On Finding and Analyzing the Backbone of the k-Core Structure of a Graph

Ricky Laishram
Syracuse University
rlaishra@syr.edu

Sucheta Soundarajan
Syracuse University
susounda@syr.edu

2022 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-5099-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDM54844.2022.00124

***Abstract*—In many network applications, dense subgraphs have proven to be extremely useful. One particular type of dense subgraph known as the k-core has received a great deal of attention. k-cores have been used in a number of important applications, including identifying important nodes, speeding up community detection, network visualization, and others. However, little work has investigated the 'skeletal' structure of the k-core, and the effect of such structures on the properties of the overall k-core and network itself. In this paper, we propose the *Skeletal Core Subgraph*, which describes the backbone of the k-core structure of a graph. We show how to categorize graphs based on their skeletal cores, and demonstrate how to efficiently decompose a given graph into its Skeletal Core Subgraph. We show both theoretically and experimentally the relationship between the Skeletal Core Subgraph and properties of the graph, including its core resilience.**

***Index Terms*—graph, k-core, structure**

I. Introduction

Over the last decade, k-cores have been used in a number of important applications, ranging from network visualization [1]–[3] to community detection [4]. The k-core of a graph is the maximal subgraph such that all nodes in the subgraph have at least degree k in the subgraph [5]; the highest k such that a node u belongs to the k-core is the core number of node u; and the set of all nodes with a given coreness k is the k-shell.

A key question in the study of k-cores is quantifying the extent to which the hierarchy-based properties of a graph's k-core decomposition are due to connections within shells vs. between shells. To understand this organization, we propose the concept of a *Core-Valid Subgraph (CVS)*. A CVS of a graph G is a spanning subgraph such that the core numbers of all nodes in the subgraph are the same as their core numbers in the original graph. One can further define a *Skeletal Core Subgraph (SCS)*, which is a *minimal* CVS– one for which removal of even one edge will reduce the core number of at least one node. Intuitively, one can think of a SCS as a *skeleton* of the k-core structure.

We first provide a fast greedy algorithm to generate an SCS. Next, based on the SCS, we propose the *Core Centralized Score (CCS)* for graphs, which measures how centralized the graph's hierarchical structure is, and can be used to visualize a graph. We analyze minimal SCSs on 33 real-world networks from a variety of domains. Among other things, we observe that networks from different domains tend to have different centralization levels. Finally, we present an application of SCSs to the task of community detection.

Soundarajan was supported in part by NSF Award 1908048.

II. Motivation

There are a number of reasons why one might wish to find a skeletal core subgraph (SCS).

First, like any 'backbone'-type structure, SCS is a minimal structure that retains key properties of the network. The SCS, of course, retains a network's core structure, but as we show in Sections VIII, by using a node scoring function related to a graph's SCSs, one can identify a set of nodes that captures key non-k-core aspects of a graph's structure (community structure, in this case). Such structures can be used when, for instance, space or processing time is an issue.

Second, the SCS structure of a graph gives overall insight into the organization of its k-core. It is well-known that different types of graphs have different k-core structures, but by using the SCS, we provide a single scoring function that quantifies the degree to which a graph's k-core organization is centralized (lower shells depend on higher shells) or decentralized (more independence between shells). Because an important application of k-cores generally is understanding the resilience of a network to cascading deletions (e.g., as in [6]), having a single score for such properties is useful.

Third, by finding a SCS or set of SCSs, one gains a better understanding of which edges are most important to the k-core structure of the network, and which are redundant. This is useful for an edge-focused analogue to the anchored k-core problem, which seeks to identify which nodes are most important to keep in the k-core to protect it from cascading failures [7]: if it is known that certain edges are present in many SCSs, then preserving those edges is of special importance.

III. Related Works

Seidman [5] and Matula and Beck [8] defined the k-core subgraph as the maximal connected subgraph where each vertex has at least degree k. Matula and Beck showed how to find the core number of each vertex, and for finding the k-core hierarchy of the graph [8]. The k-core decomposition has found application in many important domains, and gives insight into the hierarchical organization of the graph. Applications include network visualization [9], studying the topology of large networks (such as the Internet) [10], and

978-1-6654-5099-7/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDM54844.2022.00124

Notation	Description
$G_k = \langle V_k, E_k \rangle$	The k-shell subgraph.
$E_{i,j}$	The edges between the i-shell and the j-shell.
$\kappa(v, G), \kappa(v)$	The core number of node v in graph G.
$k^*(G), V_{k^*}$	The degeneracy and corresponding k-core of the graph G (highest value of k for which there is a non-empty k-core).
$\tilde{\Gamma}(v, G)$	The neighbors of v in graph G in $\kappa(v, G) - core$.
$CS(u)$	The core strength of u

TABLE I: Notations used in this paper.

accelerating community detection [4]. k-cores have also been used across many scientific fields, such as for explaining jamming transitions in particles [11] and for predicting the structural collapse of ecosystems [12].

The literature contains various other concepts of network backbones, including types of spanning trees have been proposed. The Minimum Spanning Tree has found application as a way to study the backbone of a transportation network [13]. In [14], the authors proposed the Maximum Spanning Tree and showed that it can be used to extract the backbone of a city network.

IV. Definitions

Our goal in this work is to gain insight into the k-core structure of a graph by understanding the relationships between its k-shells (each containing nodes with core number equal to k). As we will show, these relationships can give significant insights into the behavior of the graph across various tasks.

To understand the k-core structure of a graph, we propose the notion of a *Skeletal Core Subgraph (SCS)* of a graph G. A SCS is a minimal spanning subgraph in which all nodes have the same core number as in G, and can be interpreted as a k-core 'skeleton' of the graph. By examining the structure of a SCS, one can identify the type of hierarchical organization displayed by the graph. Then, using the SCS, we propose a metric to characterize how centralized or decentralized a particular network is. For the rest of our discussion, we will use the notations described in Table I. In cases where the graph is clear from the context, we will drop G for brevity.

Definition IV.1. Given a graph $G = \langle V, E \rangle$, a *Core Valid Subgraph (CVS)* of G is a spanning subgraph of G (i.e., includes every node from G) in which every node has the same core number as in G.

Definition IV.2. A *skeletal core subgraph (SCS)* $\sigma(G) = \langle V, \sigma(E) \rangle$ of G is a minimal CVS of G– i.e., one in which removal of even a single edge will change the core number of at least one node.

Note that G is a CVS of itself, and there may be multiple CVSs and SCSs of a given graph.

V. Finding Skeletal Core Subgraphs

Here, we first provide relevant background on the types of modifications that one can make to a node while preserving its core number. Next, we describe a fast greedy heuristic to find a single minimal SCS. Finally, we describe the Core Centralized Score, a metric for describing how centralized a network's k-core structure is.[1]

A. Background

The **core strength** of a node u in graph G was previously defined as the minimum number s for which there exist s of neighbors of u, such that disconnection of those neighbors from u would cause the core number of u to decrease by at least 1 [6]. $CS(u)$ is computed as $d_\kappa(u) - \kappa(u) + 1$, where $d_\kappa(u)$ is defined as the number of connections that u has to nodes of equal or higher core number.

Note that because of how core numbers are defined, it is possible for an edge removal to cause a cascading drop in core numbers across a chain of nodes. That is not accounted for in this definition of core strength; however, because our goal is to use core strength to determine which edges can be safely removed while preserving core numbers, no node's core number should drop with an edge removal, and so such cascades are impossible. We will use $CS(u, G)$ (or $CS(u)$, when clear from context) to denote the core strength of u in graph G in the rest of the discussion.

The core strength of all nodes in a graph can be efficiently calculated by performing a k-core decomposition, and then, for each node u, counting the number of neighbors with equal or greater core number than itself. The time complexity is $\mathcal{O}(|E|)$.

Theorem V.1. *Given a graph G with core numbers defined by $\kappa(u, G)$, the removal of edge (u, v) will affect the core numbers of G if and only if either: (1) $CS(u) = 1$ and $\kappa(v) \geq \kappa(u)$ or (2) $CS(v) = 1$ and $\kappa(u) \geq \kappa(v)$.*

Proof. In this discussion, let $\kappa(u)$ represent the core number of u before removal of edge (u, v).

First, we will show that if one of the two conditions is met, then the core numbers of G will change. WLOG, suppose condition (1) is met. If $CS(u) = 1$, that means that if u loses one edge to a neighbor with equal or higher core number, u will have fewer than $\kappa(u)$ connections to neighbors with core number greater than or equal to $\kappa(u)$, and so its core number will drop.

Next, suppose that the core numbers of G change after removal of edge (u, v). This must mean that the core numbers of u and/or v have also changed. That is, it is not possible for neither u and v to change core number but another node to change when edges (u, v) is deleted. WLOG, suppose node u changed core number. It is clear that removal of an edge cannot increase a node's core number; thus, u's core number must have dropped on removal of edge (u, v). If u's core number is now less than $\kappa(u)$, it must have fewer than $\kappa(u)$ connections to nodes of core number $\geq \kappa(u)$. However, because it previously had core number $\kappa(u)$, it had at least $\kappa(u)$ such connections. Thus, either (a) node v has core number $\geq \kappa(u)$ and u had exactly $\kappa(u)$ connections to nodes

[1]Code is available at https://www.dropbox.com/sh/cg6ghtqlhhleeuw/AABtN7DtDEglw3v_H7YOZM0aa?dl=0.

with core number $\geq \kappa(u)$: i.e., condition (1) in the theorem statement is met, or (b) node v had core number $\geq \kappa(u)$ originally, removal of edge (u, v) dropped node v's core number to below $\kappa(u)$, and u had exactly $\kappa(u)$ connections to nodes with core number $\geq \kappa(u)$: i.e., conditions (1) and (2) in the theorem statement are both met. □ □

Corollary V.1. *$\sigma(G) = \langle V, \sigma(E)\rangle$ is a minimal skeletal core subgraph of G if and only if there exists no edge (u, v) such that $CS(u, \sigma(G)) > 1$ and $CS(v, \sigma(G)) > 1$, where $CS(u, \sigma(G))$ is the core strength of node u in $\sigma(G)$ [6].*

B. Finding a Skeletal Core Subgraph: A Greedy Algorithm

In this section, we present a fast randomized greedy algorithm for finding a SCS. This algorithm relies on Theorem V.1, which provides us a way to efficiently check whether a given graph is a SCS: after performing the k-core decomposition, calculate the core strength of all the nodes, and check if there are any edge where both endpoints have core strength greater than 1.

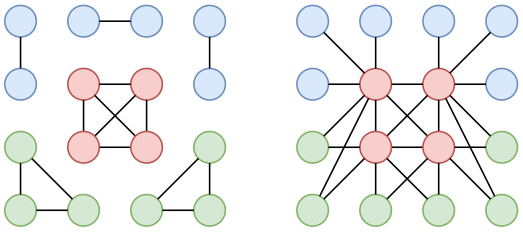
Algorithm: In this section, we describe `GreedySCS`, a randomized greedy algorithm for finding a SCS. The algorithm repeatedly deletes edges from the graph, using the result from Theorem V.1 to check that the subgraph's core numbers are correct. At each step, all the edges that connect nodes with core strength greater than 1 are candidates for deletion. A random edge from these candidates is selected and removed from the graph. Then, the core strengths are recomputed and the candidate sets are generated again. This continues until no more candidate edges remain.

Algorithm 1 Greedy algorithm to reduce a graph to a minimal skeletal core subgraph.

```
1: function GreedySCS(G = ⟨V, E⟩)
2:     R ← ∅
3:     repeat
4:         CS ← CoreStrength(G)
5:         R ← {(u, v) ∈ E : (CS[u] > 1 ∧ CS[v] > 1) ∨
       (CS[u] > 1 ∧ CS[v] = 1 ∧ κ(u) < κ(v))}
6:         X ← Random element of R
7:         E ← E \ {X}
8:     until R = ∅
9:     return G
```

Theorem V.2 (Complexity of Algorithm 1)**.** *The time and space complexity of Algorithm 1 are both $\mathcal{O}(|E|)$.*

Proof. Computing the core strength of all the nodes for the first time can be done in $\mathcal{O}(|E|)$. For the subsequent steps, instead of recomputing it, we can simply calculate it for only those nodes involved in an edge deletion since we have the guarantee that the core number does not change due to the edge deletion. (This follows directly from the definition of core strength. If both endpoints have core strength greater than 1, both have 'extra' edges that can be deleted without affecting the core numbers.)



(1) Decentralized Skeletal Core Subgraph (2) Centralized Skeletal Core Subgraph

Fig. 1: Toy example showing Decentralized (Figure 11 and Centralized (Figure 12) SCSs. The red, green and blue nodes have core numbers of 3, 2 and 1 respectively. In the left graph, all the nodes connects to a node in the degeneracy core (red node). In the right graph, all the nodes are connected to another one with the same core number.

Then, updating the core strength of a node can be done in constant time with a proper data structure. The loop in Algorithm 1 repeats at most $|E|$ times, and each edge deletion results in update of the core strength of two nodes. Inside each loop, the sets R and E can be found quickly through proper pruning.

So, the overall running time of Algorithm 1 is $\mathcal{O}(|E|)$.

The space required to store the graph is $\mathcal{O}(|E|)$, the core strengths of all the nodes can be stored in $\mathcal{O}(|V|)$, and that for R is $\mathcal{O}(|E|)$.

So, the overall space complexity of Algorithm 1 is $\mathcal{O}(|E|)$. □ □

VI. Measuring Hierarchy: the Core Centralized Score

To better understand the effects of different type of changes to the core structure of a SCS, we need to categorize them into different types. We start by categorizing the edges based on the core numbers of its endpoints:

1) **Inter-Shell Edges:** These are the edges whose end vertices have the same core number.
2) **Intra-Shell Edges:** These are the edges whose end vertices have different core numbers.

Depending on the number of inter and intra-shell edges, we have two extreme cases of skeletal core graphs. We refer to these two extremes as centralized and decentralized skeletal core graphs.

1) **Decentralized Skeletal Core Subgraph**: These are the skeletal core graphs with no inter-shell edges.
2) **Centralized Skeletal Core Subgraph**: These are the skeletal core graphs with: (a) no intra-shell edges, except in the degeneracy core, and (b) all the inter-shell edges have one endpoint in the degeneracy core.

As an example, consider the graphs shown in Figure 1. The color of the nodes indicates their core number – red is 3, green is 2 and blue is 1. Both graphs are SCSs. In Figure 2.1 all the nodes connect only to others with the same core number,

making this an example of a decentralized SCS. In Figure 2.2, all nodes connect to a node in the degeneracy core (red nodes), making this an example of a centralized skeletal core graph.

In the rest of the discussion, we will use $\sigma_D(G) = \langle V, \sigma_D(E)\rangle$ and $\sigma_C(G) = \langle V, \sigma_C(E)\rangle$ to denote decentralized skeletal and centralized core graphs respectively.

A. Core Centralized Score

For any graph, depending on the number of inter and intra-shell edges, its minimal SCSs will fall between the extreme decentralized and centralized structures. To quantify where a graph falls within this range, we propose the *Core Centralized Score* measure.

The basic idea behind the Core Centralized Score is that for a node u, the closer its neighbors in the $\kappa(u)$-core are to the degeneracy core, the more central the node is u. So, for a SCS $\sigma(G) = \langle V, \sigma(E)\rangle$, we define the Core Centralized Score as,

$$CE\left(\sigma(G)\right) = \frac{1}{|\overline{V}|}\sum_{v\in\overline{V}}\frac{1}{|\tilde{\Gamma}(v)|}\sum_{u\in\tilde{\Gamma}(v)}\frac{\kappa(u)-\kappa(v)}{k^*-\kappa(v)}, \quad (1)$$

where $\overline{V} = V \setminus V_{k^*}$. (Refer to Table I for notation.)

Higher values of this score indicate that a graph's SCSs are closer to a centralized SCS, and lower values indicates that its SCSs are closer to a de-centralized SCS. Decentralized graphs have a centralized score of 0, and centralized graphs have a score of 1.

Core Centralized Score for General Graphs: To quantify how far a graph is from a centralized or decentralized skeletal core, we extend the concept of Core Centralized Score to a general graph. *The core centralized score of a general graph is defined as the average core centralized score over all of its SCSs.*

Recall that a graph may have many SCSs. Thus, to compute this value, we compute the likelihood of each edge remaining in a SCS. For a graph $G = \langle V, E\rangle$, the likelihood of an edge (u, v) remaining in a SCS is dependent on the core number and number of neighbors in the same core of the node with lower core number (both nodes if they have the same core number). That is,

$$p(u,v) = \begin{cases} \frac{\kappa(u)}{e(u)}\frac{\kappa(v)}{e(v)} & \text{if } \kappa(u) = \kappa(v) \\ \frac{\kappa(u)}{e(u)} & \text{if } \kappa(u) < \kappa(v) \\ \frac{\kappa(v)}{e(v)} & \text{if } \kappa(u) > \kappa(v) \end{cases} \quad (2)$$

where,

$$e(u) = |\tilde{\Gamma}(u, G)|. \quad (3)$$

For edge (u, v), $p((u, v), G')$ gives us a measure of how likely the edge is to be in the skeletal core. If $p((u, v), G') = 1$, the edge (u, v) has to be in all the minimal SCSs of G'.

Then, we define the *Core Centralized Score* of graph G as,

$$CE\left(G\right) = \frac{1}{|\overline{V}|}\sum_{v\in\overline{V}}\frac{1}{|\tilde{\Gamma}(v)|}\sum_{u\in\tilde{\Gamma}(v)}p(u,v)\frac{\kappa(u)-\kappa(v)}{k^*-\kappa(v)}. \quad (4)$$

Type	Network	$\lvert V\rvert$	$\lvert E\rvert$	k^*	CE	Range Minimal SCSs
AS	733_19971108†	3015	5196	9	0.53	91.3-91.6%
	733_19990309†	4759	8896	12	0.49	92.9-93.1%
	Ore1_010331†	10,670	22,002	17	0.56	94.5-94.6%
	Ore1_010428†	10,886	22,493	17	0.56	94.8-94.9%
BIO	Dmela‡	7393	25,569	11	0.47	87.8-88.0%
	Yeast_Protein‡	1846	2203	5	0.16	82.1-82.8%
CA	GrQc†	5241	14,484	43	0.05	89.4-89.6%
	HepTh†	9875	25,973	31	0.05	83.4-83.6%
	Erdos992‡	5094	7515	7	0.38	91.2-91.5%
INF	OpenFlights‡	2939	15,677	28	0.44	90.3-90.6%
	Power‡	4941	6594	5	0.06	79.9-80.4%
	USAir97‡	332	126	26	0.63	92.5-92.9%
P2P	Gnutella08†	6301	20,777	10	0.27	78.4-78.7%
	Gnutella09†	8114	26,013	10	0.32	85.1-85.3%
	Gnutella25†	22,687	54,705	5	0.89	80.4-80.5%
SOC	Hamsterster‡	2426	16,097	24	0.34	89.3-89.5%
	Advogato‡	5167	39,432	25	0.70	89.6-89.7%
	Wiki_Vote‡	889	2914	9	0.39	84.2-85.0%
	Wiki_talk†	2.3M	5.0M	131	0.89	99.0-99.0%
	Youtube†	1.1M	3.0M	51	0.62	93.5-93.5%
TECH	PGP‡	10,680	24,316	31	0.08	84.5-84.8%
	Routers_rf‡	2113	6632	15	0.26	86.2-86.6%
	WHOIS‡	7476	56,943	88	0.36	91.5-91.6%
WEB	Spam‡	4767	37,375	35	0.40	91.7-91.9%
	Webbase‡	16,062	25,593	32	0.07	94.1-94.2%
SOCFB	Amberst‡	2235	90,954	63	0.69	80.4-80.6%
	Bowdoin‡	2250	73,643	56	0.67	83.2-83.4%
	Colgate‡	3482	155,053	65	0.68	79.6-79.7%
	Caltech‡	762	16,651	35	0.81	80.9-81.2%
EMAIL	Univ‡	1133	5451	11	0.38	84.5-85.0%
	EU‡	32,430	54,397	22	0.75	97.2-97.3%
	Enron†	33,696	180,811	43	0.49	93.5-93.6%
ITEM	Amazon†	334,863	925,872	6	0.67	80.6-80.7%

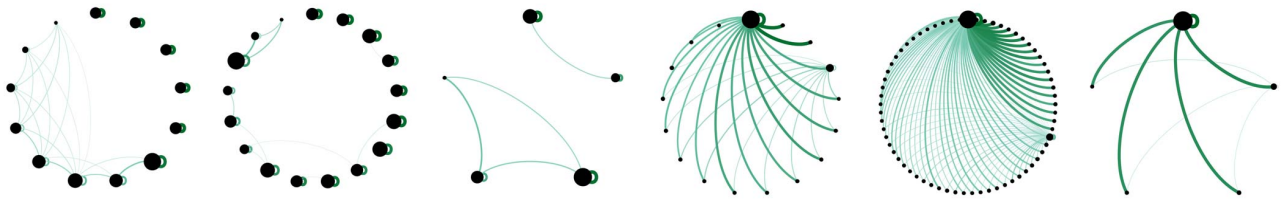
TABLE II: Networks used for experiments. In this table, $|V|$ is the number of nodes, $|E|$ is the number of edges, and k^* is the degeneracy. These data were downloaded from SNAP (denoted by †) and Network Repository (denoted by ‡).

VII. Analysis of Skeletal Core Graphs

In this section, we first present a technique for visualizing the skeletal core subgraphs of various real-world networks. Then, we perform a comparative analysis of the subgraphs found by the algorithm presented in Section V.

We consider networks from diverse domains, presented in Table II. In Table II, the domains of the network are indicated by AS for the autonomous system networks, BIO for biological networks, CA for collaboration, INF for infrastructure, P2P for peer-to-peer, SOC for social, TECH for technological, WEB for internet, SOCFB for Facebook, EMAIL for email, and ITEM for item. k^* represents the degeneracy of the network and CE is the core centralized score.[2]

[2]Data downloaded from https://networkrepository.com/index.php and https://snap.stanford.edu/data/.



(1) CA_HepTh (2) WEB_Webbase (3) BIO_Yeast_Protein (4) Oregon1_010331 (5) SOCFB_Colgate (6) P2P_Gnutella25

Fig. 2: Visualization of SCS structure of various real networks, as described in text. Graphs are arranged in increasing order of their core centralized score.

A. Visualization

Using the concepts defined earlier, one can nicely visualize the core structure of a graph G via the following steps:

1) If G is not already a skeletal core subgraph, assign edge weights as in Equation 2. If G is itself a skeletal core subgraph, edges have weight of 1. Denote the edge weights by $w(u, v)$.
2) Merge nodes belonging to the same shell into a super-node containing all the nodes in V_i.
3) Two super-nodes s_i and s_j have an edge between them if there is an edge (u, v) such that: (a) $u \in V_i$ and $v \in V_j$, and (b) $(u, v) \in E$.
4) Suppose that $i \leq j$. Then the weight of edge (s_i, s_j) is given by:

$$w(s_i, s_j) = \frac{\sum_{(u,v):u \in V_i \wedge v \in V_j} w(u, v)}{\sum_{(u,v):u \in V_i} w(u, v)}.$$

That is, the weight of (s_i, s_j) is the ratio of the sum of all the edges between V_i and V_j and the sum of all the edges with at least one endpoint in V_i. Note that we allow for self-loops, and the edges are directed (from i to j).
5) The node weight of s_i is the sum of all the in-edges to s_i.
6) The super-nodes are arranged in a circle in decreasing order of the k value for each shell. The degeneracy shell is at the 12 O'clock position.
7) The size of each node is adjusted according to the node weight, and the thickness of edges is adjusted according to the edge weight. That is, larger nodes represent the shells on which other shells depends on for their core number, and thicker edges represents strong dependence of the lower shell to the higher shell.

Figure 2 shows the skeletal core subgraph visualization for various real-world graphs. The graphs are arranged in increasing order of their core centralized scores. In these figures, edges with weight less than 0.1 are not shown for visual clarity.

We observe that in the cases of very low core centralized scores (left), different shells are almost independent of each other, with few inter-shell connections. On the other extreme (right) we have those cases with very high core centralized score – almost all shells depends on the degeneracy shell.

These visualizations give us an intuitive understanding of some properties such as the resilience and cascading collapse. As an example, in a cascading collapse of k-core structure [6], there are few inter-shell connections in a decentralized graph, and so cascading failures are limited. On the other hand, in centralized graphs, a failure in a higher shell can affect lower shells as well.

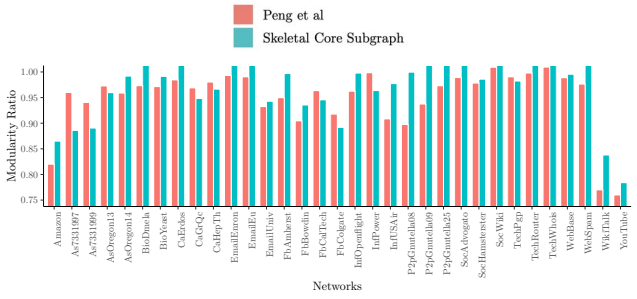
We observe some consistent patterns across domains. For instance, co-authorship networks tend to have a low core centralized score, possibly indicating large, mostly disjoint cohorts of collaborating researchers. In contrast, online social networks have high values. We hypothesize that this is because adding new friends takes almost no effort, and the only path for information flow is thorough such connections. Thus, there is a huge incentive to connect to the central parts of the network.

B. Properties of Skeletal Core Subgraphs

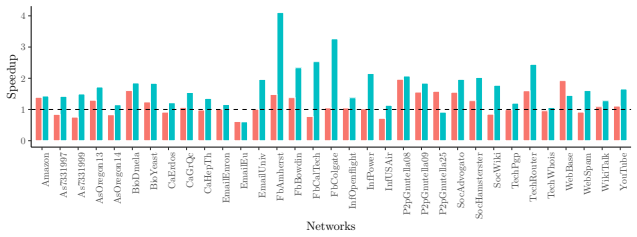
Here, we analyze the structures of the minimal SCSs. For each graph, we first generate 100 minimal SCSs using `GreedySCS`, and present their sizes. Then, we compute the core centralized score $CE(G)$, as described earlier. Higher values indicate a more centralized core structure.

Results are presented in Table II. In all datasets, the minimal SCSs are quite large: the smallest (P2P_Gnutella08) is still approximately three-quarters the size of the entire network. The P2P networks and SOCFB networks all exhibit comparatively small minimal SCSs, suggesting that a significant fraction of edges in these networks do *not* go towards supporting the core structure. In contrast, the autonomous systems networks have large minimal SCSs, indicating that almost every edge is important to the core structure.

The Facebook networks tend to exhibit relatively high CE (core centralized) scores, while the tech and co-authorship networks are much less centralized. We hypothesize that this is due to the role of the networks: in online social media networks, connecting to central/influential nodes is easy and desirable, while forming connections to central nodes in other types of networks may be harder. The power network is extremely decentralized, perhaps because it was designed specifically to be resilient.



(1) Comparison of community quality detected by [4] to the SCS modification. 'Modularity ratio' indicates the ratio of the modularity of the communities found using a seed subgraph to that of the communities found by using the entire network. Higher is better.



(2) Comparison of speedup to detect community by [4] to the Skeletal Core Subgraph modification. Higher values are better. Running times include the time required to find the seed subgraph.

Fig. 3: Experimental comparison demonstrating the application of Skeletal Core Subgraph in community detection. In both plots, higher values are better. With SCSs, we are able to find good quality communities, significantly faster.

VIII. Application: Community Detection

k-cores can be used to accelerate community detection using the technique described in [4]. This method first runs a community detection algorithm on a k-core subgraph of appropriate size and then extrapolates results outwards. We modify the method described in [4] by changing the seed subgraph selection: instead of selecting the seed nodes as those with highest core numbers, we select the same number of nodes, but those that are in the shells with the highest sum of weights in the visualization as described in Section VII.

To verify, we run community detection using Louvian method [15] to act as a basis for comparison. We then perform community detection as described in [4] (with subgraph selected by core number), and then with the skeletal core subgraph modification. In both cases, the number of nodes in the seed subgraphs are the same– roughly 30% of the nodes in the graph, though this varies by graph. We compare the quality of the communities detected by both methods through modularity. We consider the ratio of modularity of the communities found through [4], and then with the skeletal core subgraph modification to the ones found by performing community detection on the whole graph.

Figure 3 shows the results of the experiments on various real-world graphs. We can see that with the skeletal core subgraph modification, the algorithm is able to achieve better communities compared to the original method. We can also see that the speedup for the community detection, shown in Figure 3, is improved significantly. In nearly all networks, using the SCS to define the seed subgraph is faster and results in higher quality communities (as measured by modularity), as compared to using the k-core to define the seed subgraph.

IX. Conclusion

In this paper, we introduced the concept of the **Skeletal Core Subgraph (SCS)** of a network. A SCS can be thought of as a skeleton of the k-core Structure of the graph. We show that graphs can be categorized along the spectrum between **Decentralized Skeletal Core Subgraph** and **Centralized Skeletal Core Subgraph**. We provided a greedy heuristic to find SCSs, and then studied the SCS properties of 30 real-world graphs from different domains. Finally, we explored the application of the Skeletal Core Subgraph on the process of community detection of graphs, showing that the SCS can be used to find good quality communities, while improving the speedup of the process.

References

[1] M. A. Al-garadi, K. D. Varathan, and S. D. Ravana, "Identification of influential spreaders in online social networks using interaction weighted k-core decomposition method," *Physica A: Statistical Mechanics and its Applications*, vol. 468, pp. 278–288, 2017.

[2] E. Gregori, L. Lenzini, and C. Orsini, "k-dense communities in the internet as-level topology," in *COMSNETS*, 2011.

[3] C. Orsini, E. Gregori, L. Lenzini, and D. Krioukov, "Evolution of the internet k-dense structure," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, 2014.

[4] C. Peng, T. G. Kolda, and A. Pinar, "Accelerating community detection by using k-core subgraphs," *arXiv preprint arXiv:1403.2226*, 2014.

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

[6] R. Laishram, A. E. Sariyüce, T. Eliassi-Rad, A. Pinar, and S. Soundarajan, "Measuring and improving the core resilience of networks," in *Proceedings of the 2018 World Wide Web Conference*, 2018.

[7] R. Laishram, A. Erdem Sar, T. Eliassi-Rad, A. Pinar, and S. Soundarajan, "Residual core maximization: An efficient algorithm for maximizing the size of the k-core," in *SIAM*. SIAM, 2020.

[8] D. Matula and L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *Journal of the ACM*, vol. 30, no. 3, 1983.

[9] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Finding critical users for social network engagement: The collapsed k-core problem," in *AAAI*, 2017.

[10] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, "A model of internet topology using k-shell decomposition," *Proceedings of the National Academy of Sciences*, vol. 104, no. 27, 2007.

[11] F. Morone, K. Burleson-Lesser, H. Vinutha, S. Sastry, and H. A. Makse, "The jamming transition is a k-core percolation transition," *Physica A*, vol. 516, 2019.

[12] F. Morone, G. Del Ferraro, and H. A. Makse, "The k-core as a predictor of structural collapse in mutualistic ecosystems," *Nature Physics*, vol. 15, no. 1, 2019.

[13] N. Akpan and I. Iwok, "A minimum spanning tree approach of solving a transportation problem," *International Journal of Mathematics and Statistics Invention*, vol. 5, no. 3, pp. 09–18, 2017.

[14] S. Scellato, A. Cardillo, V. Latora, and S. Porta, "The backbone of a city," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 50, no. 1, pp. 221–225, 2006.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and É. Lefebvre, "The louvain method for community detection in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 10, 2011.