The k-peak decomposition: Mapping the global structure of graphs

Priya Govindan
Rutgers University
priyagn@cs.rutgers.edu

Chenghong Wang
Syracuse University
cwang132@syr.edu

Chumeng Xu
Cornell University
cx87@cornell.edu

Hongyu Duan
Syracuse University
hoduan@syr.edu

Sucheta Soundarajan
Syracuse University
susounda@syr.edu

ABSTRACT

The structure of real-world complex networks has long been an area of interest, and one common way to describe the structure of a network has been with the k-core decomposition. The core number of a node can be thought of as a measure of its centrality and importance, and is used by applications such as community detection, understanding viral spreads, and detecting fraudsters. However, we observe that the k-core decomposition suffers from an important flaw: namely, it is calculated globally, and so if the network contains distinct regions of different densities, the sparser among these regions may be neglected.

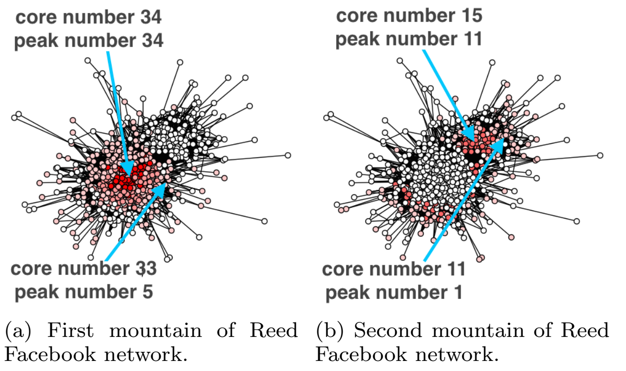
To resolve this issue, we propose the ***k*-peak** graph decomposition method, based on the k-core algorithm, which finds the centers of distinct regions in the graph. Our contributions are as follows: (1) We present a novel graph decomposition- the k-peak decomposition- and corresponding algorithm, and perform a theoretical analysis of its properties. (2) We describe a new visualization method, the *'Mountain Plot'*, which can be used to better understand the global structure of a graph. (3) We perform an extensive empirical analysis of real-world graphs, including technological, social, biological, and collaboration graphs, and show how the k-peak decomposition gives insight into the structures of these graphs. (4) We demonstrate the advantage of using the k-peak decomposition in various applications, including community detection, contagion and identifying essential proteins.

CCS Concepts

•**Mathematics of computing** → **Graph theory;** •**Human-centered computing** → *Visualization techniques;*

Keywords

Graphs; k-core; k-peak; graph visualization



(a) First mountain of Reed Facebook network. (b) Second mountain of Reed Facebook network.

Figure 1: Mountains (distinct regions) obtained as a result of k-peak decomposition. The most central (highest peak number) nodes, peripheral nodes and nodes outside of the mountain are shown in dark red, in lighter shades of red and in white, respectively. Pruning the graph by core number would have found the peripheral nodes of the first mountain before finding the central nodes of the second mountain. Note that the second mountain consists of 2 components. (Best viewed in color.)

1. INTRODUCTION

In recent years, there have been significant efforts to characterize the structure of real-world graphs, which requires an understanding of both global as well as local structure. It is thus interesting and useful to characterize the globally central parts of the graph as well as the local inter-dependence of nodes. The k-core graph decomposition has become a common way to describe the global structure of a graph. A k-core is the maximal subgraph with minimum degree k in the subgraph. This decomposition has well-understood properties [20, 3], and several efficient algorithms have been proposed for finding it, including algorithms for large graphs or streaming settings [7, 16, 19]. The k-core decomposition has achieved widespread success for applications that require finding a central, well-connected subgraph that, in a sense, captures the most important structure of the graph.

Weakness of the *k*-core decomposition: While useful, the k-core decomposition suffers from one important weakness, illustrated as follows: Suppose that a graph contains distinct and independent regions of different edge densities. In such cases, the k-core decomposition will fail to find a good global representation of the graph, instead focusing on the densest of these regions. If one wishes to find the central regions across the entire graph, the k-core would not be effective.

For example, consider the Facebook friendship graph of all the users in the world. In Iceland, 100% of the population has Internet access, while in Eritrea, only 1.1% of the population has Internet access. If one finds the k-core of the graph for some large value of k, this core will likely capture the important aspects of the Icelandic region, but none of the Eritrean region. Naturally, any method that finds a core or other type of dense subgraph will exclude many nodes, but the critical observation in this case is that *the structure of the dense region is independent of the structure of the sparser, excluded region.* If one lowers the value of k to capture the sparser region, too much of the dense region will be captured, negating the gains obtained by finding the core. Note that this observation is in contrast to the traditional notion of a network with strong core-periphery structure, in which a high k-core, while excluding peripheral nodes, would capture the essential structure of the entire graph. In this example, the k-core would be much more informative if one could set different values of k for different regions.

Moreover, the k-core decomposition does not capture the local structure in graphs. To understand the local structure, one might ask the following questions: *How does the core number of a node depend on those of its neighbors? What are the patterns of inter-dependence between nodes?*

The proposed k-peak decomposition: To address this weakness, we propose the k-peak decomposition. Intuitively, the k-peak is similar to the k-core decomposition, except that different values of k are used in different regions of the graph. The k-peak decomposition divides the graph into separate 'mountains', each of which represents a different region of the graph. Alternatively, one can view the the k-peak decomposition as a notion of centrality that gives equal consideration to nodes in sparser regions of the networks. For example, Figure 1 shows the two highest mountains in the Reed University Facebook network. Each mountain contains a central region (dark red), as well as nodes dependent on this region (lighter shades of red). We propose an efficient algorithm ($O((\sqrt{N})(M+N))$ running time for N nodes and M edges) to find this decomposition and present a novel visualization method that allows one to obtain a global picture of the network's structure with respect to the k-peak decomposition.

Additionally, we show how the k-peak decomposition can be used to gain a nuanced understanding of the relationships between a node and its region. To illustrate its value, we perform this decomposition on a multitude of graphs from different domains. Finally, we show how the k-peak can replace or supplement the k-core in a variety of applications.

Distinctions from related concepts: The proposed k-peak decomposition is related to the concepts of community structure and core-periphery structure, but differs from these notions in important ways. A 'community' is generally thought of as a well-connected group of nodes, while a mountain in the k-peak decomposition is a region that may contain multiple communities. A mountain is thus something coarser than, but also fundamentally different from, a community, as it lacks the expectation of well-connectedness.

The k-peak decomposition is more closely related to the notion of core-periphery structure; indeed, the regions found by this decomposition can sometimes be thought of as core-periphery structures, and others have observed the presence of multiple core-periphery structures in a network [18]. However, the k-peak decomposition and corresponding mountain plot offer significantly more nuance and insight into graph regions than simply identifying them as core-periphery structures. In our descriptive study in Section 6, we show how the mountain plot can be used to identify narrow, 'column'-like structures, classical core-periphery structures, as well as many other patterns.

Our contributions are summarized as follows: **(1)** We propose the **novel k-peak graph decomposition** to understand the global structure of distinct regions within a graph, and present an **efficient algorithm** to find it. **(2)** We perform a **rigorous theoretical analysis** of the properties of the k-peak decomposition. **(3)** We create the 'mountain plot', a **new visualization technique** to understand a graph's global structure. **(4)** We use the k-peak decomposition to **empirically analyze** the structures of 22 diverse real graphs. **(5)** We show how the k-peak decomposition can be used in various k-core-based **applications** to achieve improved results (e.g., up to an average of 53% improvement in spreading a contagion through a network).

2. PREVIOUS WORK

The k-core was introduced by Seidman [20] as a concept to find densely connected subgraphs. Batagelj and Zaversnik [3] proposed an in-memory algorithm to compute k-cores in a graph. The wide interest in the use of k-core decomposition led to the development of efficient algorithms in distributed[15], streaming[19] and external memory[7][16] settings. The well-understood properties of k-core and the efficient algorithms to compute it have encouraged its use in solving several problems. The k-core decomposition has been used in applications such as community detection[17], improving contagion[11], in choosing nodes for network experiments[21] and finding essential proteins in a protein-protein interaction network[23]. Liu et al. [13] proposed a similar modification to the k-core algorithm as our algorithm but their goal was to perform clustering.

There has been considerable research on studying the k-core decomposition, its process, and how to generalize it. The k-core decomposition process has been shown to reveal an ordering of nodes based on the iteration at which it was 'peeled' from the graph[2][9]. There has been considerable research on generalizing and extending the definition of k-core to higher structures. A k-truss defined on an edge was proposed by Cohen [5]. More recently, Sariyuce et al. [6] further extended the generalization of k-core by defining it on a subgraph. The goal of all the previous graph decompositions based on k-cores was to find dense subgraphs, rather than globally distinct regions. To the best of our knowledge, the graph decomposition presented in this paper is the first to define and find distinct regions in a graph, that have properties similar to k-core .

The global structure of a graph as a core-periphery structure was formalized by Borgatti and Everett [4] as a single core-periphery component. Rombach et. al[18] extended the definition to a more general model that allows for multiple core-periphery structures. An alternate model based on clustering co-efficient was proposed by Holme[8]. Zhang et. al[24] proposed a stochastic block model based approximate algorithm to find core-peripheries in graphs. The goal of these methods is to clearly define a core-periphery structure and design algorithms to find them. In contrast, the goal of our work is to *reveal distinct regions (mountains) in a graph*, using a known efficient algorithm (k-core). Although some of the mountains returned by k-peak decomposition are similar to core-periphery structures, the mountains also reveal interesting structures quite different from an ideal core-periphery. Moreover, rather than simply labeling or quantifying a region as a core-periphery structure, the k-peak decomposition allows an understanding of the specific shape and dependencies within that region.

Wang et al [22] presented a visualization technique to maps dense subgraphs. The mountain plots presented in this paper are similar to their work with respect to the goal of plotting groups of nodes in a 2-dimensional plot, with the nodes ordered on the x-axis. But, our method differs in both the subgraphs (mountains) found and the ordering of the nodes in the plot. Specifically, the mountains found, depend on the k-peak decomposition algorithm and the ordering of nodes in a mountain in the plot depends on both the core and peak numbers. The plotting of both the core and peak numbers in the mountain plot gives information about the internal structure of the subgraphs that form the mountains.

3. K-PEAK DECOMPOSITION

Because the k-peak decomposition is based on the k-core decomposition, we first provide a review of key k-core definitions. We then introduce our proposed decomposition and related concepts.

Recall that our goal behind the k-peak decomposition is twofold: first, to identify the distinct regions of the network and second, to obtain a deeper understanding of the relationships and dependencies between nodes.

3.1 Preliminaries

A k-core is defined as follows:

Definition 1. ***k-core:*** *Given a graph G, a k-core is a maximal subgraph G_{k-core} such that each node in G_{k-core} has degree at least k in G_{k-core}.*

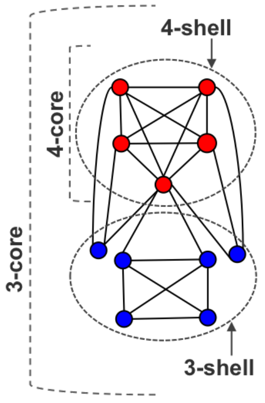
The *core number* of a node is the highest value k such that the node is a part of a k-core. The *k-core decomposition* is the assignment of core numbers to nodes. To find a k-core, the k-core decomposition algorithm recursively removes nodes with degree less than k. The graph *degeneracy* is the highest k such that there exists a k-core in the graph and the *degeneracy core* is the corresponding k-core. The set of all nodes with core number k is the *k-shell*. Alternatively:

Definition 2. ***k-shell:*** *Given a graph G, a k-shell is the induced subgraph over the maximal set of nodes such that (1) the k-shell does not include nodes from any existing higher shells (i.e., p-shells where $p > k$), and (2) each node in the k-shell has at least k connections to nodes in the k-shell or higher shells.*

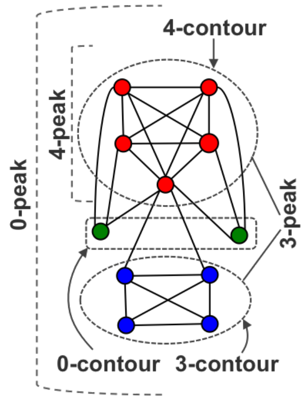
Note that the k-core is the induced subgraph of the union of j-shells for $j \geq k$.

3.2 Proposed k-Peak Graph Decomposition

We posit that the global structure of a network can be viewed as a set of regions, each of which resembles a mountain with a central 'peak'. Mountains are drawn on topographical maps, with contours representing different elevations. Our terminology is based on this analogy.



(a) k-shells and k-cores



(b) k-contours and k-peaks

Figure 2: k-core and k-peak decomposition in a toy graph. The peak number of a node (the k-contour it belongs to) is at most its core number (the k-shell it belongs to).

First, we define a k-contour in a manner analogously to the k-shell:

Definition 3. ***k-contour:*** *Given a graph G, a k-contour is the induced subgraph over the maximal set of nodes such that (1) the k-contour does not include nodes from a higher contour (i.e., a p-contour where $p > k$), and (2) each node in the k-contour has at least k connections to other nodes in the k-contour .*

The difference between the k-contour and k-shell definitions is that in a k-shell, we count a node's connections to higher shells, whereas in a k-contour, we only count connections within the same contour.

The definition of a k-contour can be restated as follows:

Definition 4. ***k-contour:*** *Given a graph G, let d be the degeneracy of G. A k-contour of G is a maximal induced subgraph $G_{k-contour}(V_{k-contour}, E_{k-contour})$ such that*

$$G_{k-contour} = \begin{cases} G_{k-core} & \text{if } k = d \\ (G \setminus \bigcup_{j=k+1}^{d} G_{j-contour})_{k-core} & \text{if } k < d \end{cases}$$

If the degeneracy of the graph is d, then the d-contour is equivalent to the d-shell and the d-core is the induced subgraph of the d-contour.

The ***peak number*** of a node is the value k such that the node belongs to a k-contour.[1]

Definition 5. ***k-peak decomposition:*** *Given a graph $G(V, E)$, a k-peak decomposition is defined as the assignment of each node to exactly one k-contour.*

See Figure 2 for a toy example. Notice that the peak number of the two nodes with degree 3 (shown as green nodes in Fig 2b), is less than their core number.

Similar to the k-core, we define the k-peak as the induced subgraph of the union of j-contours, where $j \geq k$.

Definition 6. ***k-peak:*** *Given a graph G, a k-peak is the induced subgraph of the union of j-contours, $\forall j \geq k$.*

[1]Note that each node can belong to only one contour.

For a given graph G, the k-core decomposition is unique- i.e., each node has a single unique core number. In Theorem 1 below, we show that the k-peak decomposition is also unique- i.e., each node has a single unique peak number.

Theorem 1. *Given a graph G and $k \in \mathbb{Z}_+$, the k-peak is unique for all $k > 0$.*

Proof. Let d be the degeneracy of graph G. First, consider values of $k > d$. It is clear that there is no non-empty k-peak, because if such a k-peak existed, there would be a non-empty j-contour for some $j \geq k$. But then there would be a set of nodes, each with at least j connections to other nodes in the set; i.e., a j-core. But we assumed the d-core was the highest core of the graph. Contradiction, so the k-peak is empty, and so is unique.

Next, consider $k = d$. Then the d-core satisfies Definition 3 of the d-contour: (1) There are no higher contours, so no node in the d-core belongs to a higher contour, and (2) By Definition 1 of a k-core, each node in the d-contour has at least d edges to other nodes in the d-contour. Thus, the k-peak is simply the d-core, and so is unique.

Finally, consider $k < d$. Suppose for a contradiction that there exists some $k < d$ such that the k-contour is not unique. Select j to be the largest such value, so the j-contour is not unique but the $j+1$-contour and higher contours are unique. Let S_1 and S_2, $S_1 \neq S_2$, be two sets that are both j-contours. Define $S = S_1 \cup S_2$. Since S_1 and S_2 satisfy Definition 3, $\forall u \in S$, u does not belong to a higher contour. Additionally, u must have at least j neighbors in either S_1 or S_2 (depending on which of these two sets it belongs to), and so it certainly has at least j neighbors in S. Thus, S satisfies conditions (1) and (2) of Definition 3 of a j-contour. However, because S is a proper superset of S_1 and S_2, both S_1 and S_2 violated the maximality requirement in Definition 3. This is a contradiction, so the k-contour must be unique. Thus, the k-peaks are unique for all integers $k > 0$. □

3.3 *k*-Mountains

Using the k-peaks, one can identify the k-mountains of the network, each of which can be thought of as a distinct regions within the larger network.

A k-mountain is defined as follows:

Definition 7. ***k-mountain:*** *A k-mountain is the induced subgraph of G containing all nodes $v \in G$ such that $C_v^k < C_v^{k+1}$, where C_v^k is the core number of node v after the nodes in the k-peak are removed (if a node itself is removed, define its core number after removal as 0).*

Intuitively, the k-mountain contains the k-contour as well as nodes whose core numbers are affected by its removal (after higher contours have been removed). Next, we prove that for every node in a k-mountain, there is a path from that node to the k-contour using only other nodes in the k-mountain. If the k-contour is a connected subgraph, then the k-mountain is also a connected subgraph. If the k-contour consists of multiple connected subgraphs, then the k-mountain may also contain multiple connected subgraphs.

The essence of the following proof is the observation that if edges are removed from a graph and the core number of some node changes as a result, then either that node was directly modified, or its core number changed as a result of a neighbor's core number changing.

Theorem 2. *Given a graph G, for every node u in a k-mountain of G, there is a path from u to a node in the k-contour of G that only uses other nodes in the k-mountain of G.*

Proof. The determination of whether a node belongs to the k-mountain depends only on its core number after the $k+1$-peak is removed vs. its core number after the k-peak is removed.

Define H to be the graph after the $k+1$-peak is removed. If G contains a k-contour, that means that the degeneracy of H must be equal to k. It thus suffices to consider only H, and show the theorem for the d-mountain of H, where d is the degeneracy of H.

Let H' be the resulting graph after nodes in the d-contour (degeneracy core) are removed from H. Define an *affected node* to be a node with a lower core number in H' than in H (i.e., a node in the d-mountain). Suppose for a contradiction that there is some affected node u, but there is no path from u to the d-contour using only other affected nodes.

Let S be the maximal connected (in H) set of affected nodes that contains u. Because there is no path along affected nodes from u to the d-contour, and S is connected in H, then there is no path from any node in S to the degeneracy core. S also clearly cannot contain any nodes in the d-contour itself.

Let T be the set of nodes that are adjacent to a node in S, but are themselves not in S. Because a node's core number cannot increase after the degeneracy core is removed, every node in T must have the same core number in both H and H', because otherwise S would not be maximal.

Suppose the core number of u in H is k_u, and define K be the k_u-core in H. Let $S_k = S \cap K$ and $T_k = T \cap K$. In H, each node in S_k had at least k_u connections to other nodes in $S_k \cup T_k$ (because it has at least k_u connections to K, and all of its neighbors are in either S or T). In H', the nodes in S_k still have at least k_u connections to $S_k \cup T_k$. This is because only edges within the degeneracy core were removed, and we have assumed that S (and thus also S_k) are disjoint from the degeneracy core. Thus, none of the edges incident to S were affected.

Let K' be the k_u-core in H'. $K' \subseteq K$ since $H' \subseteq H$. Note that none of the nodes in S_k can be part of K', because these are the nodes that had core number k_u in H and some lower core number in H'. We know that the core numbers of all nodes in T are unchanged. Hence the core number of the nodes in T_k is still k_u. Thus $T_k \subset K'$. But then $S_k \cup K'$ is a k_u-core. This is a contradiction, because this means that K' was not maximal, and thus was not the k_u-core of H'. Contradiction, so the claim is proved. □

4. COMPUTING THE *K*-PEAK DECOMPOSITION

The pseudocode for finding the k-peak decomposition of a graph G is presented in Section 4.1 and proof of its correctness and analysis of its performance is given in Section 4.2.

4.1 Algorithm

The k-peak decomposition algorithm is shown in Algorithm 1. This algorithm iteratively removes the highest k-core of the graph and computes the core number of the remaining nodes at each iteration. The peak number of a node is the degeneracy of the graph before the node was removed. This process is carried on until the graph is empty.

Algorithm 1 k-peak decomposition

```
Input: The original graph $G(V, E)$
Output: $P$ {Peak Numbers of nodes}
1: while $V \neq \varnothing$ do
2:     $d$-contour ← degeneracy core of $G$ {Via $k$-core decomposition of $G$}
3:     $\forall u \in d$-contour, $P_u \leftarrow d$
4:     $V \leftarrow V \setminus d$-contour {Removing $d$-contour from $G$}
5: end while
```

4.2 Theoretical Analysis

Lemma 1. *Given graph $G(V, E)$, Algorithm 1 finds the k-peak decomposition.*

Proof. To show the correctness of Algorithm 1, we will show that it correctly assigns peak numbers to each node.

First consider all nodes u with peak number equal to the degeneracy d of the input graph G. Each of these nodes u is in the d-core, which is also the d-shell. Since there is no k-contour for $k > d$, by Definition 4, Algorithm 1 correctly sets the peak numbers of these nodes.

Next consider nodes with peak number less than d. Suppose for a contradiction that for some peak number $k < d$, the algorithm does not work correctly (i.e., a node with true peak number k is assigned an incorrect peak number). Consider the largest such value k, so that the algorithm correctly assigns peak numbers to all nodes with peak number above k. Let G_k be the graph remaining after the k+1-peak (which by our assumption is correct) is removed. Let d_k be the degeneracy of graph G_k. There are two possibilities: $k > d_k$ or $k = d_k$ (if $k < d_k$, nodes in the degeneracy core of G_k would have been removed as part of the $k+1$-peak, so would not be in G_k).

Suppose $k > d_k$. Then no nodes are assigned peak number k (because the nodes in the degeneracy core of G_k are assigned peak number d_k.). If this is incorrect, then the true k-contour of G should be non-empty. By Definition 4, the k-contour is a k-core in G_k. This is a contradiction, because we assumed that the degeneracy of G_k is less than k.

Now suppose $k = d_k$. After removing the $k+1$-peak, the algorithm finds the degeneracy d_k-core of G_k, and these nodes are assigned a peak number of $d_k = k$. We assumed that this set is not equal to the true k-contour of G. All nodes in the true k-contour of G must be contained in G_k, because otherwise they would belong to a higher contour, which is assumed to be correct. The true k-contour of G is a maximal set of nodes in G_k that have at least k connections to one another; but this is simply the k-core of G_k. Thus, the d_k-core of G_k is equal to the k-contour of G. This proves the correctness of Algorithm 1. $\square$

The next two Lemmas discuss the running time and space requirements of Algorithm 1. Let N and M represent, respectively, the number of nodes and edges in G.

Lemma 2. *Algorithm 1 requires $O(\sqrt{N}(M + N))$ time.*

Proof. Suppose that a graph has b non-empty k-contours with peak numbers $k = \{k_1, ...k_b\}$, of sizes $s_1, ...s_b$. Suppose this list of contours is ordered in descending order, so k_1 is equal to the degeneracy of the graph. These k_i values must be unique, non-negative integers.

Because there are b non-empty contours, we must have $k_1 \geq b - 1$, so $s_1 \geq b$ (because each node in this contour has at least $b-1$ connections within the contour). Similarly, $k_i \geq b - i$ and $s_i \geq b - i$, for each i. Let N be the number of nodes in the graph. Then, $N = s_1 + s_2 + ... + s_b \geq b + (b - 1) + ... + 1 = (b)(b + 1)/2 \geq (b^2)/2$. Rewriting, we get $b \leq \sqrt{2N}$, so there are at most $\sqrt{2N}$ contours. Finding each contour requires performing a core decomposition on the graph, requiring $O(N + M)$ time. Thus, finding the k-peak decomposition requires $O(\sqrt{N}(M + N))$. $\square$

Lemma 3. *The algorithm 1 requires $O(M + N)$ space*

Proof. Each iteration of the while loop in Algorithm 1 computes a k-contour by computing the k-core decomposition of the current graph. Finding a k-core decomposition of G requires $O(M + N)$ space, which can be reused in subsequent iterations. The vector P requires $O(N)$ space. Thus, the space requirement at any iteration is at most $O(M + N)$. $\square$

Theorem 3. *Algorithm 1 returns correctly finds the k-peak decomposition of a given graph G in $O(\sqrt{N}(N + M))$ time using $O(M + N)$ space.*

Proof. From Lemmas 1, 2, 3, thus proved. $\square$

5. PROPOSED VISUALIZATION

By using k-peaks, one can gain a deeper understanding of the network structure. In this section, we present the *mountain plot*, a novel visualization technique that uses the k-peak decomposition to understand both the global and local structure of the network.

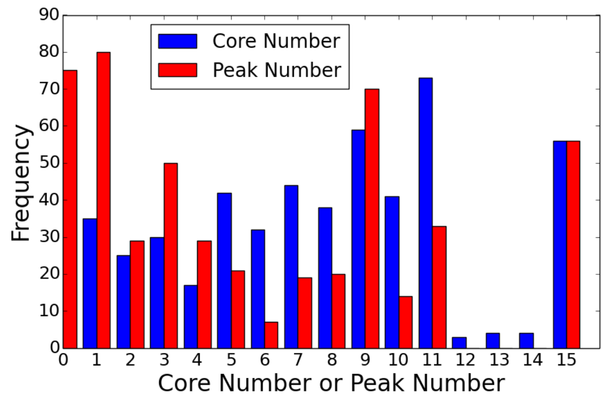
5.1 Mountain Plot

A major motivation behind the k-peak decomposition was understanding the dependencies between the core numbers of nodes. An obvious way to visualize these dependencies would be to plot histograms of the core numbers and peak numbers of the nodes in a graph G. An example of such a plot on the FB-Grad network is presented in Figure 3a. As expected, there are more nodes with higher core numbers than peak numbers. However, this simple plot suffers from two major problems: (1) It does not give us information on individual nodes, and (2) It fails to show the effect the removal of a k-contour on the rest of the graph.

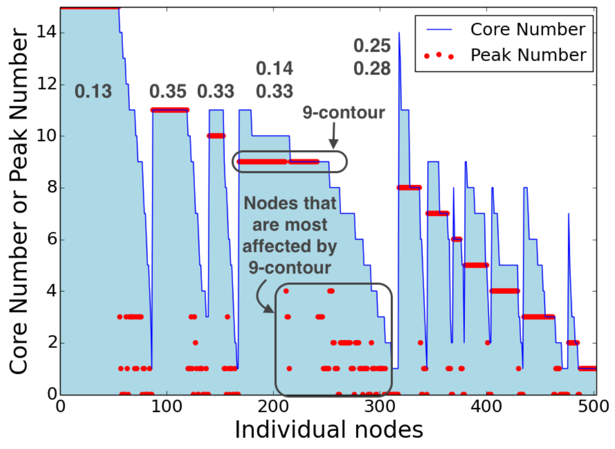
Our proposed visualization method, the *mountain plot*, provides a concise summary of the global peak structure of the network, depicts how the core and peak numbers of individual nodes differ, and allows for an understanding of node dependencies on k-contours.

To generate a mountain plot, we find mountains as in Section 3.3. Note that these mountains are overlapping. However, to keep the mountain plot concise, we associate each node with only one mountain.[2] We find k-contours as in Algorithm 1, where each contour forms the center of a mountain. We associate each node in the graph with the

[2]Note that we are not redefining the notion of a mountain, but are distinguishing between mountains that exist in the graph structure vs. mountains as they are plotted.



(a) Histogram of core and peak numbers, showing the difference in distribution of core and peak numbers.



(b) Mountain plot annotated with the core-periphery measure for each component in each mountain.

Figure 3: Facebook Grad network: The mountain plot gives more insight than the histogram. The mountains represent underlying distinct regions of the graph; e.g., the fourth mountain identified by the 9-contour, has a component with core-periphery structure (ρ=0.33, see Section 6.2) that is explained by the many nodes with low peak number (high dependency) in the mountain.

contour whose removal produced the greatest drop in its core number: these nodes constitute a plot-mountain. (Note that a node in the contour itself may not be assigned to that plot-mountain if it were more affected by an earlier plot-mountain.) The resulting plot-mountains are thus non-overlapping.

We sort these plot-mountains in descending order of the peak numbers of their associated contours. Within each mountain, we sort the member nodes in descending order of core number. For nodes with the same core number in a plot-mountain, we further sort them by their peak-number. This gives us an ordering of the nodes. On the x-axis of the mountain plot, we show the nodes in this order. Each integer along the x axis corresponds to the permuted ID of a node. We plot the core number of this node in blue, and the peak number in red. We then draw a line connecting all blue markers to outline the mountains.

For example, Figure 3b is a mountain plot of the FB-grad network. This plot shows us that the FB-grad network contains 12 mountains. The highest red markers within each mountain show the peak number of the contour corresponding to that mountain. The height of the mountain, in blue, shows the core numbers of the node within the mountain.

5.2 Interpreting a Mountain Plot

The mountain plot can give us the following information:

- The **number of mountains** in the mountain plot tells us the number of distinct regions in the graph
- The **mountains in the left of the plot** represent the most well-connected regions of the graph. The mountains towards the right corresponds to the sparser and more disconnected parts of the graph.
- The **height and width**, respectively, of a mountain increase with the core number and the number of nodes in the mountain. In the graphs that we observed, the nodes with the highest peak number in a mountain are those in the k-contour giving rise to the mountain.[3]
- The **difference between the core numbers and peak numbers** in a mountain gives information about the connectedness and dependencies of the subgraph represented by a mountain (See Section 6).

From the mountain plot, we can immediately get a sense of the global structure of the graph.

6. DESCRIPTIVE STUDY

We consider the 22 real graphs listed in Table 1, including social, co-authorship, email, web, and biological graphs. We begin in Section 6.1 by discussing structures observed in the mountain plots. In Section 6.2, we analyze the structures found in the detected mountains.

The mountains found by the k-peak decomposition sometimes exhibit core-periphery structure [4]; however, the mountain plot allows for substantially more nuance in the way that we understand these graph regions (i.e., by examining the shapes of the mountains), rather than simply labeling them as core-periphery structures. Depending on the network structure, the detected regions may *not* be core-periphery structures (e.g., as we will see, the CondMat network in Figure 4b with 'column' mountains). Additionally, although in practice the k-peak decomposition sometimes finds core-periphery-like structures, it is important to note that this is not its purpose: rather, the goal is to find distinct regions and understand patterns of node dependencies.

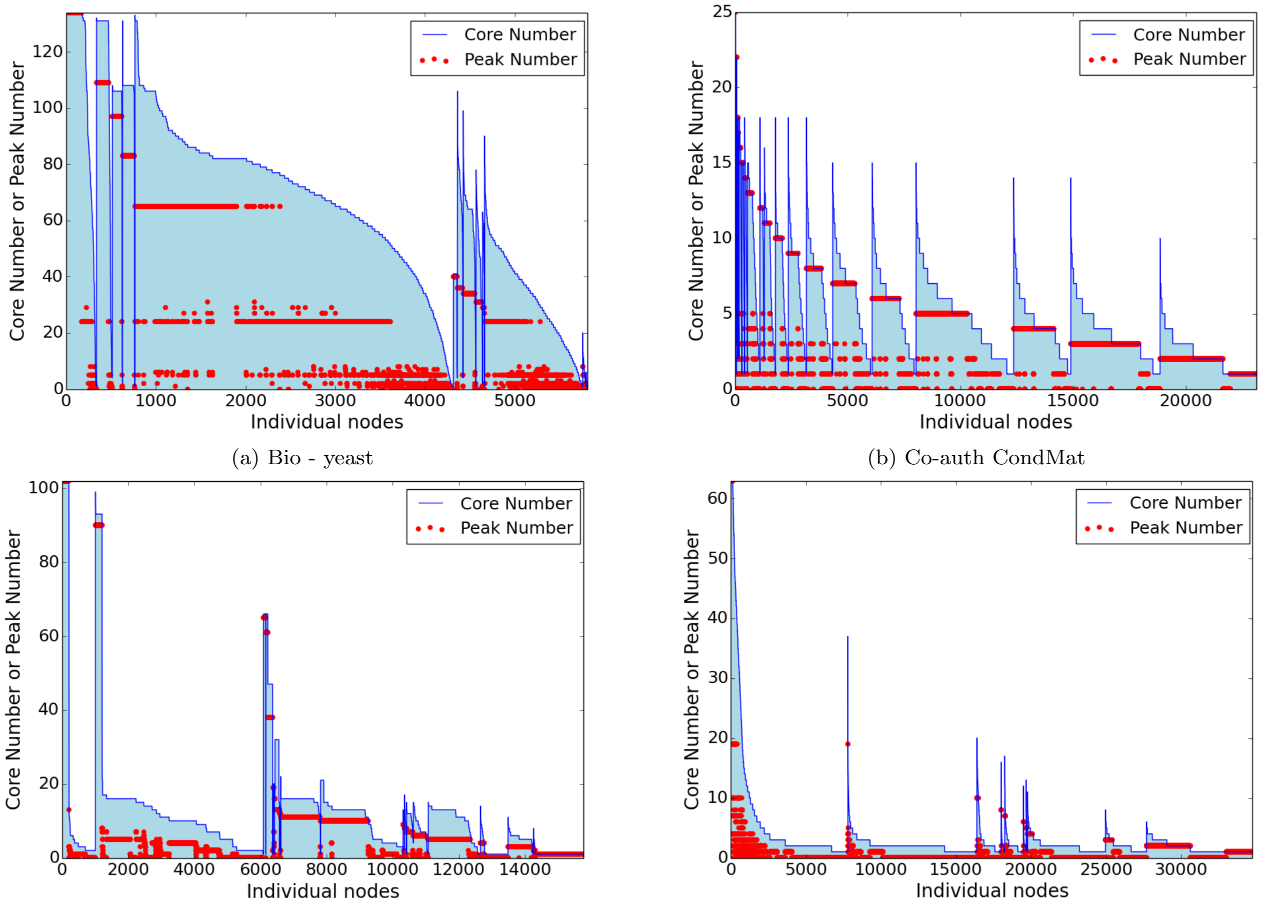
6.1 Mountain Plots of Real Graphs

In Figures 3b and 4, we present a mountain plot for representative social, biological, co-authorship, Internet and web graphs.

We make several observations from the mountain plots.

Number and Sizes of Mountains: First, the number of mountains identified by the k-peak decomposition corresponds to the number of distinct regions within the network, and the width of the mountains in the mountain plot shows the size of these regions. For example, Figure 4a shows us that the yeast biological network contains one large mountain, one medium-sized mountain, and several very small mountains. In contrast, Figure 4b shows

[3]Because we assign each node to the mountain that affected its core number most, it may be that the nodes in the k-contour giving rise to the mountain are not actually assigned to that mountain: however, in practice, this rarely happens, and mostly for the weaker, less well-connected mountains.



(a) Bio - yeast

(b) Co-auth CondMat

(c) Web - google

(d) Internet topology

Figure 4: Mountain plots for Protein3, CondMat, and Google Web and Internet topology graphs. Observe the number, shapes, and sizes of mountains (i.e. regions).

us that the CondMat co-authorship network contains many medium-sized mountains; but these mountains appear on the right side of the plot, indicating that they are not associated with the highest k-contours. These mountains, while large, are not as well-connected as the smaller mountains on the left side of the plot.

Many of the social graphs we consider tend to have a small number of meaningful mountains. In contrast, the co-authorship graphs all had the same basic pattern as the CondMat graph in Fig. 4b: small, well-connected mountains on the left side, followed by weaker, larger mountains.

Shapes of Mountains: We observe two extreme types of mountains, with gradations in between: the 'column' type mountains (e.g., at the beginning of Figures 4a, and 4b), and the 'wide' mountains (e.g., the largest mountain in Figures 4a and 4c). A column mountain is typically formed of a large clique on which very few other nodes depend for their core numbers. In contrast, the wide mountains are the well-formed core-periphery structures, with a small core and a large, highly-dependent periphery.

Co-authorship graphs, such as in Figure 4b, tend to have column mountains. These mountains likely correspond to papers with many authors who tend to work together most of the time, thus forming a clique that is not well-connected to the rest of the network. These narrow column mountains very rarely exist in social graphs. Web graphs and Internet graphs as shown in Figure 4c and 4d, contain several mountains that begin with a column, but suddenly expand into a wide base at the bottom (e.g., the first two mountains in 4d). These represent tightly-connected structures on which many low-core-number nodes depend: these mountains likely represent cliques with low-degree nodes on the periphery, as opposed to a gradually expanding core-periphery structure like we see in Figure 4a.

Social networks, as in Figure 3b, tend to have one or two large, meaningful core-periphery structures, and a collection of mountains that are closer to column-like. This latter type of mountain (e.g., the first mountain in Figure 3b represents a tightly-knit group of individuals with a small number of additional high-core-number individuals (contrast this with the Web graph mountains, which are tightly-knit groups with a large number of additional low-core-number nodes).

By looking at the mountain plot, we get a sense of the overall connectedness of the network as a whole: does it contain distinct regions or is it one cohesive structure?

Difference between a node's core and peak number: For each node, the mountain plot tells us the difference between that node's core number and peak number. This difference is simply the distance between the appropriate blue line and red points. This difference tells us the extent to which a node's core number is dependent on higher k-contours: the larger the drop, the more dependent the node

	Graph	$\lvert V\rvert$	$\lvert E\rvert$	d	ρ_G	ρ_5	ρ_{10}
Bio	Protein1[1]	1,702	3,155	7	0.06	0.15	0.15
	Protein2[1]	2,239	6,432	10	0.03	0.20	0.20
	Protein3[10]	5,808	362,421	134	0.09	0.22	0.17
Co-auth	GrQC [12]	5,241	14,484	43	0.03	0.24	0.25
	HepTH [12]	9,875	25,973	31	0.02	0.25	0.24
	HepPH [12]	12,006	118,489	238	0.07	0.29	0.28
	AstroPh [12]	17,903	196,972	56	0.03	0.29	0.29
	CondMat [12]	23,133	93,439	25	0.02	0.24	0.23
Msg	Digg [1]	30,360	85,155	9	0.03	0.07	0.07
	Enron [12]	33,696	180,811	43	0.06	0.16	0.24
Social	FB Grad [14]	503	3,256	15	0.05	0.26	0.24
	FB Caltech[14]	769	16,656	35	0.14	0.05	0.05
	FB Reed [14]	962	18,812	34	0.14	0.10	0.08
	FB Ugrad [14]	1,220	43,208	47	0.09	0.05	0.03
	FB Baylor [14]	12,803	679,817	96	0.05	0.12	0.10
	Brightkite [12]	58,228	214,078	52	0.04	0.27	0.28
Tech	Gnutella [12]	26,498	65,359	5	0.01	0.19	0.19
	Internet [1]	34,761	107,720	63	0.08	0.18	0.26
Web	Foldoc [1]	13,356	91,471	12	0.02	0.02	0.06
	Google [1]	15,763	148,585	102	0.06	0.13	0.12
	Stanford [12]	281,903	2,312,497	71	0.04	0.28	0.40

Table 1: Statistics of the real graphs, varying in size and type. Here d is the degeneracy of G and ρ_G, ρ_5 and ρ_{10} are the core-periphery measures for G, average of the first 5 and 10 mountains respectively.

is. For example, in the first mountain in Figure 3b, we see nodes with core numbers 9-13, whose peak numbers dropped to 3 and 2, due to the removal of the degeneracy core of the graph. On the other hand, in the fourth mountain, while there are nodes with low peak number, these nodes tended to have a low core number to start with: thus, this mountain contains fewer high-core-number nodes that are highly dependent on the mountain's k-contour. For another example across graphs, compare the core and peak numbers of nodes in the the first mountain in Fig 4d (Internet topology graph) with that of the second mountain in Fig 4c (web-graph). In this example, although both mountains have a small tightly-knit core, the periphery (nodes in the mountain excluding the nodes in the k-contour) of the web-graph is more dependent on the k-contour for its core number than that in the Internet topology graph.

6.2 Core-Periphery Structure in a Mountain

Although finding core-periphery structures is not our goal, it is interesting to measure how well the detected regions match the ideal core-periphery structure. According to Borgatti and Everett's formulation, the core-periphery structure is represented by an idealized adjacency matrix in which the first r nodes form the core and the remaining s nodes form the periphery ($r + s = N$). Every node in the core is connected to every node in the graph, but nodes in the periphery are only connected to nodes in the core. To measure how well a graph exhibits core-periphery structure, one reorders the adjacency matrix so that the core nodes appear first, and then measures ρ, the Pearson's r correlation between the true adjacency matrix and the idealized adjacency matrix (in our mountains, the 'core' contains nodes in the associated contour, while the 'periphery' contains everything else). $\rho \in [-1, 1]$, with values above 0 indicating positive core-periphery structure.

Because the k-peak decomposition is premised on the notion that sparser graph regions can exhibit meaningful structure, we modify this measure by dividing the values in the matrix by $2M$, so that the sum of (weighted) edges in the idealized and true adjacency matrices are the same. We measure significance using a QAP test, by repeatedly randomly permuting the set of nodes in the core and periphery (while maintaining their sizes), and calculating the correlation between the permuted adjacency matrix and the idealized adjacency matrix: p is simply the fraction of times that the permuted adjacency matrix exhibits a greater or equal correlation than the true adjacency matrix.

The first five mountains in Figure 3b are annotated with their core-periphery ρ values. Note that the fourth and fifth mountains consist of two connected components, so two values are shown. While these values seem low as correlations, the overall ρ of the entire graph is only 0.05,[4] and so the mountains exhibit much greater core-periphery structure. These values are significant at the $p = 0.01$ level.

The average ρ values for the first 5 and first 10 mountains is given in Table 1 for all graphs. Note that the values vary substantially by network: although the k-mountains are almost always more core-periphery-like than the graph as a whole, some mountains exhibit strong core-periphery structure (e.g., the first 10 mountains of the Stanford graph), while others are much weaker (e.g., Google web in Fig. 4c).

7. APPLICATIONS: *K*-PEAK VS. *K*-CORE

We now show how the k-peak decomposition can replace or supplement the k-core decomposition in a variety of applications: community detection, contagion, and finding essential proteins in a protein-protein interaction network. Peng, et al. [17] present a three-step method using k-cores to accelerate community detection that finds communities in a k-core with $\sim 30\%$ of the nodes, and uses these community labels to infer community memberships of nodes outside the k-core. The key intuition behind this process is that the community memberships of nodes in the k-core are useful in identifying the community memberships of nodes outside of the core.

7.1 Accelerating Community Detection

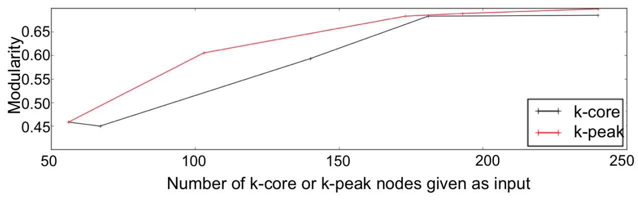
This intuition is not justified in a graph with multiple distinct regions. We propose that instead of using the k-core, one should use the k-peak. To motivate this argument, consider the example Facebook network from Section 1: a high k-core captures the Icelandic community structure but not the Eritrean structure, and lower values of k capture too much of the Icelandic region, negating gains in speed.

To illustrate this, we show results for the Facebook Grad and Facebook Baylor networks in Fig. 5 (similar results observed in other cases).

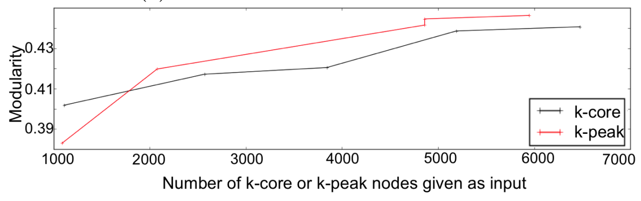
We find k-cores and k-peaks containing 10% - 50% of the nodes in the graph, run the algorithm from [17] on these cores and peaks, and measure the modularity of the detected partitionings. When using the k-peak, the algorithm finds a much better partitioning than when using the k-core.

This application captures the essence of our motivation for the k-peak: when the graph contains distinct, independent

[4]To measure the ρ of an entire graph, we consider all values of k, and set nodes in the k-core to be the 'core' and nodes outside to be the periphery.



(a) Facebook Grad network



(b) Facebook Baylor network

Figure 5: Community detection using k-cores and k-peaks.

regions of different densities, the k-core is less suitable than the k-peak, which finds the cores of these different regions.

7.2 Locating Essential Proteins

Biological researchers are interested in experimentally identifying 'essential' proteins in protein-protein interaction (PPI) networks. An essential protein is typically defined as one that is required for the survival or reproduction of an organism, as opposed to one that simply increases fitness [25]. Proteins with higher core numbers in this network are more likely to be essential [23]. Here, we show how peak number and core number can be used together.

First, we observe that peak number should not be used as a replacement for core number in this application. Suppose, for instance, that a PPI network has degeneracy d. Consider a node that is tightly connected to this core, but only has core number of $d - 1$. This node is still part of this central, well-connected region, and so is likely to be essential, but has a very low peak number. If we considered peak number alone, we would overlook nodes like this. However, if one considers only core number, essential nodes with low core numbers are excluded. We argue that to better identify these nodes, peak number is useful.

We consider the Protein3 graph listed in Table 1, a Yeast PPI network. Over the entire network, essential nodes are 18% of the entire set of nodes, and core numbers range from 1 to 134. Suppose we define a 'low' core number to be anything less than or equal to some value t. Identifying essential low-core-number nodes is necessary to get a complete picture of essential proteins. For each low core number $c \leq t$, let R_c be the set of nodes with highest peak number within the set of all nodes with core number c. Let R represent the union of R_c subsets over all low core numbers c.

At $t = 30$, R contains 12% of the low-core-number nodes, but 34% of the essential nodes with low core numbers. Similarly, at $t = 10$, R consists of 8% of the nodes with low core number, but 22% of the essential nodes; and if $t = 20$, R consists of 12% of the nodes, but 28% of the essential nodes. In other words, by finding nodes with high peak numbers within a given k-shell, one can improve accuracy.

Effectively, this experiment is identifying nodes that have low core numbers, but are central within their regions: if two nodes have the same core number, but one exists in the periphery of a mountain and the other is in the core of a mountain, the second is much more likely to be essential.

	Infection Probability (b)					
p	0.01	0.02	0.05	0.1	0.2	0.5
0.01	0.06 (0.2)	0.04 (0.12)	0.0 (0.03)	0.0 (0.01)	0.0 (0.01)	0.0 (0.0)
0.05	0.12 (0.17)	0.09 (0.14)	0.07 (0.1)	0.04 (0.05)	0.01 (0.02)	0.0 (0.0)
0.1	0.17 (0.28)	0.17 (0.25)	0.11 (0.15)	0.08 (0.09)	0.04 (0.05)	0.01 (0.01)
0.5	0.55 (0.59)	0.53 (0.55)	0.46 (0.47)	0.38 (0.37)	0.27 (0.27)	0.13 (0.14)

Table 2: Average fraction improvement (std. deviation) in number of infected nodes when initially infected nodes p are chosen according to peak number vs. core number.

7.3 Identifying Influential Nodes

It is well known that nodes with high core number tend to be more important in spreading disease or information through a network. Kitsak, et al. [11] show that in an SIR contagion, the core number of the initially infected node is better correlated with the final number of infected nodes than is the degree of the node.

We show that if multiple nodes are initially infected, selecting nodes with high peak number leads to a larger set of infected nodes than infecting nodes with high core number. We perform simulations as follows: Given a network G, we initially infect p fraction of top core number or top peak number nodes in G, with infection probability b using a SIR contagion. As in [11], we set the lifetime of the infection to be 1 unit of time, and run the simulation until the infection dies out. For every network listed in Table 1, for a range of values of p, and b, we perform 100 simulations.

Table 2 shows the average fraction improvement obtained by using peak numbers instead of core numbers. For example, a value of 0.5 would indicate that the contagion beginning at nodes with high peak numbers infected 50% more nodes than the contagion beginning at nodes with high core numbers. The standard deviation is sometimes high, but this variation typically occurs on the upper end: that is, using peak number almost always outperforms using core number, but the degree of improvement varies.

In almost all cases, values are positive, indicating that beginning the contagion at nodes with high peak numbers infects a greater number of nodes than beginning the contagion at nodes with high core numbers.

8. CONCLUSION

We have proposed the novel k-peak graph decomposition. We show how the k-peak decomposition can be used to better understand the global structure of large networks, and present an algorithm to efficiently find the decomposition. Additionally, we introduce a new network visualization plot, the mountain plot, which depicts the separate regions of the network. We apply the k-peak decomposition to a variety of real-world networks, and show how it gives substantially deeper insight than the k-core decomposition alone. Finally, we show how the k-peak decomposition can replace the k-core decomposition in several applications, with large gains in performance.

Acknowledgments

We would like to thank Chengbin Peng for sharing his community detection code [17]. This work was supported by NSF IIS 1447793 and NSF CCF 1535878.

9. REFERENCES

[1] KONECT: The koblenz network collection. `http://konect.uni-koblenz.de/networks`, May 2015.

[2] J. Abello and F. Queyroi. Fixed points of graph peeling. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*, pages 256–263. IEEE, 2013.

[3] V. Batagelj and M. Zaversnik. An O(m) algorithm for cores decomposition of networks. *Advances in Data Analysis and Classification*, 5(2):129–145, 2011.

[4] S. P. Borgatti and M. G. Everett. *Social Networks*, pages 375–395, 2000.

[5] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. 2008.

[6] S. A. Erdem, C. Seshadhri, A. Pinar, and U. V. Catalyurek. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *WWW*, 2015.

[7] P. Govindan, S. Soundarajan, T. Eliassi-Rad, and C. Faloutsos. Nimblecore: A space-efficient external memory algorithm for estimating core numbers. In *ASONAM*, 2016.

[8] P. Holme. Core-periphery organization of complex networks. *Physical Review E*, 72(4):046111, 2005.

[9] J. Jiang, M. Mitzenmacher, and J. Thaler. Parallel peeling algorithms. *ACM Trans. Parallel Comput.*, 3(1):7:1–7:27, Aug. 2016.

[10] H. Kim, J. Shin, E. Kim, H. Kim, S. Hwang, J. E. Shim, and I. Lee. Yeastnet v3: a public database of data-specific and integrated functional gene networks for saccharomyces cerevisiae. *Nucleic Acids Research*, 42:731–736, 2014.

[11] M. Kitsak, L. Gallos, S. Havlin, F. Liljeros, L. Muchnik, E. Stanley, and H. Makse. Identification of influential spreaders in complex networks. *Nature Physics*, 6(11):888–893, 2010.

[12] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[13] Y. Liu, N. Shah, and D. Koutra. An empirical comparison of the summarization power of graph clustering methods. *CoRR*, 2015.

[14] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In *WSDM*, pages 251–260, 2010.

[15] A. Montresor, F. D. Pellegrini, and D. Miorandi. Distributed k-core decomposition. *IEEE TPDS*, 24(2):288–300, 2013.

[16] M. P. O'Brien and B. D. Sullivan. Locally estimating core numbers. In *ICDM*, pages 460–469, 2014.

[17] C. Peng, T. G. Kolda, and A. Pinar. Accelerating community detection by using k-core subgraphs. *CoRR*, abs/1403.2226, 2014.

[18] M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha. Core-periphery structure in networks. *SIAM Journal on Applied Mathematics*, pages 167–190, 2014.

[19] A. E. Saríyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and U. V. Çatalyürek. Streaming algorithms for k-core decomposition. *PVLDB*, 6(6):433–444, 2013.

[20] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

[21] J. Ugander, B. Karrer, L. Backstrom, and J. M. Kleinberg. Graph cluster randomization: Network exposure to multiple universes. In *KDD*, pages 329–337, 2013.

[22] N. Wang, S. Parthasarathy, K. Tan, and A. K. H. Tung. CSV: visualizing and mining cohesive subgraphs. In *SIGMOD*, pages 445–458, 2008.

[23] S. Wuchty and E. Almaas. Peeling the yeast protein network. *Proteomics*, 5(2):444–449, 2005.

[24] X. Zhang, T. Martin, and M. E. Newman. Identification of core-periphery structure in networks. *Physical Review E*, 91(3):032803, 2015.

[25] X. Zhang, J. Xu, and W. xin Xiao. A new method for the discovery of essential proteins. *PLoS One*, 8(3), 2013.