Skeletal Cores and Graph Resilience

Danylo Honcharov[1(✉)], Ahmet Erdem Sarıyüce[2], Ricky Laishram[1], and Sucheta Soundarajan[1]

[1] Syracuse University, Syracuse, NY 13244, USA
{dhonchar,rlaishra,susounda}@syr.edu
[2] University at Buffalo, Buffalo, NY 14260, USA
erdem@buffalo.edu

Abstract. In network analysis, one of the most important structures is the k-core: the maximal set of nodes such that each node in the k-core has at least k neighbors within the core. Recently, the notion of the *skeletal k-core*– a minimal subgraph that preserves the core structure of the graph– has attracted attention. However, the literature to date has contained only a biased greedy heuristic for sampling skeletal cores, which resulted in a skewed analysis of the network. In this work, we introduce a novel MCMC algorithm for sampling skeletal cores uniformly at random, as well as a novel algorithm for estimating the size of the space of skeletal k-cores, which, as we show, is important for understanding the core resilience of the network. With these algorithms, we demonstrate the relationship between *resilience* of the network and the core structure of the graph and suggest fast heuristics for evaluating graph structure from a skeletal cores perspective. We show that the normalized number of skeletal cores in the graph correlates with the resilience of k-core towards edge deletion attacks.

Keywords: networks · robustness · k-cores

1 Introduction

Within the machine learning and network science literature, the study of dense subgraphs has received a great deal of attention. Examples of such structures include cliques [18], k-clubs [21] and k-trusses [7]. Of particular interest is the k-core, which plays a role in applications such as community detection [19,22], influence maximization [11], visualization [3,12], anomaly detection [26] and understanding network topology [3,27]. A network's k-core is defined as the maximal subgraph of the network such that every node in the subgraph has at least k neighbours also in the subgraph [24]. The core number of a node is the highest value of k for which that node belongs to a k-core.

k-cores have a history of being studied in the context of network resilience [17, 20]. To understand the structural properties of k-cores and their effect on resilience, the concept of the *skeletal core* was proposed [16]. The skeletal core of a graph is defined as a minimal subgraph that preserves the core number of each node. Skeletal cores can be seen as a "backbone" of the k-core structure of

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14171, pp. 293–308, 2023.
https://doi.org/10.1007/978-3-031-43418-1_18

the graph, and so play an important role in the structure of the network. While useful for applications such as speeding up community detection, they are crucial in the context of analyzing the network's *resilience* or *robustness* [16]. To this end, understanding the space of skeletal cores can provide valuable insight into the graph structure: understanding when a network's core structure will change, requires understanding when it remains the same.

However, because skeletal cores are a new concept, there are a number of important open problems surrounding them. For instance, the literature contains only a single greedy heuristic for identifying a single skeletal core, and it is not known how to sample skeletal cores of a network uniformly at random, which fundamentally limits the understanding of how skeletal cores are distributed and how they affect the core structure. Relatedly, it is also not known how to estimate the number of skeletal cores to which an edge belongs, making it difficult to gauge the importance of an edge with respect to the overall core structure of the network. Without solving these problems, one cannot properly understand the effect of a network's skeletal core structure on the resilience of the network.

In our work, we first introduce a novel MCMC (Markov Chain Monte Carlo) algorithm for sampling skeletal k-cores uniformly at random and a corresponding algorithm for estimating the number of skeletal cores in the graph. We then use these algorithms to study the relationship between the space of skeletal cores and the robustness of complex networks. We suggest practical heuristics for the estimation of the probability of a given edge being part of a skeletal core and experimentally validate these heuristics with respect to the ground truth.

Our main contributions are as follows:

1. We provide a novel MCMC algorithm for sampling skeletal cores of the graph uniformly at random.
2. We suggest heuristics that can be used to estimate the likelihood of an edge being part of a skeletal core and evaluate them experimentally.
3. We provide a novel algorithm for the estimation of the size of the skeletal core space and experimentally demonstrate the relationship between the number of skeletal cores in the graph and the core resilience of the network.
4. We demonstrate how the proposed algorithms can be used to identify moments of fundamental change of the k-core structure during a process of edge deletion.

2 Related Work

First introduced by Seidman in 1983 [24], the k-core of a graph $G = (V, E)$ is defined as the maximal connected subgraph such that any node in the subgraph has a degree at least k. k-cores are an important part of network analysis, and have been used for many tasks, including community detection [19], speedups of the graph algorithms [22], influence maximization [11], anomaly detection [26], prediction of protein functions [2], and others. k-cores have also been used to gain deeper insight into a graph structure, including through visualization [12].

A skeletal core of a graph G is defined as a minimal subgraph H of G such that all nodes in H have the same core number as in G [16]. The literature contains a greedy algorithm (discussed in more detail in Sect. 4.2) for generating a single skeletal k-core for the given graph and suggests several applications for skeletal cores [16]. However, the greedy algorithm for finding skeletal cores does not sample uniformly at random from the whole space of skeletal cores, and so may lead to bias in analysis. Moreover, because graphs may have many skeletal cores, simply generating one such skeletal core may not provide sufficient insight into the graph structure. The estimation of the *expected* properties of skeletal cores (e.g. expected size of skeletal core or expected overlap between skeletal cores) is a more useful tool for graph analysis, but it requires the ability to sample cores uniformly at random.

Skeletal cores are particularly important to the robustness of networks. While network robustness can be defined in many ways [8,10], of relevance to this work is the resilience of a graph's k-core structure against changes in the graph structure. k-core numbers are used as a way to quantify connectedness and importance of the nodes in the network, and core structure can be used to study the local density of the graphs. As such, it is important to estimate how these properties may be affected if the graph is changed: for example, how do random communications failures in a communications network affect overall connectivity?

The core resilience measure was suggested to measure the propensity of nodes to change core number when edges from the graph are dropped [17], and the impact of noise and sampling process on the k-core of a graph has been studied [1]. Understanding when changes in the graph structure happen during some processes (e.g. edge removal/addition) has been a long-standing research topic in network science. Examples include research on changes in random networks [9], including the emergence of k-cores in the random graphs [23] and dynamics of the k-core during edge removal in the random graphs [13].

3 Background

Here, we introduce concepts necessary for the presentation of our work.

3.1 k-Cores

The k-core is defined as a maximal subgraph for which any node in the subgraph is connected to at least k neighbours in subgraph. A node's core number is the maximum value k such that the node belongs to a k-core. The core numbers of a graph can be obtained using a 'peeling' k-core decomposition technique, which runs in $O(|E|)$ time [4].

3.2 Core Strength

Core strength was introduced in [17]. The core strength of a node u provides an upper bound on the number of edges to same or higher-shell neighbors that can

be removed from the node without it decreasing its core number. The proposed MCMC algorithm (Sect. 4.1) uses core strength to identify transitions; and the greedy estimator algorithm from Sect. 4.2 uses it for estimating the number of skeletal cores.

Formally the core strength of u is defined as $CS(u) = |N_{\geq}(u)| - K(u) + 1$, where $N_{\geq}$ is the set of neighbours of u with a core number greater than or equal to that of u (i.e., those that support u's core number) and $K(u)$ denotes the core number of u.[1]

3.3 Core Valid Subgraph and Skeletal k-Core

A Core Valid Subgraph CVS of a graph $G = (V, E)$ is defined as a spanning subgraph of G such that all nodes in G have the same k-core number in CVS as they do in G [16]. In other words, CVS preserves the core structure of G.

A skeletal k-core is defined as a minimal CVS: i.e., one for which removal of any edge would lead to at least one node decreasing its core number [16]. Of interest in the context of our work is the greedy algorithm for finding a single skeletal core [16]. This algorithm serves to identify the starting state for the MCMC algorithm in Sect. 4.2, and can be summarized as follows:

1. Identify a set of edges of G such that any edge in the set can be removed without k-core numbers decreasing. Denote this set as R.
2. Select one of the edges $e \in R$ and remove it from the G.
3. Recompute R and repeat 1-3, until $R = \emptyset$.
4. Return G as a skeletal core.

3.4 Core Resilience

Core resilience was proposed in [17] to measure the robustness of a network's k-core structure against random edge removal. It provides a way to compare the estimated ability of the core structure of different networks to withstand changes (for example, due to failure of the communication channels between nodes or because of network evolution). Consider two graphs: $G = (V, E)$ and $G' = (V, E')$, where G' is a subgraph of G formed by randomly deleting $p\%$ of the edges from G. Denote the top $r\%$ of the nodes with the highest k-core numbers in G as V_r. The resilience $R_r^{(p)}$ of G can then be defined as: $R_r^{(p)} = \tau_b(\{((K(u, G), (K(u, G')), u \in V_r\})$, where $K(u, G)$ denotes the core number of node u in graph G, and τ_b denotes the expected Kendall-Tau rank correlation [14] between the two rankings (other rank correlations may be used). $R_r^{(p_1,p_2)}$ is defined as the mean core resilience as the percentage of dropped edges changes from p_1 to p_2 [17]: $R_r^{(p_1,p_2)} = \frac{\int_{p_1}^{p_2} R_r^{(x)}(G)dx}{p_2 - p_1}$. We use core resilience in our

[1] Note that this sometimes overestimates the desired value, as loss of an edge (u, v) can trigger reductions in core numbers of other nodes, and thus lower the core number of other neighbors of u; however, computing the exact value is more computationally intensive.

experimental analysis to show the relationship between skeletal cores and the robustness of the graph.

4 Algorithms to Explore the Space of Skeletal Cores

Here, we introduce an MCMC-based algorithm for sampling skeletal cores u.a.r., and then demonstrate how to estimate the number of skeletal cores in a graph.

Exploring the space of skeletal cores in unbiased way is useful for different applications. Let's consider the scenario of the pandemic of a contagious virus which spreads through the network of personal interactions. As was shown by the research of COVID-19 [25], k-cores of high complexity are known to sustain an outbreak even if the network becomes partially disconnected; thus, estimation of *which* edges are most important for the k-core could be useful to minimize the spread of the disease. While skeletal cores of the personal interaction network provide important insight regarding these edges, application of previously suggested greedy algorithm will provide a biased sample of the skeletal cores, leading to the non-optimal decision-making. Similarly, for the opposite problem of maintaining the k-core structure, a skewed sample of skeletal cores is undesirable. The proposed MCMC algorithm does not suffer from this drawback, and is guaranteed to provide unbiased sampling.

The proposed MCMC algorithm uses the notion of *core strength* [17] (described in Sect. 3.2) to identify transitions between different skeletal k-cores.

In our experimental analysis, we show that the core resilience of a graph is strongly correlated with its skeletal core properties.

4.1 Sampling Skeletal Cores Uniformly at Random

In this section, we describe an MCMC algorithm for sampling skeletal k-cores uniformly at random. At a high level, the proposed method is described as following: First, begin from any skeletal core T_0 (for example, one obtained by the greedy algorithm in [16]). This skeletal core is the initial state of a Markov Chain M. Next, randomly transform T_0 into another skeletal core T_1, or stay at T_0, with probabilities of D/D_{max} and $1 - D/D_{max}$ correspondingly, where D stands for the number of possible transformations from the current state. D_{max} needs to be bigger than any possible number of transitions from one state. This is equivalent to conducting a random walk over M. Repeat the procedure until the process converges to the stationary distribution. If transition probabilities are defined correctly, this stationary distribution will be an uniform distribution over the space of skeletal cores. Once the stationary distribution is reached, return the current skeletal core.

The key idea behind the proposed algorithm is to correctly transition between skeletal cores of the graph. When at a skeletal core T, no edge can be deleted from T without affecting core numbers (because T is skeletal), unless another edge (or edges) is added to compensate. When these replacement edges are added, this may necessitate further removal of edges to ensure that the new subgraph is still skeletal. We consider only allowed transitions that go up to two steps in

any direction, and we show in the proof that this is enough to reach any skeletal core, while limiting the number of possible transitions at each step.

More formally, the proposed algorithm consists of the following steps:

1. Begin from some skeletal core T_0 of the original graph $G = (V, E)$. T_0 can be obtained from the original graph by using a greedy algorithm, like that proposed in [16], or in any alternative way.
2. Initialize the set of possible transitions $R_0 = \emptyset$.
3. Denote the current skeletal core as T. To identify possible transitions, iterate over all edges $(u, v) \in T$ and generate all skeletal cores that can be obtained from T by removal of an edge (u, v), followed by the addition of an edge $(u, i) \in G \backslash T$ or an edge $(v, j) \in G \backslash T$ (or both) and corresponding removal of $(i, p) \in T$ and/or $(j, k) \in T$, if needed. Add these transitions to R.
4. Similarly, for all edges $(u, v) \in G \backslash T$, generate all skeletal cores, that can be obtained from T by addition of (u, v), the removal of $(u, i) \in T$ or $(v, j) \in T$ (or both), and corresponding addition of $(i, p) \in G \backslash T$ and/or $(j, k) \in G \backslash T$. Add these transitions to R.
5. Select one of the transitions from R uniformly at random or stay at T with probability $1 - D/D_{max}$, where D_{max} needs to be bigger than any possible number of transitions from one state. D_{max} can be bounded in several ways - for example, it's trivial to show that $D_{max} \leq |E| * d_m^4$, where d_m is the highest degree in the graph.[2]
6. Repeat steps 2–5 until the Markov Chain converges. Convergence can be identified in several ways [6]. One simple example could be running several instances of Markovian chains and comparing their outputs.
7. Return the current skeletal core.

The running time for this algorithm is high. In each iteration, the algorithm iterates over all edges and considers $O(d_m^4)$ possible transitions for every edge in the worst-case scenario. Hence, the running time for one step is $O(|E| * d_m^4)$. The overall running time of the proposed algorithm depends primarily on the mixing time of the Markov chain M. While we do not propose proof that M is rapidly mixing, experiments suggest that the stationary distribution is reached relatively fast in most cases. Nonetheless, the main disadvantage of the algorithm is running time, which makes it prohibitively expensive to run on big graphs.

In the next section, we discuss how properties of the space of skeletal cores can be estimated more quickly.

Theorem 1. *The described MCMC algorithm will sample skeletal cores u.a.r.*

Due to space constraints, the proof of the theorem can be found in the extended version of this paper.[3]. In the proof, we show that space of skeletal cores is connected and that transitions between adjacent states are symmetric.

[2] Experimentally this bound proved to be very loose, which may reduce the rate of convergence.

[3] Extended version and source code are available at https://github.com/honcharov-danylo/extended_skeletal.

def *estimator*$(G = (V, E) : Graph, K : HashTable)$**:**
  $R = \emptyset$; $c = 0$; $d = 1$
  repeat
    $CS = getCoreStrength(G)$
    $R = \{(u, v) \in E : (CS[u] > 1 \wedge CS[v] > 1) \vee$
    $\vee (CS[u] > 1 \wedge K(u) < K(v))) \vee (CS[v] > 1 \wedge K(v) < K(u)))\}$
    $d* = |R|$; $c+ = 1$
  until $R! = \emptyset$
  return $d/c!$
def *estimate_size*$(G = (V, E) : Graph, K : HashTable, N : int)$**:**
  $S = List()$; $W = List()$
  for $i = 0; i < N; i++$ **do**
    $S.add(estimator(G, K))$
  end
  return $AVG(S)$

Algorithm 1: Number of skeletal cores in the graph

4.2 Estimating the Number of Skeletal Cores

Analyzing the properties of the skeletal cores is important for many applications. For instance, the k-core can be seen as a form of the equilibrium in a game-theoretic model [5]; and, correspondingly, the skeletal core can be seen as the minimal edge-induced subgraph that maintains this equilibrium. The question of what this subgraph looks *on average* is crucial for understanding the network as a whole. Another example could be using the average size of skeletal cores to find the "breaking point" of the k-core structure, as shown in Sect. 5.4.

Because the MCMC algorithm is computationally expensive, here we suggest an alternative approach for the estimation of the *expected* properties of skeletal cores. We provide an algorithm that provides the expected number of skeletal k-cores in the graph, and explain how it can be used to estimate properties of skeletal cores (e.g. average size) without relying on the MCMC approach. An outline of the algorithm can be found in Algorithm 1, further referred to as the GE-algorithm. The algorithm takes a graph and k-core numbers of all nodes as input and returns the expected number of skeletal cores in the graph.

The idea behind the algorithm is as follows: first, "unravel" the DAG H, formed by transitions of the greedy heuristic for finding a single skeletal core [16] (discussed in Sect. 3.3) to the tree and count its leaves, adjusting our estimate for overcounting caused by such "unravelling". To improve the estimate, we repeat the process n times and use D_{avg} as the final estimate.

4.3 Proof of the Algorithm Correctness

Theorem 2. *Algorithm 1 returns the expected number of skeletal k-cores in the graph.*

Proof. Denote the original input graph as G. Use a greedy algorithm, (see Sect. 3.3) to obtain a single skeletal core S from G. Next, create a new graph H,

where nodes denote graphs that can be obtained during the process above and directed edges denote transitions between these graphs in the greedy algorithm. It can be shown that the greedy algorithm is capable of producing every skeletal core in the graph. H is directed and acyclic, because an edge from $u \in H$ to $v \in H$ requires that v is a subgraph of u in G. Nodes without out-edges are skeletal cores of G (by definition). The root of H is G itself. Next, modify H by "unravelling" the DAG to a tree by making copies of all nodes that have more than one parent. Denote the resulting graph as H_m.

Because H_m is a tree, obtaining the expected number of terminal nodes (leaves) in H_m could be done easily with the algorithm suggested by Knuth [15] for the estimation of the space of the backtracking tree. This algorithm initializes a counter D with 1 at the root of the tree (which denotes the original graph G) and performs a random walk down the tree, multiplying D by the out-degree of every node on the path, until it reaches a leaf. When a leaf is reached, the counter will contain the expected number of the leaves in the graph [15].

However, the skeletal cores (leaves) in H_m may be copied (and counted) several times. As edges removed from G to obtain any skeletal core S_i may have been removed in any order, there are $A_i!$ ways to reach S_i from G in H, where A_i is the number of edges removed from G to obtain S_i. As any of these paths will lead to a unique copy of the skeletal core S_i in H_m, we know that S_i will be counted $A_i!$ times in H_m. Dividing the estimate by $A_i!$ accounts for this.

Theorem 3. *The running time of Algorithm 1 is* $O(|E|)$.

Proof. First, consider the running time of the function *estimator*. To compute core strength, the function requires $O(|E|)$ time in the first iteration. For subsequent iterations, core strength values can be updated in $O(1)$ after every edge deletion (as at most two values of Core Strength will be affected). Identifying a set of edges R takes $O(|E|)$ time the first time, but we can update it quickly if the selected data structure allows us to quickly access and remove edges with one known endpoint. One example of such a data structure could be a hashmap with nodes as keys and edges from R as values. The function *estimate_size* takes $O(k|E|)$ time, where k is the number of samples. Assuming that k is a small fixed constant with $k << |E|$, the overall running time of the algorithm is $O(|E|)$.

4.4 Estimation of Expected Properties of Skeletal Core

We can use a modified version of Algorithm 1 to obtain the expected properties of skeletal cores in the graph. Suppose that we want to get the expected value of property P of the skeletal core (e.g., the number of edges or average degree).

Denote $P(S_i)$ as the value of P for the core S_i and define $C = \sum_{S_i \in S} P(S_i)$. C is the sum of P of all skeletal cores in the space S. Assign 0 as the *cost* of any node in the H which is non-terminal (i.e., not a skeletal core), and assign $P(S_i)$ as the cost of the skeletal core S_i.

In this case, an expected estimate of *total cost* C for one Monte Carlo search will be $P(S_i) * D$ [15], and over multiple experiments it will be $C = \sum_{i=0}^{n} P(S_i) * D_i/n$. The expected value of P will then be $E[P] = C/D_{avg} = P(S_i) * D_i/D_{avg}$.

4.5 Normalized Number of Skeletal Cores

It is useful to introduce a notion of skeletal core density, which allows one to have a sense of the number of skeletal cores that a graph has relative to its size. To this end, we introduce a novel metric of a normalized number of skeletal k-cores. Denoting the expected size of the space of skeletal cores as e_{est} and the expected number of edges in the skeletal core as s_{exp}, we define the *normalized* number of skeletal cores in the graph as $e_n = log(e_{est})/log(\binom{|E|}{s_{exp}})$. The denominator represents the logarithm of the total number of possible subgraphs in G that have s_{exp} edges.

5 Experiments and Analysis

In this section, we demonstrate the relationship between skeletal k-cores and core resilience; and show how skeletal cores can be used to analyze the "breaking point" of the k-core structure.

5.1 Datasets

Networks used in our experiments are listed in Table 1. We compare networks from different domains: AS denotes networks from autonomous systems; P2P stands for peer-to-peer, BIO indicates graphs from bioinformatics; CA denotes co-authorship networks; INF is used for infrastructure-related graphs; CO denotes collaboration networks; SOC indicates social networks; TECH stands for technological networks; WEB is used for internet network; EMAIL indicates email networks; MISC denotes networks that do not fall into the categories above.

5.2 Algorithm Validation

Before using the proposed algorithms to perform network analysis, we demonstrate that they are effective at the desired objectives.

First, we show that the MCMC sampling algorithm reaches its stationary distribution quickly. Any MCMC algorithm needs several steps before convergence. While we do not provide theoretical proof that Markovian chain defined by the suggested algorithm is rapidly mixing, experiments demonstrate fast convergence of the algorithm. To show the speed of convergence of the algorithm experimentally, we perform a sampling of skeletal cores for a different number of steps, setting the number of steps to be equal to the fixed fractions of the number of edges in the original graph.

Intuitively, after convergence, the distribution of the properties of skeletal cores is stable. In Fig. 1, the distribution of sizes of skeletal cores is plotted for a different number of steps of the algorithm. Due to the lack of space, we show

Table 1. Datasets. $|V|$ denotes number of nodes, $|E|$ denotes number of edges, k_{max} denotes highest k-core. † denotes SNAP as a source of the network, ‡ stands for NetworkRepository, § stands for KONECT, §§ stands for Netzschleuder.

Type	Network	$\|V\|$	$\|E\|$	k_{max}	Type	Network	$\|V\|$	$\|E\|$	k_{max}
AS	auto_as19990111†	549	1249	11	CO	arena_jazz‡	198	2742	29
	auto_as19980318†	3455	6168	10	SOC	wiki‡	889	2914	9
	auto_as19971108†	3015	5156	9		hamsterer‡	2426	16630	24
	oregon010331†	10670	22002	17		musae_facebook†	22470	170823	56
P2P	gnutella08†	6301	20777	10		musae_git†	37700	289003	34
BIO	dmela‡	7393	25569	11	TECH	tech_routers‡	2113	6632	15
	protein‡	1870	2203	5		whois‡	7476	56943	88
CA	erdos‡	5094	7515	7		tech_pgp‡	10680	24316	31
	netscience§	1464	2744	19	WEB	webspam‡	4767	37375	35
	ca-HepPh‡	12008	118521	238	EMAIL	email_enron†	36692	183831	43
	ca_grq†	4158	13422	43		email_EuAll†	265214	364481	37
INF	openflights‡	2939	15677	28	MISC	moreno_innovation§	245	927	6
	inf-power‡	4941	6594	5		norwegian(net1m)§§	1421	3855	11
						moreno_oz‡	217	2345	14

plots only for 3 networks. For higher number of steps, the distributions of skeletal k-core sizes look very similar, which suggests convergence.[4]

As was seen in Sect. 4.4, we can compute the expected values of properties of skeletal cores. To test this algorithm, we compare these estimated properties to the average properties of skeletal cores sampled uniformly at random using MCMC algorithm. Due to space limitations, we perform only an estimation of the expected number of edges in the skeletal cores for selected graphs.

Results can be seen in Table 2. We compute E_{avg} as a simple average over sizes of skeletal cores, sampled with the greedy algorithm from [16], E_{exp} is the expected value over skeletal cores (method from Sect. 4.4), and E_u is obtained by sampling skeletal cores uniformly at random with the MCMC algorithm. Our goal is to show that E_{exp} is closer to E_u than E_{avg} is to E_u. Indeed, for most graphs, the suggested expected estimate is much closer to E_u, the ground truth obtained from the MCMC algorithm (and for the two networks where it is not closer, the difference is very small).

5.3 Skeletal Core Analysis

Here, we experimentally validate the theoretical results of the GE-algorithm to estimate the number of skeletal cores, as presented in Sect. 4.2. For every graph, we sample 50 skeletal cores and compute the normalized number of skeletal cores as shown in Sect. 4.5. Results are plotted against core resilience $R_{50}^{(0,50)}$ in Fig. 2. (Core resilience is computed as described in Sect. 3.4.)

[4] Such convergence diagnostics for MCMC methods don't *guarantee* convergence, and should be seen as a type of statistical analysis [6].

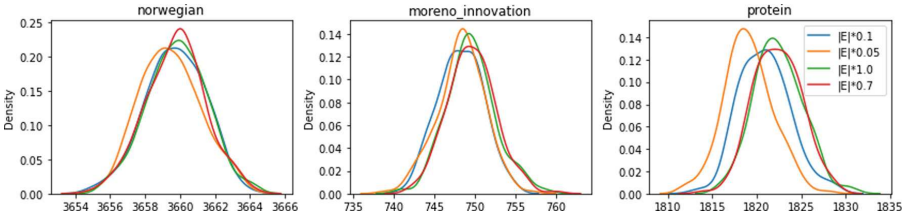

Fig. 1. Distribution of size of skeletal cores, sampled with a different number of steps of MCMC (number of steps are set up as fixed fractions of the number of edges in the graph). Colors denote the number of transitions made by the algorithm until a sample was taken. (Color figure online)

Table 2. Expected number of edges in skeletal cores. E_{avg} is simple average over skeletal cores, sampled with a greedy algorithm, E_{exp} is the expected value over skeletal cores (method from Sect. 4.4), E_u is obtained by MCMC algorithm.

Network	E_{avg}	E_{exp}	E_u
auto_as19971108	4717.13	**4724.53**	4722.28
auto_as19980318	**5649.84**	5648.00	5655.04
auto_as19990111	**1130.45**	1128.28	1131.47
GrQc	11926.69	**11936.83**	11953.45
netscience	2628.29	**2629.01**	2630.30
norwegian	3657.30	**3657.82**	3660.14
wiki	2464.67	**2466.16**	2470.61
erdos	6864.28	**6870.62**	6877.34
protein	1816.31	**1817.76**	1821.81
inf-power	5286.41	**5288.73**	5314.86

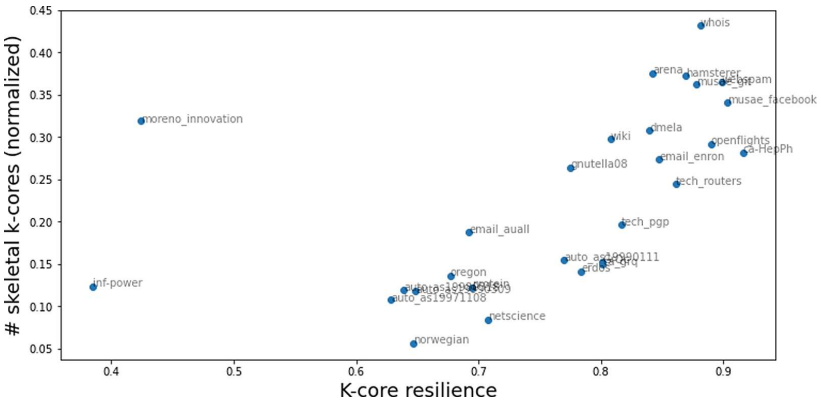

Fig. 2. Normalized number of skeletal cores vs Core Resilience

There is a very strong correlation between the two measures. The greater the density of skeletal cores, the higher core resilience is. This suggests a clear reason why certain networks have high core resilience: the core numbers of nodes are supported in many different ways. The only outlier is the *moreno_innovation* network, discussed below.

Relationships between different skeletal cores of the graph (e.g. overlap) can provide further insight into the network's robustness. One example can be found in Fig. 3. The *moreno_innovation* network has many edges that belong only to a *fraction* of skeletal cores of the network. In other words, there are many edges that can be useful to skeletal cores, but are not always necessary. This property explains the reason for an unusually high number of skeletal cores in this graph, as was seen in Fig. 2, because these edges might support a higher number of skeletal cores comparatively to other networks. This suggests a smaller overlap between different skeletal cores, and so when edges are randomly deleted destruction of many skeletal cores is more likely.

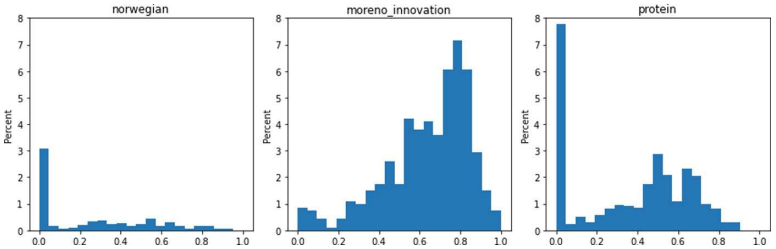

Fig. 3. Fraction of skeletal cores to which each edge belongs. Results were obtained by sampling 200 skeletal cores with MCMC. We ignore edges for which at least one endpoint has a core strength of 1, as they belong to all skeletal cores. *moreno_innovation* has a high normalized number of skeletal cores, but low core resilience; *protein* and *norwegian* networks have a low number of skeletal cores and low core resilience.

Estimating the Probability of an Edge Belonging to a Random Skeletal k-Core. The likelihood that an edge is part of an arbitrary skeletal core can be used for visualizations of skeletal cores or evaluation of the "centralization" of the core structure [16]. As our work is the first that can estimate this likelihood in an unbiased way, we compare against heuristics introduced by [16], which we denote as the *Centralized Skeletal Score (CSS)* heuristics.

In our proposed heuristics, we approximate the probability that an edge connected to a node $u \in V$ will be part of a random skeletal core. Denote the number of edges that can be removed from node u without dropping the k-core number of any node as:

$$\omega(u) = |\{v \in N(u) : K(v) > K(u) \vee (K(u) = K(v) \wedge CS[v] \neq 1)\}|. \quad (1)$$

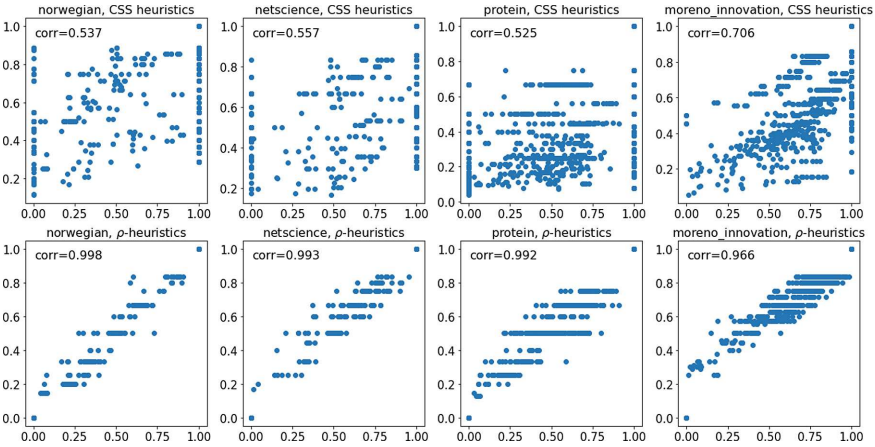

Fig. 4. Comparison of the heuristics. y-axis denotes the heuristics value for the edge, x-axis denotes the proportion of skeletal cores that the edge belongs to (MCMC sample). CSS-heuristics are shown on the top row, ρ-heuristics on the bottom. ρ-heuristics outperform the competition.

Similarly, define the number of edges that cannot be removed from u and must be present in every skeletal core:

$$\gamma(u) = |\{v \in N(u) : K(v) = K(u) \wedge CS[v] = 1\}|. \tag{2}$$

In a skeletal core, u will have at least $K(u)$ neighbours, so we need to select at least $K(u) - \gamma(u)$ nodes from $\omega(u)$ for the skeletal core. Thus, for node u:

$$\rho(u) = \begin{cases} max(\frac{K(u)-\gamma(u)}{\omega(u)}, 0) & \text{if } \omega(u) \neq 0 \\ 1 & \text{if } \omega(u) = 0 \end{cases} \tag{3}$$

For an edge $(u, v) \in E$ we define heuristics as:

$$\rho((u, v)) = \begin{cases} max(\rho(u), \rho(v)) & \text{if } K[u] = K[v] \\ \rho(u) & \text{if } K[u] < K[v] \\ \rho(v) & \text{if } K[u] > K[v] \end{cases} \tag{4}$$

We refer to these heuristics as "ρ-heuristics". In Fig. 4, heuristics from [16] and the proposed ρ-heuristics are plotted against the ground truth, as estimated with the MCMC algorithm. The correlation between ρ-heuristics and ground truth is significantly higher as compared to the earlier CSS heuristics.

5.4 Identifying the "Breaking Point" of the k-Core Structure

The normalized number of skeletal cores from Sect. 5.2 can be interpreted as the "density" of the skeletal cores amongst subgraphs obtained by removal of

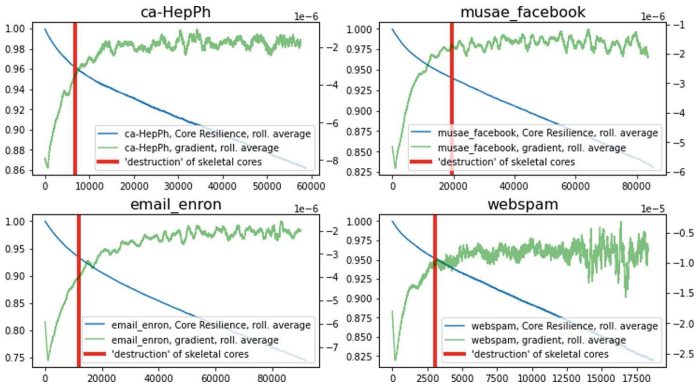

Fig. 5. Skeletal cores and "breaking point" of the network. x-axis denotes number of edges removed, left y-axis denotes the core resilience of the network for a certain percentage of removed edges, and the right y-axis denotes the gradient of Core Resilience. After the destruction of skeletal cores (red line), the gradient of Core Resilience (green line) flattens, indicating a fundamental loss of k-core structure. (Color figure online)

approximately $|E| - s_{exp}$ edges uniformly at random. If we remove edges past this point, the graph will have fewer edges than the expected size of skeletal cores, thus, edges will be unable to support the k-core.

In Fig. 5, we see that the average size of the skeletal core provides a good estimate of when the k-core structure disappears during a random edge removal process. Before the number of deleted edges equals the size of the average skeletal core (red line), core resilience (blue line) drops rapidly; but after that point, it shows a roughly linear decrease (as seen by the gradient, in green, flattening). One explanation is that prior to this point, a single edge deletion can cause a cascade in which many nodes drop their core number. After this point, the core structure is essentially destroyed, and such cascades of are unlikely.

6 Conclusion

In this paper, we introduced an MCMC algorithm for sampling skeletal cores uniformly at random as well as an algorithm for estimating the expected number of skeletal cores and their properties. We demonstrated the relationship between skeletal core structure and the core resilience of the graph, suggested a heuristic to estimate the likelihood of an edge being part of a skeletal core, and showed that skeletal cores can be used to find the "breaking point" of the k-core structure.

Acknowledgements. Honcharov and Soundarajan were supported by NSF Award #1908048. Sarıyüce was supported by NSF Award #1910063.

References

1. Adiga, A., Vullikanti, A.K.S.: How robust is the core of a network? In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8188, pp. 541–556. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40988-2_35
2. Altaf-Ul-Amine, M., et al.: Prediction of protein functions based on k-cores of protein-protein interaction networks and amino acid sequences. Genome Inform. **14**, 498–499 (2003)
3. Alvarez-Hamelin, J.I., Dall'Asta, L., Barrat, A., Vespignani, A.: k-core decomposition: a tool for the visualization of large scale networks. arXiv preprint cs/0504107 (2005)
4. Batagelj, V., Zaversnik, M.: An o(m) algorithm for cores decomposition of networks. arXiv preprint cs/0310049 (2003)
5. Bhawalkar, K., Kleinberg, J., Lewi, K., Roughgarden, T., Sharma, A.: Preventing unraveling in social networks: the anchored k-core problem. SIAM J. Discret. Math. **29**(3), 1452–1475 (2015)
6. Brooks, S.P., Roberts, G.O.: Assessing convergence of Markov chain Monte Carlo algorithms. Stat. Comput. **8**(4), 319–335 (1998)
7. Cohen, J.: Trusses: cohesive subgraphs for social network analysis. National security agency technical report 16.3.1 (2008)
8. Ellens, W., Kooij, R.E.: Graph measures and network robustness. arXiv preprint arXiv:1311.5064 (2013)
9. Erdős, P., Rényi, A., et al.: On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci **5**(1), 17–60 (1960)
10. Freitas, S., et al.: Graph vulnerability and robustness: a survey. IEEE Trans. Knowl. Data Eng. **35**(6), 5915–5934 (2022)
11. Al-garadi, M.A., Varathan, K.D., Ravana, S.D.: Identification of influential spreaders in online social networks using interaction weighted k-core decomposition method. Physica A: Stat. Mech. Appl. **468**, 278–288 (2017)
12. Govindan, P., Wang, C., Xu, C., Duan, H., Soundarajan, S.: The k-peak decomposition: mapping the global structure of graphs. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1441–1450 (2017)
13. Iwata, M., Sasa, S.: Dynamics of k-core percolation in a random graph. J. Phys. A Math. Theor. **42**(7), 075005 (2009)
14. Kendall, M.G.: A new measure of rank correlation. Biometrika **30**(1/2), 81–93 (1938)
15. Knuth, D.E.: Estimating the efficiency of backtrack programs. Math. Comput. **29**(129), 122–136 (1975)
16. Laishram, R., Soundarajan, S.: On finding and analyzing the backbone of the k-core structure of a graph. In: 2022 IEEE International Conference on Data Mining (ICDM), pp. 1017–1022. IEEE (2022)
17. Laishram, R., et al.: Measuring and improving the core resilience of networks. In: Proceedings of the 2018 World Wide Web Conference, pp. 609–618 (2018)
18. Luce, R.D., Perry, A.D.: A method of matrix analysis of group structure. Psychometrika **14**(2), 95–116 (1949)
19. Malvestio, I., Cardillo, A., Masuda, N.: Interplay between k-core and community structure in complex networks. Sci. Rep. **10**(1), 1–12 (2020)
20. Medya, S., Ma, T., Silva, A., Singh, A.: A game theoretic approach for core resilience. In: International Joint Conferences on Artificial Intelligence Organization (2020)

21. Mokken, R.J., et al.: Cliques, clubs and clans. Qual. Quant. **13**(2), 161–173 (1979)
22. Peng, C., Kolda, T.G., Pinar, A.: Accelerating community detection by using k-core subgraphs. arXiv preprint arXiv:1403.2226 (2014)
23. Pittel, B., Spencer, J., Wormald, N.: Sudden emergence of a giantk-core in a random graph. J. Comb. Theory Ser. B **67**(1), 111–151 (1996)
24. Seidman, S.B.: Network structure and minimum degree. Soc. Netw. **5**(3), 269–287 (1983)
25. Serafino, M., et al.: Superspreading k-cores at the center of COVID-19 pandemic persistence. medRxiv (2020)
26. Shin, K., Eliassi-Rad, T., Faloutsos, C.: Corescope: graph mining using k-core analysis–patterns, anomalies and algorithms. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 469–478. IEEE (2016)
27. Zhang, H., Zhao, H., Cai, W., Liu, J., Zhou, W.: Using the k-core decomposition to analyze the static structure of large-scale software systems. J. Supercomput. **53**, 352–369 (2010)