with nodes corresponding to r-cliques and edges decoding two nodes being part of the same s-clique, because each s-clique contains $\binom{s}{r}$ r-cliques. This information is lost in the graph representation leading to inaccurate results. At some level, we are performing a hypergraph version of Matula-Beck. The proofs therefore need to be adapted to this setting.

Analogously to k-cores, the main procedure `set-k` (Algorithm 1) assigns a number, denoted by $\kappa(\cdot)$, to each r-clique in G.

ALGORITHM 1: `set-k`(G, r, s)

```
1  Enumerate all $r$-cliques and $s$-cliques in $G(V, E)$;
2  For every $r$-clique $R$, initialize $\delta(R)$ to be the number of $s$-cliques containing $R$;
3  Mark every $r$-clique as unprocessed;
4  for each unprocessed $r$-clique $R$ with minimum $\delta(R)$ do
5      $\kappa(R) = \delta(R)$;
6      Find set $\mathcal{S}$ of $s$-cliques containing $R$;
7      for each $S \in \mathcal{S}$ do
8          if any $r$-clique $R' \subset S$ is processed then
9              Continue;
10         for each $r$-clique $R' \subset S$, $R' \neq R$ do
11             if $\delta(R') > \delta(R)$ then
12                 $\delta(R') = \delta(R') - 1$ ;
13     Mark $R$ as processed;
14 return array $\kappa(\cdot)$ ;
```

It is convenient to denote the set of r-cliques in G by a sequence $\{(R_i)\}$: $R_1, R_2, \dots,$ where R_i is the ith processed r-clique in `set-k`. We will refer to this index as *time*. When we say "at time t," we mean at the beginning of the iteration where R_t is processed.

Claim 1. *The sequence $\{\kappa(R_i)\}$ is monotonically non-decreasing.*

Proof. This holds because the loop processes R in non-decreasing order of $\delta(R)$ and Step 11 ensures that no new value of $\delta(\cdot)$ decreases below the current $\kappa(R)$. □

—Because of Claim 1, we can define *transition time t_i* to be the first time when the κ-value becomes i. Formally, t_i is the unique index such that $\kappa(R_{t_i}) = i$ and $\kappa(R_{t_i-1}) < i$.

—We say s-clique S is *unprocessed at time t* if all $R \in K_r(S)$ are unprocessed at time t. This set of s-cliques is denoted by $\mathcal{S}_t$.

—The *supergraph $\mathcal{G}_t$* has node set $K_r(\mathcal{S}_t)$, and $R, R' \in K_r(\mathcal{S}_t)$ are connected by a link if $R \cup R'$ is contained in some s-clique of $\mathcal{S}_t$. Links are associated with elements of $\mathcal{S}_t$ (and there may be multiple links between R and R').

We prove an auxiliary claim relating the $\delta(\cdot)$ values to $\mathcal{S}_t$.

Claim 2. *At time t, for any unprocessed r-clique R, $\delta(R)$ is at least the $\mathcal{S}_t$-degree of R. If $t = t_k$ (for some k), then $\delta(R)$ is exactly the $\mathcal{S}_t$-degree of R.*

Proof. Pick unprocessed R'. The value of $\delta(R)$ is initially the number of s-cliques containing R'. It is decremented only in Step 12, which happens only when a processed s-clique containing R' is found. (Sometimes, the decrement will still not happen because of Step 11.) Hence, the value of $\delta(R')$ at time t is at least the number of unprocessed s-cliques containing R'.

Suppose $t = t_k$. For any preceding $\hat{t} < t$, the current $\kappa(\cdot)$ value is always at most k. For unprocessed (at time t) R, $\delta(R) > k$. Hence the decrement of Step 12 will always happen, and $\delta(R)$ is exactly the $\mathcal{S}_t$-degree of R. □

Claim 3. *Every k-(r, s)-nucleus is contained in $\mathcal{S}_{t_k}$.*

Proof. Consider k-(r, s)-nucleus $\mathcal{S}$. Take the first $R \in K_r(\mathcal{S})$ that is processed. At this time (say, t), no $S \in \mathcal{S}$ can be processed. Hence, $\mathcal{S} \subseteq \mathcal{S}_t$. By Claim 2, $\delta(R)$ is at least the $\mathcal{S}_t$-degree of R, which is at least the $\mathcal{S}$-degree of R. The latter is at least k, since $\mathcal{S}$ is a k-(r, s)-nucleus. By definition of t_k, $t \geq t_k$ and hence $\mathcal{S}_t \subseteq \mathcal{S}_{t_k}$. Thus, $\mathcal{S} \subseteq \mathcal{S}_{t_k}$. □

The main lemma shows that the output of `set-k` essentially gives us the nuclei.

Lemma 4.1. *The k-(r, s)-nuclei are exactly the connected components of $\mathcal{G}_{t_k}$.*

Proof. Consider a k-(r, s)-nucleus $\mathcal{S}$. By Claim 3, it is contained in $\mathcal{S}_{t_k}$. By the nucleus definition, $\mathcal{S}$ is connected (as links) in $\mathcal{G}_{t_k}$. Let $\mathcal{S}'$ be the (set of links) connected component of $\mathcal{G}_{t_k}$ containing $\mathcal{S}$. By Claim 2, at time t_k, for any $R \in K_r(\mathcal{S}')$, $\delta(R)$ is exactly the $\mathcal{S}_{t_k}$-degree of R. Since $\mathcal{S}'$ is a connected component of $\mathcal{G}_{t_k}$, the $\mathcal{S}_{t_k}$-degree is the $\mathcal{S}'$-degree, which in turn is at least k. In other words, $\mathcal{S}'$ satisfies both conditions of being a k-(r, s)-nucleus, except maximality. By maximality of $\mathcal{S}$, $\mathcal{S} = \mathcal{S}'$. □

Building the forest of nuclei: From Lemma 4.1, it is fairly straightforward to get all the nuclei. First run `set-k` to get the processing times and the $\kappa(\cdot)$ values. We can then get all t_k times as well. Suppose for any r-clique in G, we can access all the s-cliques containing it. Then, it is routine to traverse $\mathcal{G}_{t_k}$ to get the links of connected components. To avoid traversing the same component repeatedly, we produce nuclei in reverse order of k. In other words, suppose all connected components of $\mathcal{G}_{t_{k+1}}$ have been determined. For $\mathcal{G}_{t_k}$, it suffices to determine the connected components involving nodes processed in time $[t_k, t_{k+1})$. Any time a traversal encounters a node in $\mathcal{G}_{t_{k+1}}$, we need not traverse further. This is because all other connected nodes of $\mathcal{G}_{t_{k+1}}$ are already known from previous traversals, and therefore it suffices to visit all nodes and links of $\mathcal{G}_0$ exactly once. More information can be found in Section 4.2.

4.1. Bounding the Complexity

There are two options of implementing this algorithm. The first is faster but has forbiddingly large space. The latter is slower but uses less space. In practice, we implement the latter algorithm. We use $ct_r(v)$ for the number of r-cliques containing v and $ct_r(G)$ for the total number of r-cliques in G. We denote by $RT_r(G)$ the running time of an arbitrary procedure that enumerates all r-cliques in G.

Theorem 4.2. *It is possible to build the forest of nuclei in $O(RT_r(G) + RT_s(G))$ time with $O(ct_r(G) + ct_s(G))$ space.*

Proof. The very first step of `set-k` requires the clique enumeration. Suppose we store the global supergraph $\mathcal{G} = \mathcal{G}_0$. This has a node for every r-clique in G and a link for every s-clique in G. The storage is $O(ct_r(G) + ct_s(G))$. From this point onwards, all remaining operations are linear in the storage in terms of r- and s-cliques. This is by the analysis of the standard core decomposition algorithm of Matula and Beck [1983]. k-(r, s) nucleus decomposition is the peeling process on r-cliques and their S-degrees and can be thought of as the higher-order variant of Matula and Beck [1983]. Every time we process an r-clique, we can delete it and all incident links from $\mathcal{G}$. Every link is touched at most a constant number of times during the entire running on `set-k`. As explained earlier, we can get all the nuclei by a single traversal of $\mathcal{G}$. □

Theorem 4.3. *It is possible to build the forest of nuclei in $O(RT_r(G) + \sum_v ct_r(v)d(v)^{s-r})$ time with $O(ct_r(G))$ space.*

Proof. Instead of explicitly building $\mathcal{G}$, we only build adjacency lists when required. The storage is now only $O(ct_r(G))$. In other words, given an r-clique R, we find all s-cliques containing R only when R is processed/traversed. Each R is processed or traversed at most once in `set-k` and the forest building. Suppose R has vertices $v_1, v_2, \ldots, v_r$. We can find all s-cliques containing R by looking at all $(s-r)$-tuples in each of the neighborhoods of v_i. (Indeed, it suffices to look at just one such neighborhood.) This takes time at most $\sum_R \sum_{v \in R} d(v)^{s-r} = \sum_v \sum_{R \ni v} d(v)^{s-r} = \sum_v ct_r(v) d(v)^{s-r}$. □

Let us discuss these runtimes in more detail. When $r < s \leq 3$, it clearly benefits to go with Theorem 4.2. Triangle enumeration is a well-studied problem and there exist numerous optimized, parallel solutions for the problem. In general, the classic triangle enumeration of Chiba and Nishizeki takes $O(m^{3/2})$ [Chiba and Nishizeki 1985] and is much better in practice [Cohen 2009; Schank and Wagner 2005; Suri and Vassilvitskii 2011]. This helps to keep the time and space complexities bounded.

For our best results, we build the (3, 4)-nuclei, and the number of 4-cliques is too large to store. We go with Theorem 4.3. The storage is now at most the number of triangles, which is manageable. The running time is basically bounded by $O(\sum_v ct_r(v)d(v))$. The number of triangles incident to v, $ct_3(v)$ is $cc(v)d(v)^2$, where $cc(v)$ is the clustering coefficient of v. We therefore get a running time of $O(\sum_v cc(v)d(v)^3)$. This is significantly superlinear, but clustering coefficients generally decay with degree [Sala et al. 2010; Seshadhri et al. 2014]. Overall, the implementation can be made to scale to tens of millions of edges with little difficulty.

4.2. Implementation Details for (3, 4)-Nucleus Decomposition

While it is possible to design a generic (r, s)-nucleus decomposition algorithm, runtime performance can be drastically improved by case specific implementations. The main reason for this is that in some cases the number of cliques is so large that maintaining an explicit list becomes inefficient, even intractable. Here we present the implementation details for (3, 4)-nucleus decomposition and highlight the heuristics and data structures we used for higher runtime performance. As mentioned in Section 4.1, we use Theorem 4.3 for (3, 4)-nucleus decomposition, which is more space efficient and enables us to work on large graphs with millions of edges. We start with enumerating all the triangles by leveraging the MinBucket heuristic [Berry et al. 2014; Schank and Wagner 2005; Suri and Vassilvitskii 2011] for efficiency. This heuristic is based on enumerating wedges where the middle vertex has the lowest degree. Ties are broken consistently to provide a total ordering. At the beginning, we create a directed graph structure from the input graph, where an edge between vertices u and v is oriented from u to v, if $d(u) < d(v)$ or $d(u) = d(v) \wedge u < v$. This directed graph structure regularizes the skewed degree distribution and enables faster triangle enumeration and 4-clique counting for each triangle. A triangle is represented by the ids of three participating vertices. To facilitate the faster lookups, we make use of hash maps based on three vertex ids to store the triangles.

In the main loop (starting in line 4 of Algorithm 1), we use the bucket sort to update the 4-clique counts of triangles during the peeling process. We use a bucket data structure that is based on linked lists for storing its bucket contents and on a hash map to quickly locate the link list entry of any given triangle. At each iteration, we select the triangle with lowest 4-clique count, assign this count as its κ value, find the 4-cliques it is involved in, and check if the neighbor triangles in those 4-cliques are unprocessed. If so, then we decrease their 4-clique counts. This process continues until all triangles got their κ values.

Once we have the κ values for all triangles, we can construct the forest of nuclei by traversing the entire graph only once. The key idea is to find the subgraphs where all triangles have the same κ value (which is defined as subcore for k-core case in Sariyüce et al. [2013]) and link them to each other based on containment. There is no

Table I. Properties for the Real-World Graphs of Different Types and Sizes

							[Tsourakakis et al. 2013]		(3,4)-nucleus	
	$\|V\|$	$\|E\|$	$\|\triangle\|$	$\|K_4\|$	$\sum_v c_3(v)d(v)$	(sec)	Density	size	Density	size
dolphins	62	159	95	27	2.2K	< 1	0.68	8	0.71	8
polbooks	105	441	560	319	$23.8K$	< 1	0.67	13	0.62	13
adjnoun	112	425	284	58	$17.6K$	< 1	0.60	15	0.22	32
football	115	613	810	732	$26.3K$	< 1	0.89	10	0.89	10
jazz	198	$2.74K$	$18K$	$78K$	$2.3M$	< 1	1.00	30	1.00	30
celegans n.	297	$2.34K$	3241	2010	$418K$	< 1	0.61	21	0.91	10
celegans m.	453	$2.04K$	3284	2967	$565K$	< 1	0.67	17	0.64	18
email	$1.13K$	$5.45K$	5343	3419	$1.2M$	< 1	1.00	12	1.00	12
facebook	$4.03K$	$88.23K$	$1.6M$	$30M$	$712M$	93	0.83	54	0.98	109
protein_inter.	$9.67K$	$37.08K$	$22.3K$	$12.6K$	$35M$	< 1	1.00	11	1.00	11
as-22july06	$22.96K$	$48.43K$	$46.8K$	$114K$	$199M$	< 1	0.58	12	1.00	18
twitter	$81.30K$	$2.68M$	$13M$	$104M$	$1.8B$	396	0.85	83	1.00	26
soc-sign-epinions	$131.82K$	$841.37K$	$4.9M$	$58M$	$1.4B$	242	0.71	79	1.00	112
coAuthorsCiteseer	$227.32K$	$814.13K$	$2.7M$	$18M$	$2.1B$	50.1	1.00	87	1.00	87
citationCiteseer	$268.49K$	$1.15M$	$847K$	$370K$	$297M$	3.4	0.71	10	1.00	13
web-NotreDame	$325.72K$	$1.49M$	$8.9M$	$232M$	$33.9B$	671	1.00	151	1.00	155
amazon0601	$403.39K$	$3.38M$	$3.9M$	$4.4M$	$802M$	23	1.00	11	1.00	11
web-Google	$875.71K$	$5.10M$	$13M$	$40M$	$11.4B$	163	1.00	46	1.00	33
com-youtube	$1.13M$	$2.98M$	$829K$	$1.5M$	$451M$	43	0.49	119	0.92	24
as-skitter	$1.69M$	$11.09M$	$28.7M$	$148M$	$1.6B$	$1,036$	0.53	319	0.94	91
wikipedia-2005	$1.63M$	$19.75M$	$44.7M$	$78.9M$	$741B$	$1,312$	0.53	33	0.82	14
wiki-Talk	$2.39M$	$5.02M$	$9.2M$	$64.9M$	$136B$	605	0.48	321	0.59	95
wikipedia-200609	$2.98M$	$37.26M$	$84M$	$153M$	$2,015B$	$2,830$	0.49	376	0.62	103
wikipedia-200611	$3.14M$	$39.38M$	$88.8M$	$163M$	$2,197B$	$3,039$	1.00	55	1.00	32

Note: Largest graph in the dataset has more than $39M$ edges. Times are in seconds. Density of subgraph S Is $|E(S)|/\binom{|S|}{2}$, where $E(S)$ is the set of edges internal to S. Sizes are in number of vertices.

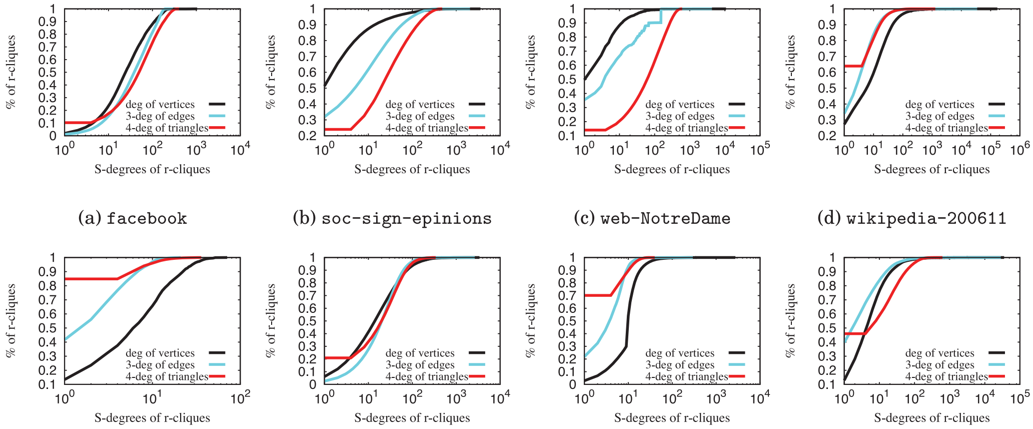
need to traverse further once we encounter a triangle with higher κ, since it is already traversed in previous iterations.

5. EXPERIMENTAL RESULTS

We applied our algorithms to a large variety of graphs, obtained from Stanford Network Analysis Project (SNAP) [2014] and the University of Florida Sparse Matrix Collection [2014]. Important properties of these graphs are given in Table I. We counted the number of triangles that each edge is involved (3-degree) and the number of 4-cliques that each triangle resides in (4-degree) (please see Definition 3.1 for details). Their cumulative distributions as well as the degree distributions of vertices are given in Figure 5 for eight selected graphs (similar behaviors observed for other graphs as well). All variants show a similar skewed distribution, but it is more regular for high S values. All the algorithms in our framework are implemented in C++ and compiled with `gcc 4.8.1` at the -O2 optimization level. All experiments are performed on a Linux operating system running on a machine with two Intel Xeon E5520 2.27GHz CPUs, with 48GB of RAM.

We computed the (r, s)-nuclei for all choices of $r < s \leq 4$, and we present a representative sample here. We mostly observe that the forest of $(3, 4)$-nuclei finds substructures that are denser and presents the hierarchical structure at a finer granularity.

As mentioned earlier, we will now treat the nuclei as just-induced subgraphs of G. A nucleus can be considered as a set of vertices, and we take all edges among these vertices (induced subgraph) to attain the subgraph. The *size* of a nucleus always refers to the number of vertices, unless otherwise specified. For any set S of vertices, the density of the induced subgraph is $|E(S)|/\binom{|S|}{2}$, where $E(S)$ is the set of edges internal to S. *We ignore any nucleus with less than 10 vertices. Such nuclei are not considered in any of our results.*



(a) facebook (b) soc-sign-epinions (c) web-NotreDame (d) wikipedia-200611

(e) email (f) twitter (g) amazon0601 (h) as-skitter

Fig. 5. Cumulative *S-degree* distribution of *r*-cliques for the (1,2), (2,3), and (3,4) cases. The 3-degree of an edge is the number of triangles that edge is involved, and the 4-degree of a triangle is the number of 4-cliques in which that triangle is involved. All variants show a similar skewed distribution, but it is more regular for higher orders (best seen in color).

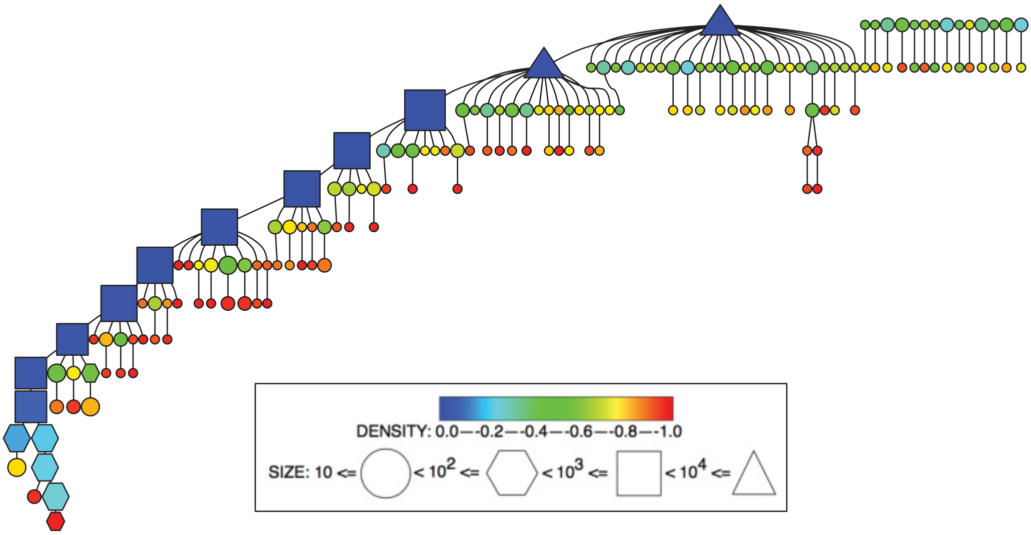


Fig. 6. (3, 4)-nuclei forest for soc-sign-epinions. There are 465 total nodes and 75 leaves in the forest. There is a clear hierarchical structure of dense subgraphs. Leaves are mostly red (>0.8 density). There are also some light blue hexagons, representing subgraphs of size ≥100 vertices with density of at least 0.2 (best seen in color).

5.1. The Forest of Nuclei

We were able to construct the forest of (3, 4)-nuclei for all graphs in Table I but only give the forests for facebook (Figure 3), soc-sign-epinions (Figure 6), and web-NotreDame (Figure 7). For the web-NotreDame figure, we could not present the entire forest, so we show some trees in the forest that had nice branching. The density is color coded, from blue (density 0) to red (density 1). The nuclei sizes, in terms of vertices, are coded by shape: circles correspond to at most 10^2 vertices, hexagons in the range $[10^2, 10^3]$,

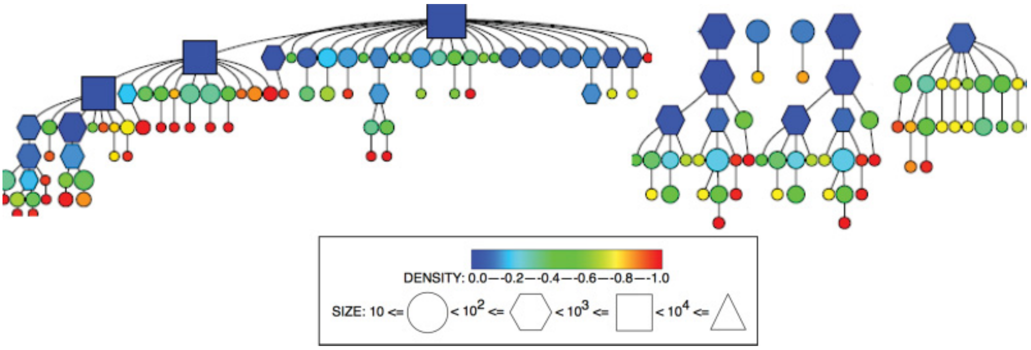

DENSITY: 0.0—-0.2—-0.4—-0.6—-0.8—-1.0
SIZE: 10 <= ○ < 10^2 <= ⬡ < 10^3 <= □ < 10^4 <= △

Fig. 7. Part of the $(3, 4)$-nuclei forest for web-NotreDame. In the entire forest, there are $2,059$ nodes and 812 leaves; 79 of the leaves are clique, up to the size of 155. There is a nice branching structure leading to a more detailed hierarchy (best seen in color).

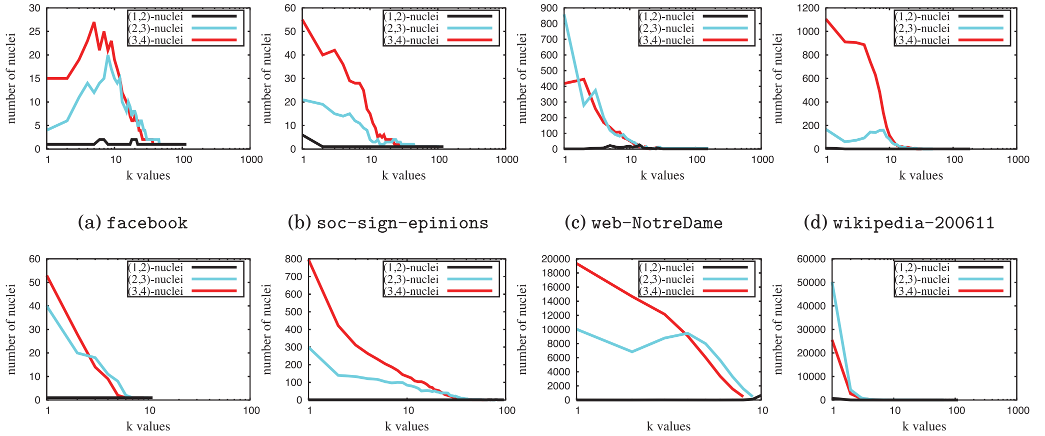
squares in the range $[10^3, 10^4]$, and triangles are anything larger. The relative size of the shape is the relative size (in that range) of the set.

Overall, we see that the $(3, 4)$-nuclei provide a hierarchical representation of the dense subgraphs. The leaves are mostly red, and their densities are almost always >0.8. But we obtain numerous nuclei of intermediate sizes and densities. In the facebook forest and to some extent in the web-NotreDame forest, we see hexagons of light blue to green (nuclei of >100 vertices of densities of at least 0.2). The branching is quite prominent, and the smaller dense nuclei tend to nest into larger, less-dense nuclei. This held in every single $(3, 4)$-nucleus forest we computed. This appears to validate the intuition that real-world networks have a hierarchical structure.

One way to quantify the branching structure in hierarchy is to check the number of nuclei at different levels. Each level corresponds to a set of nuclei with same k value: $k = 1$ at the top, and the value of k increases towards the leaves. Chainlike hierarchies have a small number of nuclei at each level, but we need higher numbers, as they indicate better branching. Figure 8 shows the number of nuclei, for each k value, obtained by $(1, 2)$- (k-core), $(2, 3)$-, and $(3, 4)$-nucleus decompositions on eight representative graphs. $(1, 2)$-nucleus decompositions result in a small number of nuclei for each level and do not show a branching structure on any of our graphs. $(2, 3)$-nucleus decompositions exhibit nice branching and have many more nuclei for each k value. For all graphs, $(2, 3)$-nuclei have many nuclei at each level up to $k = 15$. However, $(3, 4)$-nuclei presents more crowded forests for all graphs. It clearly outperforms $(2, 3)$ on all graphs, except web-NotreDame and as-skitter. Their branchings are quite similar in web-NotreDame graph, where $(2, 3)$ have more nuclei at the top-most levels, whereas the difference on wikipedia-200611 is drastic: $(3, 4)$-nuclei have 4 to 5 times more nuclei than $(2, 3)$ at each level up to $k = 10$.

The $(3, 4)$-nuclei figures provide a useful visualization of the dense subgraph structure. The web-NotreDame has a million edges, and it is not possible to see the graph as a whole. But the forest of nuclei breaks it down into meaningful parts, which can be visually inspected. The overall forest is large (about 2,000 nuclei), but the nesting structure makes it easy to absorb. We have not presented the results here, but even the wikipedia-200611 graph of 38 million edges has forest of only about 4,000 nuclei (which we were able to easily visualize with a drawing tool).

Other choices of r, s for the nuclei do not lead to much branching. We present all nucleus trees for $r < s \leq 4$ for the facebook graph in Figure 9 (except $(3, 4)$, which is given in Figure 3). Clearly, when $r = 1$, the nucleus decomposition is boring. For $r = 2$, some structure arises, but not as dramatically as in Figure 3. Results vary over graphs,



(a) `facebook` (b) `soc-sign-epinions` (c) `web-NotreDame` (d) `wikipedia-200611`

(e) `email` (f) `twitter` (g) `amazon0601` (h) `as-skitter`

Fig. 8. Nuclei counts for each k value in $(1, 2)$- (k-core), $(2, 3)$-, and $(3, 4)$-nucleus decompositions. $(1, 2)$-nuclei exhibits very few nuclei at each level. $(2, 3)$-nuclei and $(3, 4)$-nuclei have nice branchings where the $(3, 4)$-nuclei presents a better view. Note that, on `soc-sign-epinions`, `web-NotreDame`, and `wikipedia-200611` graphs, the number of nuclei decrease after $k = 20$, meaning that leaf nuclei start around those levels (best seen in color).

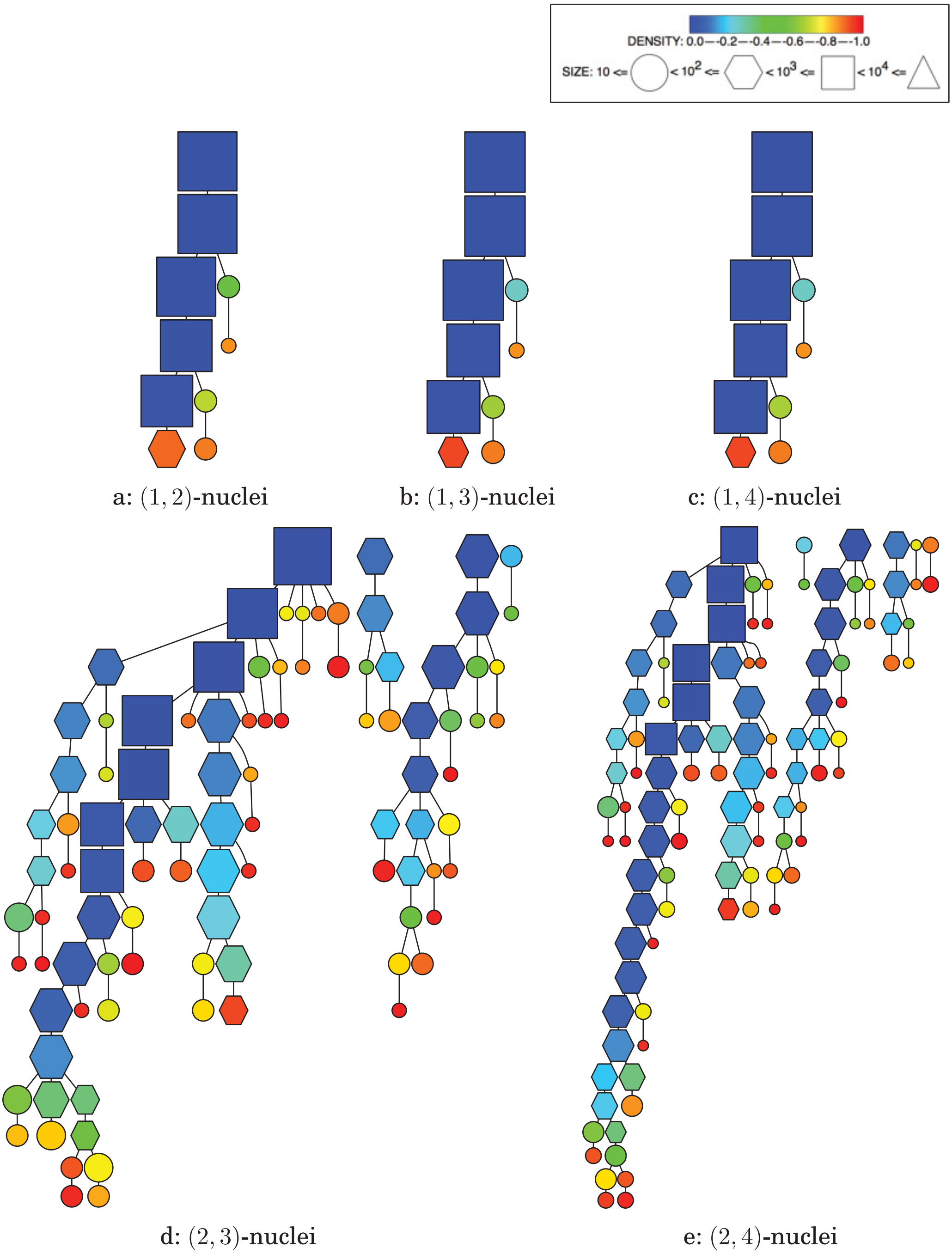
but for $r = 1$, there is pretty much just a chain of nuclei. For $r = 2$, some graphs show more branching, but we consistently see that for $(3, 4)$-nuclei, the forest of nuclei is always branched.

5.1.1. Size Histograms of Nuclei. We present Figure 10 to see the frequency of nuclei sizes obtained by $(1, 2)$-, $(2, 3)$-, and $(3, 4)$-nucleus decompositions on four representative graphs. $(2, 3)$ and $(3, 4)$ are able to produce many nuclei for various sizes, which can not be observed for k-core. Up to 100 vertices, $(3, 4)$ outputs many more nuclei than $(2, 3)$, especially on the `facebook`, `soc-sign-epinions`, and `wikipedia-200611` graphs. And most of those nuclei are quite dense, as also shown in Section 5.2. After 100 vertices, frequencies of nuclei show a similar trend for $(2, 3)$- and $(3, 4)$-nucleus decompositions.

5.2. Dense Subgraph Discovery

We compare $(1, 2)$- (k-core), $(2, 3)$-, and $(3, 4)$-nucleus decompositions with a scatter plot of all nuclei with size of vertices versus density, given in Figure 11. In general, k-core performs worse than the other decompositions, that is, it results in much fewer nuclei with lower densities. $(3, 4)$-nucleus decomposition is able to span the entire spectrum of density and size better than the others. The presence of red dots is more dominant, especially in the `facebook`, `soc-sign-epinions`, and `wikipedia-200611` graphs. Specifically, $(3, 4)$-nucleus decomposition is able to find very dense subgraphs of large sizes, which cannot be found by other decompositions. For example, only $(3, 4)$-nucleus decomposition can find a subgraph of 109 vertices with 0.98 edge density on `facebook` and a clique of 112 vertices on `soc-sign-epinions`. However, there are also few instances where $(2, 3)$-nucleus decomposition is able to find a large and dense subgraph that cannot be found by $(3, 4)$. On `wikipedia-200611`, $(2, 3)$-nucleus decomposition reports a 400 vertex subgraph with 0.44 density and $(3, 4)$ can find a subgraph with 0.28 density in that size range (404 vertices).

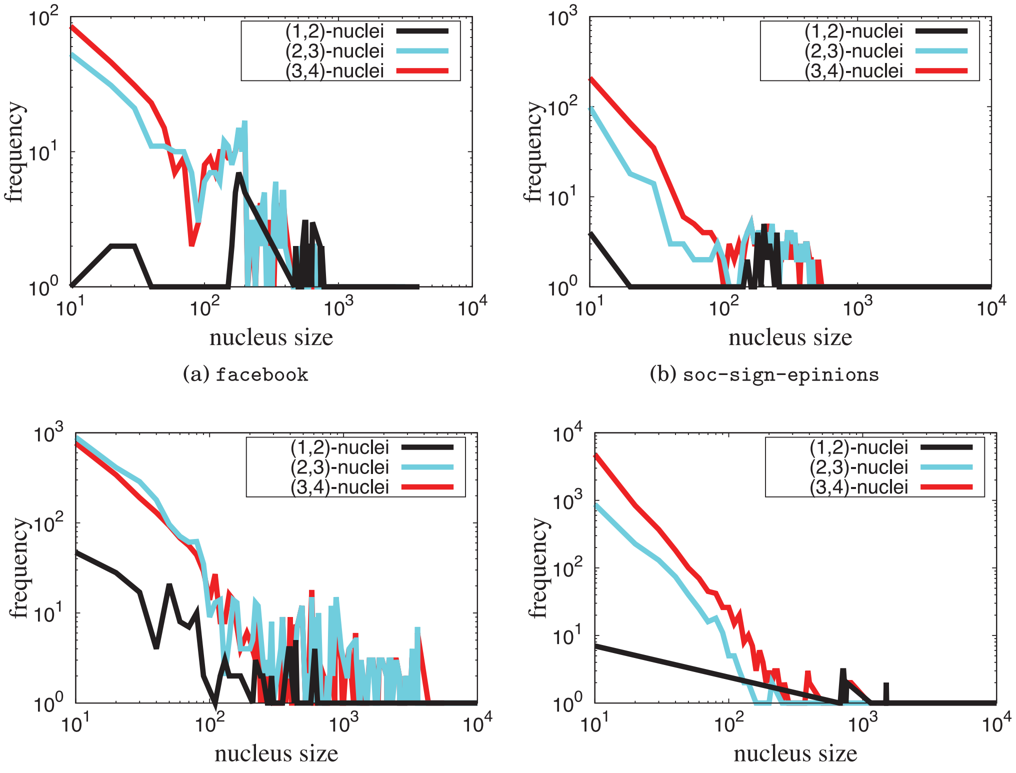
Given the superiority of $(3, 4)$-nucleus decomposition, we plot the density histograms of the $(3, 4)$-nuclei for various graphs in Figure 12. The x-axis is (binned) density, and the y-axis is the number of nuclei (all at least 10 vertices) with that density. It can be


DENSITY: 0.0—-0.2—-0.4—-0.6—-0.8—-1.0
SIZE: 10 <= ○ < 10^2 <= ⬡ < 10^3 <= □ < 10^4 <= △

a: $(1,2)$-nuclei b: $(1,3)$-nuclei c: $(1,4)$-nuclei

d: $(2,3)$-nuclei e: $(2,4)$-nuclei

Fig. 9. (r,s)-nuclei forests for `facebook` when $r < s \leq 4$ (except $(3,4)$, which is given in Figure 3). For $r = 1$, trees are more like chains. Increasing s results in a larger number of internal nodes, which are contracted in the illustrations. There is some hierarchy observed for $r = 2$, but it is not as powerful as for the $(3,4)$-nuclei, that is, the branching structure is more obvious in the $(3,4)$-nuclei (best seen in color).



(a) facebook

(b) soc-sign-epinions

(c) web-NotreDame

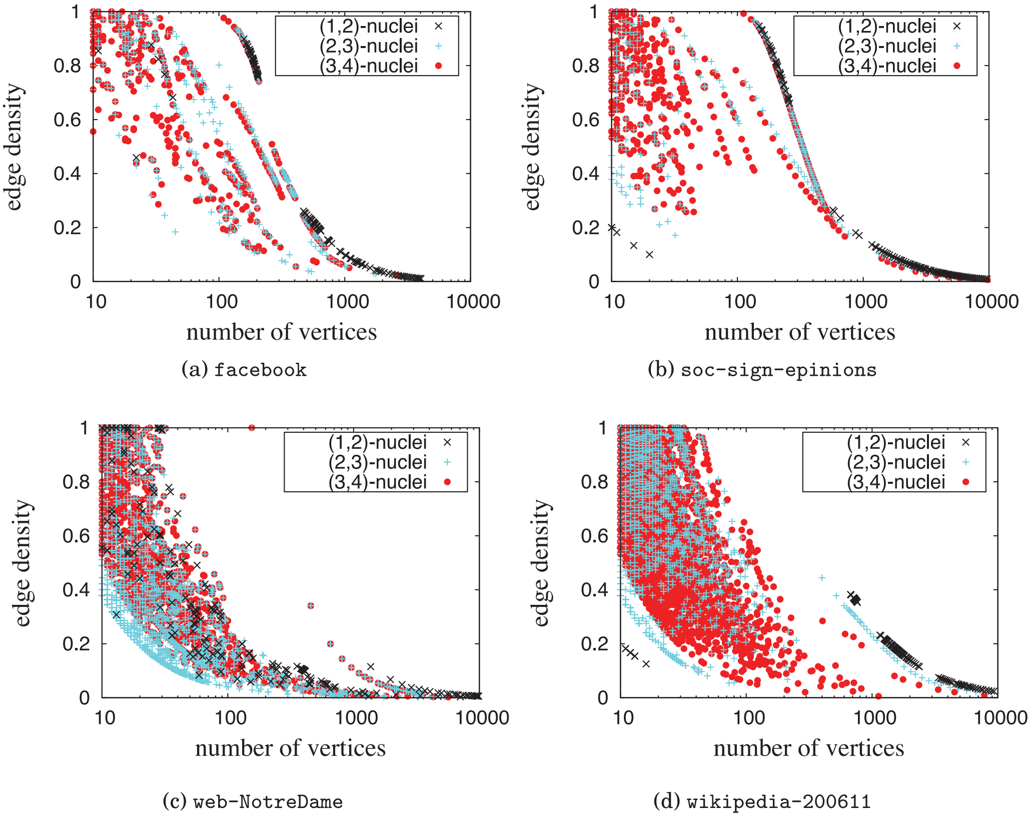
(d) wikipedia-200611

Fig. 10. Nucleus size frequencies for (1, 2)-, (2, 3)-, and (3, 4)-nucleus decompositions. (2, 3) and (3, 4) outputs many more nuclei for various sizes. (3, 4)-nuclei has up to 4 times (on wikipedia-200611) more nuclei than (2, 3) for the range of 10 to 100 vertices (best seen in color).

clearly observed that we find many non-trivial dense subgraphs. It is surprising to see how many near cliques (density > 0.9) we find. We tend to find more subgraphs of high density, and the mass of the histogram is shifted to the right. The number of subgraphs of density at least 0.5 is in the order of hundreds for most graphs.

As shown in Figure 11, an alternate presentation of the dense subgraphs is a scatter plot of all (3, 4)-nuclei with size in vertices versus density. This is given in Figure 13, where the red dots correspond to the (3, 4)-nuclei. We see that dense subgraphs are obtained in all scales of size, which is an extremely important feature. Nuclei capture more than just the densest (or high density) subgraphs but find large sets of lower density (say around 0.2). Note that 0.2 is a significant density for sets of hundreds of vertices.

5.2.1. Comparisons with Previous States of the Art. How does the quality of dense subgraphs found compare to the state of the art? In the scatter plots of Figure 13, we also show the output of two algorithms of Tsourakakis et al. [2013] in green and blue. The idea of Tsourakakis et al. [2013] is to approximate *quasi-cliques*, and their result provides two very elegant algorithms for this process. (We collectively refer to them Optimal Quasi-Cliques, OQC.) OQC algorithms only give a single output, so we performed multiple runs to get many dense subgraphs. This is consistent with what was done in Tsourakakis et al. [2013]. OQC algorithms clearly beat previous heuristics, and it is fair to say that Tsourakakis et al. [2013] is the state of the art.



(a) facebook (b) soc-sign-epinions

(c) web-NotreDame (d) wikipedia-200611

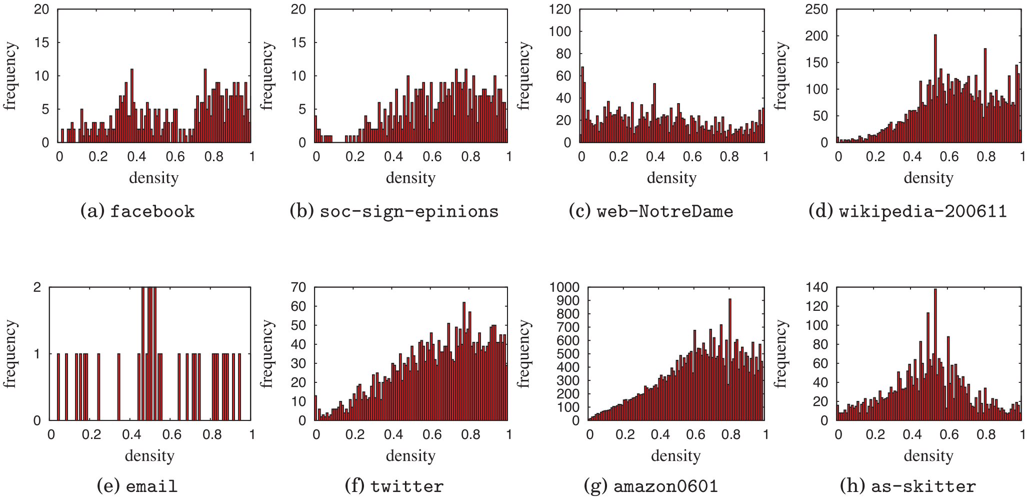
Fig. 11. Density vs. size plots for (1, 2)-, (2, 3)-, and (3, 4)-nucleus decompositions. (3, 4)-nucleus decomposition is able to span the entire spectrum better than others. Large subgraphs of high edge densities can be observed in (3, 4)-nuclei (best seen in color).

The (3, 4)-nucleus decomposition takes significantly longer than the algorithms of Tsourakakis et al. [2013]. But we get denser subgraphs in almost all runs. Moreover, the sizes are comparable if not larger than the output of Tsourakakis et al. [2013]. Surprisingly, in facebook and soc-sign-epinions, some of the best outputs of OQC are very close to (3, 4)-nuclei. Arguably, the (3, 4)-nuclei perform the worst on wikipedia-200611, where OQC finds some larger and denser instances than (3, 4)-nuclei. Nonetheless, the smaller (3, 4)-nuclei are significantly denser. We almost always can find fairly large cliques.

In Table I, we consider the OQC output vs. (3, 4)-nuclei for all graphs. Barring four instances, there is a (3, 4)-nucleus that is larger and denser than the OQC output. In all cases but one (adjnoun), there is a (3, 4)-nucleus of density (of non-trivial size) higher than the the OQC output. The nuclei have the advantage of being the output of a fixed, deterministic procedure, and not a heuristic that may give different outputs on different runs. We mention that OQC algorithms have a significant running time advantage over finding (3, 4)-nuclei for a single subgraph finding.

5.3. Overlapping Nuclei

A critical aspect of nuclei is that they can overlap. Grappling with overlap is a major challenge when dealing with graph decompositions. We believe one of the benefits of nuclei is that they naturally allow for (restricted) overlap. As mentioned earlier, no



(a) facebook (b) soc-sign-epinions (c) web-NotreDame (d) wikipedia-200611

(e) email (f) twitter (g) amazon0601 (h) as-skitter

Fig. 12. Density histograms for $(3, 4)$-nuclei of selected graphs. x-axis (binned) is the density and y-axis is the number of nuclei (at least 10 vertices) with that density. Number of nuclei with the density above 0.8 is significant: 139 for soc-sign-epinions, 355 for web-NotreDame, and 1874 for wikipedia-200611. Also notice that the mass of the histogram is shifted to the right in most graphs.

two (r, s)-nuclei can contain the same r-clique. This is a significant benefit of setting $r = 3, s = 4$ over other choices.

In Figure 14, we plot the histogram over non-trivial overlaps for $(3, 4)$-nuclei (We naturally do not consider a child nucleus intersecting with an ancestor). For a given overlap size in vertices, the frequency is the number of pairs of $(3, 4)$-nuclei with that overlap. This is shown for eight representative graphs. The total number of pairwise overlaps (the sum of frequencies) is typically around half the total number of $(3, 4)$-nuclei. We observed that the Jaccard similarities are less than 0.1 (usually smaller). This suggests that we have large nuclei with some overlap.

There are bioinformatics applications for finding vertices that are present in numerous dense subgraphs [Hu et al. 2005]. The $(3, 4)$-nuclei provide many such vertices. In Figure 15, we give a scatter plot of all intersecting nuclei, where nuclei are indexed by density. For two intersecting nuclei of density $\alpha > \beta$, we put a point (α, β). We only plot pairs where the overlap is at least five vertices. Especially for web-NotreDame, amazon0601, as-skitter, and wikipedia-200611, we get significant overlaps between dense clusters.

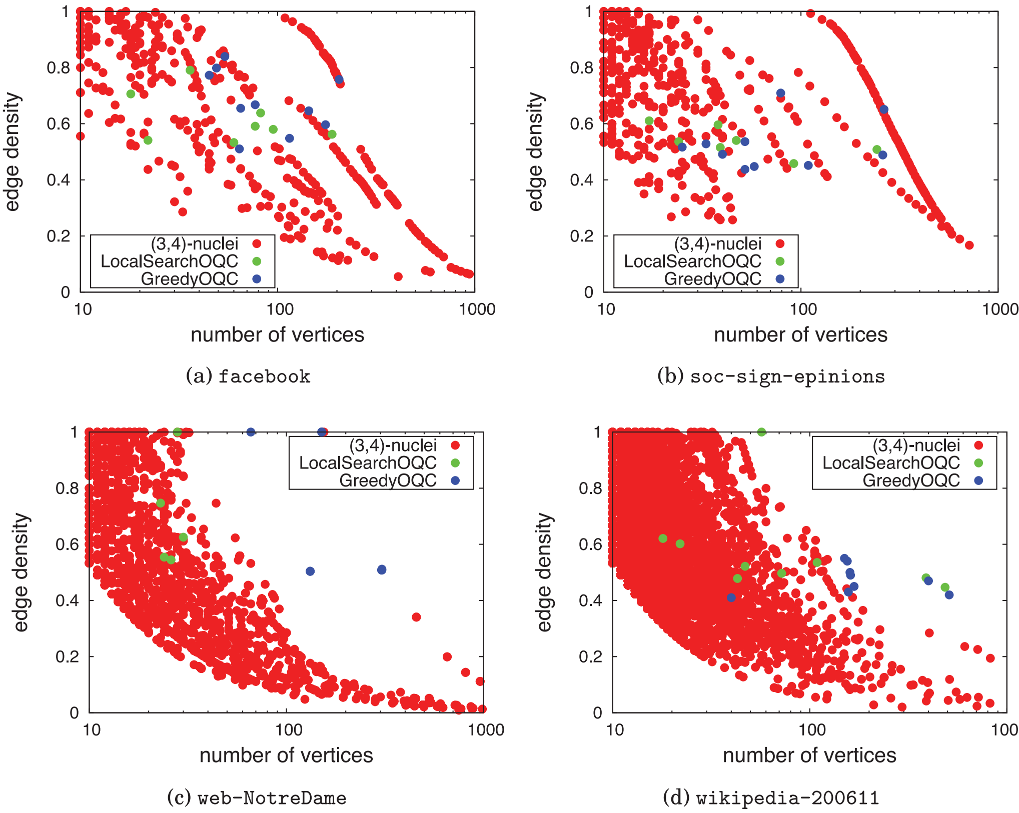
In contrast, for all other settings of r, s, we get almost no overlap. When $r = 2$, nuclei can only overlap at vertices, and this is too stringent to allow for interesting overlap.

5.4. Case Study: Analysis of APS Citation Network

We apply our nucleus decompositions to analyze the citation network of articles published by the APS in its family of journals.[2] The APS citation network has 531,478 vertices and 6,035,617 edges. We ignore the direction of edges and use the citation network as a graph of "related articles." Our aim is to compare the $(1, 2)$-, $(2, 3)$-, and $(3, 4)$-nuclei by checking whether the set of articles in a nucleus is meaningful and see how they differ for each decomposition.

$(1, 2)$ (k-core) decomposition does not result in an interesting hierarchy. It gives only 65 subgraphs of $\geq$5 vertices, and the hierarchy structure is roughly a chain. There are

[2] http://journals.aps.org/datasets.



(a) facebook (b) soc-sign-epinions

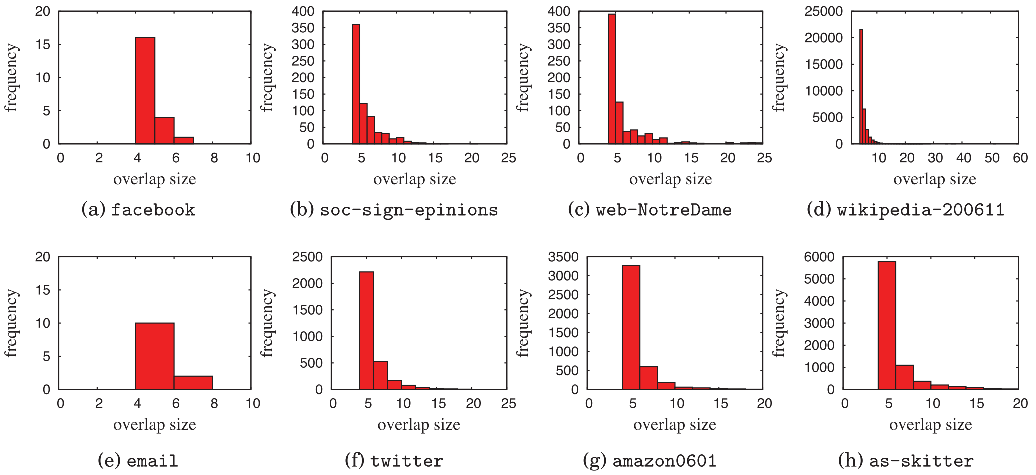
(c) web-NotreDame (d) wikipedia-200611

Fig. 13. Density vs. size plots for nuclei of four graphs. State-of-the-art algorithms are depicted with OQC variants, and they report one subgraph at each run. We ran them 10 times to get a general picture of the quality. Overall, (3, 4)-nuclei are very competitive with the state of the art and produce many subgraphs with high quality and non-trivial sizes (best seen in color).

only 20 subgraphs with more than 0.5 edge density. On the other hand, (2, 3) and (3, 4) algorithms result in 11,300 and 27,838 nuclei of ≥5 vertices. (3, 4) obtained denser subgraphs in a more detailed hierarchy, as observed for the other real-world networks. We focus on the differences between (2, 3)- and (3, 4)-nuclei. For each (2, 3)-nucleus, we checked the best matching nucleus from (3, 4)-nuclei and vice versa. The Jaccard index is used as the similarity metric. The (3, 4)-nuclei matche 92% of the (2, 3)-nuclei with more than 0.5 Jaccard similarity, but the (2, 3)-nuclei find correspondence to only 58% of the (3, 4)-nuclei. Although there exist some dense subgraphs that can only be obtained by (2, 3)-nuclei, (3, 4)-nuclei can be thought of as an approximate superset of (2, 3).

We further investigated the articles in some nuclei from (2, 3) and (3, 4) decompositions and present two interesting observations.

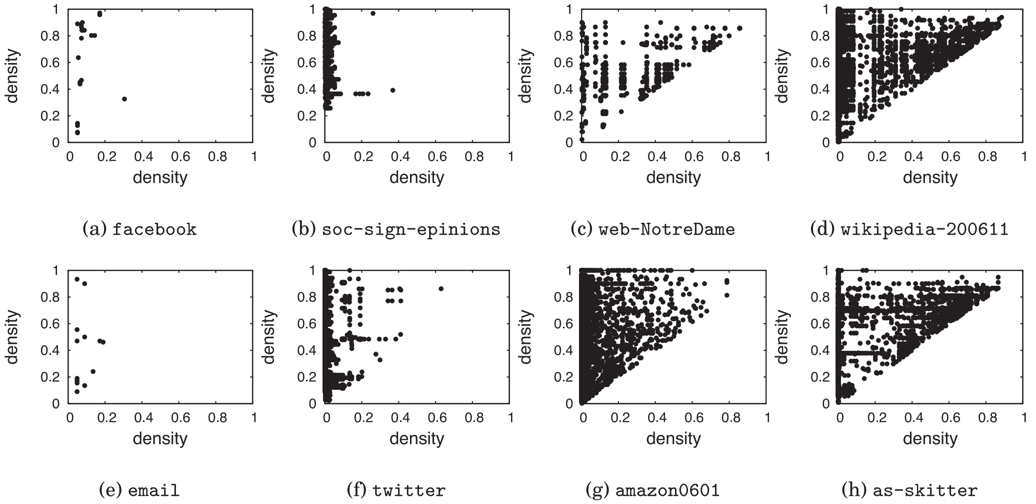
Observation 1: We check the dense subgraphs that can only be found in (3, 4)-nuclei. One such nucleus has 13 articles about the nucleosynthesis subject with 0.9 edge density, and it is located at the leaf level. None of these 13 articles can be found in a (2, 3)-nucleus with fewer than 1,000 vertices. Nucleosynthesis is the process that creates new atomic nuclei from pre-existing nucleons, primarily protons, and neutrons and is studied by nuclear physicists (not to be confused with our nucleus definition). Table II gives the articles with the author and date information. We observe that the



(a) facebook (b) soc-sign-epinions (c) web-NotreDame (d) wikipedia-200611

(e) email (f) twitter (g) amazon0601 (h) as-skitter

Fig. 14. Histograms over non-trivial overlaps for $(3, 4)$-nuclei. Child-ancestor intersections are omitted. Overlap size is in terms of the number of vertices. Most overlaps are small in size. We also observe that $(2, s)$-nuclei give almost no overlaps.



(a) facebook (b) soc-sign-epinions (c) web-NotreDame (d) wikipedia-200611

(e) email (f) twitter (g) amazon0601 (h) as-skitter

Fig. 15. Overlap scatter plots for $(3, 4)$-nuclei. Each axis shows the edge density of a participating nucleus in the pairwise overlap. Larger density is shown on the y-axis. $(3, 4)$-nuclei are able to get overlaps between very dense subgraphs, especially in web-NotreDame, amazon0601, as-skitter, and wikipedia-200611. In the wikipedia-200611 graph, there are $1,424$ instances of pairwise overlap between two nuclei, where each nucleus has the density of at least 0.8.

authors of these articles are mostly different, implying that it is not an artifact of the series of self-cited articles by certain researchers.

This observation shows that higher-order nuclei can help discover structures that will be lost within larger groups if lower-order structures are used.

Observation 2: Complex networks are well studied by physicists. We find the subgraphs in $(2, 3)$- and $(3, 4)$-nuclei that include two seminal articles on complex networks: (1) “Epidemic Spreading in Scale-Free Networks,” by Pastor-Satorras and Vespignani (10.1103/PhysRevLett.86.3200), and (2) “Random Graphs with Arbitrary Degree Distributions and Their Applications,” by Newman, Strogatz, and Watts

Table II. 13 Papers about Nucleosynthesis in the (3, 4) Nucleus

Authors	Paper	Year
Bergstrom et al.	Constraints on the variation of the fine structure constant from big bang nucleosynthesis	1999
Ichikawa, Kawasaki	Constraining the variation of the coupling constants with big bang nucleosynthesis	2002
Nollett, Lopez	Primordial nucleosynthesis with a varying fine structure constant An improved estimate	2002
Yoo, Scherrer	Big bang nucleosynt. and cosmic m.wave backg. constr. on the time var. of the Higgs vacuum exp. val.	2003
Kneller, McLaughlin	Big bang nucleosynthesis and $\wedge_{QCD}$	2003
Dmitriev et al.	Cosmological var. of the deuteron bind. energy, strong inter., and quark masses f. big bang nucleosyn.	2004
Muller et al.	Nucleosynthesis and the variation of fundamental couplings	2004
Li, Chu	Big-bang nucleosynthesis with an evolving radion in the brane world scenario	2006
Coc et al.	Coupled variations of fundamental couplings and primordial nucleosynthesis	2007
Dent et al.	Primordial nucleosynthesis as a probe of fundamental physics parameters	2007
Landau et al.	Early universe constraints on time variation of fundamental constants	2008
Dent et al.	Unifying cosmological and recent time variations of fundamental couplings	2008
Coc et al.	Variation of fundamental constants and the role of A=5 and A=8 nuclei on primordial nucleosynthesis	2012

Note: Edge Density of the Nucleus Is 0.9.

(10.1103/PhysRevE.64.026118). Both articles had a strong impact on the literature and were cited more than 3,000 times. We observe that the smallest $(2, 3)$-nucleus that includes those two articles has 155 articles with 0.12 density. They are all about the different aspects of complex networks like scale-free and small-world properties. On the other hand, the smallest subgraph that contains those two articles in the $(3, 4)$-nuclei has 104 articles with 0.18 density, and it is a subset of that $(2, 3)$-nucleus with 155 articles. Details of the mentioned nuclei and relations are given in Figure 16. The 51 different articles between the two nuclei are mostly about synchronization networks, which are known as the network of coupled dynamical systems that consists of connected oscillators where oscillators are the vertices that emit and receive signals. So the question is as follows: Where are those articles on synchronization networks in the $(3, 4)$-nuclei? We find most of them in a nucleus with 25 articles and 0.55 density and located as a sibling of the nucleus that has 104 articles. Another sibling is a nucleus with 19 articles and 0.71 density, and none of them appears in the 155-article $(2, 3)$-nucleus. In $(3, 4)$-nuclei, we observe further meaningful sets of articles on epidemic spreading and percolation theory. The article set on percolation theory is also reported in the $(2, 3)$-nuclei.

Overall, this observation shows that higher-order nuclei can provide a more fine-grained breakdown of the dense structures. Here $(3, 4)$ decomposition can match what $(2, 3)$ decomposition does and provides further information about how a dense structure is decomposed.

We believe that our nucleus decompositions can be used in different application domains to find densely connected subgraphs in adjustable granularity and to see the relations among those subgraphs in the hierarchy structure.

5.5. Runtime Results

Table I presents the runtimes in seconds for the entire construction. To provide some context, we describe runtimes for varying choices of r, s. For $r = 1, s = 2$ (k-cores), the

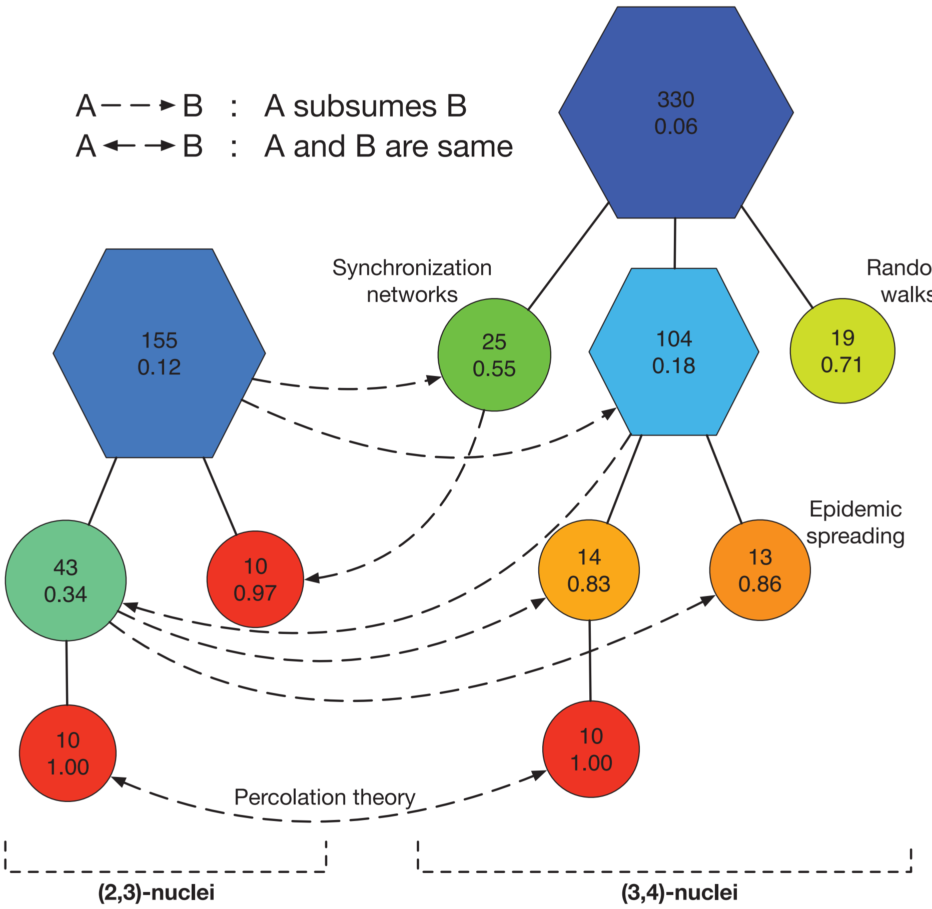


Fig. 16. (2, 3)- and (3, 4)-nuclei of the APS citation network on a complex networks subject. One-way arrows denote containment relations, and double arrows indicate the same nuclei. Subject of the articles in a nucleus is given next to them whenever possible.

decomposition is linear and extremely fast. For the largest graph (`wikipedia-200611`) we have, with $39M$ edges, that it takes only 4.26s. For $r = 2, s = 3$, the time can be two orders of magnitude higher. For $(3, 4)$-nuclei, it is an additional order of magnitude higher. Nonetheless, our most expensive run took less than an hour on the `wikipedia-200611` graph, and the final decomposition is quite insightful. It provides about 6,000 nuclei with more than 10 vertices, most of them of have density of at least 0.4. The algorithms of Tsourakakis et al. [2013] take roughly a minute for `wikipedia-200611` to produce *only one* dense subgraph.

The theoretical runtime analysis of Theorem 4.3 gives a runtime bound of $\sum_v c_3(v)d(v)$. In Table I, we show this value for the various graphs. In general, we note that this value roughly correlates with the runtime. For graphs where the runtime is in many minutes, this quantity is always in the billions. For the large wiki graphs where the $(3, 4)$-nucleus decomposition is most expensive, this is in the trillions.

6. FUTURE DIRECTIONS

The most important direction is in the applications of nucleus decompositions. We are currently investigating bioinformatics applications, specifically protein-protein and protein-gene interaction networks. Biologists often want a global view of the dense substructures, and we believe the $(3, 4)$-nuclei could be extremely useful here. In our preliminary analyses, we wish to see if the nuclei pick out specific functional units. If so, then that would provide strong validation of dense subgraph analyses for bioinformatics.

It is natural to try even larger values of r, s. Preliminary experimentation suggested that this gave little benefit in either the forest or the density of nuclei. Also, the cost of clique enumeration becomes forbiddingly large. It would be nice to argue that $r = 3, s = 4$ is a sort of sweet spot for nucleus decompositions. Previous theoretical work suggests that any graph with a sufficient triangle count undergoes special "community-like" decompositions [Gupta et al. 2014]. That might provide evidence to why triangle-based nuclei are enough.

A faster algorithm for the $(3, 4)$-nuclei is desirable. Clique enumeration is a well-studied problem [Bron and Kerbosch 1973], and we hope techniques from these results may provide ideas here. Of course, as we said earlier, any method based on storing 4-cliques is infeasible (spacewise). We hope to devise a clever algorithm or data structure that quickly determines the 4-cliques that a triangle participates in.

Last but not least, we seek for incremental algorithms to maintain the (r, s)-nuclei for a stream of edges. There are existing techniques for streaming k-core algorithms [Sariyüce et al. 2013], and we believe that similar methods can be adapted for (r, s)-nuclei maintenance.

7. CONCLUSIONS

We introduced the nucleus decomposition of a graph to extract the hierarchy of dense subgraphs. Our algorithms are generalizations of well known k-core, and k-truss concepts, and for specific parameters, our experimental evaluation showed that generalized nucleus decompositions provide better hierarchies and denser subgraphs than the state of the art. We believe that our contributions will have an impact on better understanding the relations between dense subgraphs in a graph and enable better insights for large and complex graph data.

ACKNOWLEDGMENTS

We are grateful to Charalampos Tsourakakis for sharing his code base for Tsourakakis et al. [2013]. We also thank the anonymous reviewers for many helpful suggestion to improve this article.

REFERENCES

A. B. Adcock, B. D. Sullivan, and M. W. Mahoney. 2013. Tree-like structure in large social and information networks. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 1–10.

J. Ignacio Alvarez-Hamelin, Alain Barrat, and Alessandro Vespignani. 2006. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in Neural Information Processing Systems 18*. 41–50.

R. Andersen and K. Chellapilla. 2009. Finding dense subgraphs with size bounds. In *Proceedings of the Workshop on Algorithms and Models for the Web-Graph (WAW)*. 25–37.

A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. 2012. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *Proc. VLDB Endow.* 5, 6 (Feb. 2012), 574–585.

Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. 2000. Greedily finding a dense subgraph. *J. Algor.* 34, 2 (Feb. 2000), 203–221.

Oana Denisa Balalau, Francesco Bonchi, T.-H. Hubert Chan, Francesco Gullo, and Mauro Sozio. 2015. Finding subgraphs with maximum total density and limited overlap. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining (WSDM'15)*. 379–388.

D. J. Beal, R. Cohen, M. J. Burke, and C. L. McLendon. 2003. Cohesion and performance in groups: A meta-analytic clarification of construct relation. *J. Appl. Psychol.* 88 (2003), 989–1004.

J. W. Berry, L. K. Fostvedt, D. J. Nordman, C. A. Phillips, C. Seshadhri, and A. G. Wilson. 2014. Why do simple algorithms for triangle enumeration work in the real world? In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS'14)*. ACM, New York, NY, 225–234.

Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos Tsourakakis. 2015. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC'15)*. 173–182.

C. Bron and J. Kerbosch. 1973. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM* 16, 9 (Sep. 1973), 575–577.

G. Buehrer and K. Chellapilla. 2008. A scalable pattern mining approach to web graph compression with communities. In *Proc. of the 2008 International Conference on Web Search and Data Mining (WSDM'08)*. 95–106.

M. Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'00)*. 84–95.

N. Chiba and T. Nishizeki. 1985. Arboricity and subgraph listing algorithms. *SIAM J. Comput.* 14, 1 (Feb. 1985), 210–223.

J. Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report (2008).

J. Cohen. 2009. Graph twiddling in a mapreduce world. *Comput. Sci. Eng.* 11 (2009), 29–41.

UF Sparse Matrix Collection. University of Florida Sparse Matrix Collection. Retrieved March 2014 from http://www.cise.ufl.edu/research/sparse/matrices/.

P. Colomer de Simon, M. Serrano, M. G. Beiro, J. I. Alvarez-Hamelin, and M. Boguna. 2013. Deciphering the global organization of clustering in real complex networks. *Sci. Rep.* 3, 2517 (2013).

Y. Dourisboure, F. Geraci, and M. Pellegrini. 2007. Extraction and classification of dense communities in the web. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*. 461–470.

Xiaoxi Du, Ruoming Jin, Liang Ding, Victor E. Lee, and John H. Thornton, Jr. 2009. Migration motif: A spatial - temporal pattern mining approach for financial markets. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM, New York, NY, 1135–1144.

A. Epasto, S. Lattanzi, and M. Sozio. 2015. Efficient densest subgraph computation in evolving graphs. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*. 300–310.

P. Erdős and A. Hajnal. 1966. On chromatic number of graphs and set-systems. *Acta Math. Hung.* 17 (1966), 61–99.

U. Feige. 2002. Relations between average case complexity and approximation complexity. In *Proceedings of the Symposium on Theory of Computing*. 534–543.

D. R. Forsyth. 2010. *Group Dynamics*. Cengage Learning.

A. P. Francisco and A. L. Oliveira. 2011. Fully generalized graph cores. In *Complex Networks*. Vol. 116. 22–34.

E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou. 2006. MotifCut: Regulatory motifs finding with maximum density subgraphs. In *ISMB (Supplement of Bioinformatics)* (2006-08-28). 156–157.

G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. 1989. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* 18, 1 (Feb. 1989), 30–55.

D. Gibson, R. Kumar, and A. Tomkins. 2005. Discovering large dense subgraphs in massive graphs. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB'05)*. 721–732.

A. Gionis, F. Junqueira, V. Leroy, M. Serafini, and I. Weber. 2013. Piggybacking on social networks. *Proc. VLDB Endow.* 6, 6 (2013), 409–420.

A. V. Goldberg. 1984. *Finding a Maximum Density Subgraph*. Technical Report. Berkeley, CA, USA.

R. Gupta, T. Roughgarden, and C. Seshadhri. 2014. Decompositions of triangle-dense graphs. In *Innovations in Theoretical Computer Science (ITCS)*. 471–482.

J. Håstad. 1996. Clique is hard to approximate within $n^{(1-\epsilon)}$. In *Acta Mathematica*. 627–636.

H. Hu, X. Yan, Y. Huang, J. Han, and X. J. Zhou. 2005. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics* 21, 1 (Jan. 2005), 213–221.

Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the ACM SIGMOD International Conf. on Management of Data*. 1311–1322.

L. D. Iasemidis, D.-S. Shiau, W. Chaovalitwongse, J. C. Sackellares, P. M. Pardalos, J. C. Principe, P. R. Carney, A. Prasad, B. Veeramani, and K. Tsakalis. 2003. Adaptive epileptic seizure prediction system. *IEEE. Biomed. Eng.* 50 (2003), 616–627.

R. Jin, Y. Xiang, N. Ruan, and D. Fuhry. 2009. 3-HOP: A high-compression indexing scheme for reachability query. In *Proceedings of the SIGMOD Conference*. 813–826.

S. Khot. 2006. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput.* 36, 4 (2006), 1025–1071.

Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*. 597–608.

R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. 1999. Trawling the web for emerging cyber-communities. In *Proc. of the Eighth International Conference on World Wide Web (WWW'99)*. 1481–1493.

V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal. 2010. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Vol. 40.

Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. 2008. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web (WWW'08)*. ACM, New York, NY, 695–704.

D. Lick and A. White. 1970. k-degenerate graphs. *Can. J. Math.* 22 (1970), 1082–1096.

D. Matula and L. Beck. 1983. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM* 30, 3 (1983), 417–427.

R. J. Mokken. 1979. Cliques, clubs and clans. *Qual. Quant.* 13, 2 (1979), 161–173.

R. A. Rossi, D. F. Gleich, A. H. Gebremedhin, and Md. M. A. Patwary. 2013. A fast parallel maximum clique algorithm for large sparse graphs and temporal strong components. *CoRR* abs/1302.6256 (2013).

K. Saito and T. Yamada. 2006. Extracting communities from complex networks by the k-dense method. In *Sixth IEEE International Conference on Data Mining Workshops, 2006 (ICDM Workshops 2006)*. 300–304.

A. Sala, L. Cao, C. Wilson, R. Zablit, Haitao Zheng, and Ben Y. Zhao. 2010. Measurement-calibrated graph models for social network experiments. In *WWW'10*. ACM, 861–870.

A. E. Sariyüce, B. Gedik, G. Jacques-Silva, K. L. Wu, and Ü. V. Çatalyürek. 2013. Streaming algorithms for k-core decomposition. In *Proc. VLDB Endow.* 433–444.

A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek. 2015. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*. 927–937.

T. Schank and D. Wagner. 2005. Finding, counting and listing all triangles in large graphs, an experimental study. In *Experimental and Efficient Algorithms*. 606–609.

S. B. Seidman. 1983. Network structure and minimum degree. *Soc. Netw.* 5, 3 (1983), 269–287.

S. B. Seidman and B. Foster. 1978. A graph-theoretic generalization of the clique concept. *J. Math. Sociol.* (1978).

C. Seshadhri, A. Pinar, and T. G. Kolda. 2014. Triadic measures on graphs: The power of wedge sampling. *Stat. Anal. Data Min.* 7, 4 (2014), 294–307.

SNAP. retrieved March, 2014. Stanford Network Analysis Package. Retrieved March 2014 http://snap.stanford.edu/snap.

S. Suri and S. Vassilvitskii. 2011. Counting triangles and the curse of the last reducer. In *WWW'11*. 607–614.

N. Tatti and A. Gionis. 2015. Density-friendly graph decomposition. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1089–1099.

C. Tsourakakis. 2015. The k-clique densest subgraph problem. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1122–1132.

C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. 2013. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*.

J. Wang and J. Cheng. 2012. Truss decomposition in massive networks. *Proc. VLDB Endow.* 5, 9 (2012), 812–823.

N. Wang, J. Zhang, K. L. Tan, and A. K. H. Tung. 2010. On triangulation-based dense neighborhood graph discovery. *Proc. VLDB Endow.* 4 (2010), 58–68.

S. Wasserman and K. Faust. 1994. *Social Network Analysis: Methods and Applications*. Cambridge University Press.

D. Watts and S. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393 (1998), 440–442.

B. Zhang and S. Horvath. 2005. A general framework for weighted gene co-expression network analysis. *Stat. Appl. Genet. Molec. Biol.* 4, 1 (2005), Article 17+.

Y. Zhang and S. Parthasarathy. 2012. Extracting analyzing and visualizing triangle k-core motifs within networks. In *Proc. of the 2012 IEEE 28th International Conference on Data Engineering (ICDE'12)*. 1049–1060.

F. Zhao and A. K. H. Tung. 2013. Large scale cohesive subgraphs discovery for social network visual analysis. In *Proc. VLDB Endow.* 85–96.

Received November 2015; revised December 2016; accepted January 2017