Measuring and Improving the Core Resilience of Networks

Ricky Laishram
Syracuse University
Syracuse, NY
rlaishra@syr.edu

Ahmet Erdem Sarıyüce
University at Buffalo
Buffalo, NY
erdem@buffalo.edu

Tina Eliassi-Rad
Northeastern University
Boston, MA
eliassi@ccs.neu.edu

Ali Pinar
Sandia National Laboratories
Albuquerque, NM
apinar@sandia.gov

Sucheta Soundarajan
Syracuse University
Syracuse, NY
susounda@syr.edu

ABSTRACT

The concept of k-cores is important for understanding the global structure of networks, as well as for identifying central or important nodes within a network. It is often valuable to understand the resilience of the k-cores of a network to attacks and dropped edges (i.e., damaged communications links).

We provide a formal definition of a network's core resilience, and examine the problem of characterizing core resilience in terms of the network's structural features: in particular, which structural properties cause a network to have high or low core resilience? To measure this, we introduce two novel node properties, *Core Strength* and *Core Influence*, which measure the resilience of individual nodes' core numbers and their influence on other nodes' core numbers. Using these properties, we propose the *Maximize Resilience of k-Core* (MRKC) algorithm to add edges to improve the core resilience of a network.

We consider two attack scenarios – randomly deleted edges and randomly deleted nodes. Through experiments on a variety of technological and infrastructure network datasets, we verify the efficacy of our node-based resilience measures at predicting the resilience of a network, and evaluate MRKC at the task of improving a network's core resilience. We find that on average, for edge deletion attacks, MRKC improves the resilience of a network by 11.1% over the original network, as compared to the best baseline method, which improves the resilience of a network by only 2%. For node deletion attacks, MRKC improves the core resilience of the original network by 19.7% on average, while the best baseline improves it by only 3%.

CCS CONCEPTS

• **Mathematics of computing → Graph theory**; *Graph algorithms*; *Approximation algorithms*; Paths and connectivity problems;

KEYWORDS

Graphs; Resilience; k-core;

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186127

ACM Reference Format:
Ricky Laishram, Ahmet Erdem Sarıyüce, Tina Eliassi-Rad, Ali Pinar, and Sucheta Soundarajan. 2018. Measuring and Improving the Core Resilience of Networks. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186127

1 INTRODUCTION

k-cores have emerged as an important concept for understanding the global structure of networks, as well as for identifying 'central' nodes within a network. The k-core [21] of a network is the maximal subgraph such that every node has at least k neighbors. The core number of a vertex is defined to be the largest k value such that there exists a k-core that contains the vertex, and nodes in the higher cores are considered to be more central within the network. k-cores represent cohesive subgroups of nodes, and have been used in a broad variety of important applications, such as studying the structure of internet networks [8], predicting the function of proteins [2], or understanding the evolution of networks.

There are many applications that depend on the core structure of a network (Section 3.1). Thus, it is valuable to understand the *resilience* of the network's core structure to the attacks where nodes and edges are deleted (i.e., damaged routers or communications links). *We define the (r, p)-core resilience of a network G, denoted by $\mathcal{R}_r^{\cdot(p)}(G)$, as the correlation between the core number rankings of the top r% nodes before and after p% edges or nodes are removed at random.* $\mathcal{R}_r^{\cdot(p)}(G)$ gives us rich information about the network: intuitively, it measures whether the core ordering of the top r% core number nodes in the network remains roughly the same even if that network is attacked. Because the core number is a measure of centrality, (r, p)-core resilience determines whether the most central nodes continue to be the most central nodes. We additionally present an aggregate (r, p_l, p_u)-*core resilience* measure ($\mathcal{R}_r^{\cdot(p_l,p_u)}(G)$), defined as the mean $\mathcal{R}_n^{\cdot(p)}(G)$ as we vary p from p_l to p_u.

We examine the problem of characterizing the core resilience in terms of the network's structural features: in particular, which structural properties cause a network to have high or low core resilience? To measure this, we introduce two novel node properties: *Core Strength* and *Core Influence*. We show that across real-world networks from a variety of domains, core strength and core influence are effective predictors of the core resilience of a network (Section 4.5). This allows designers or operators of infrastructure,

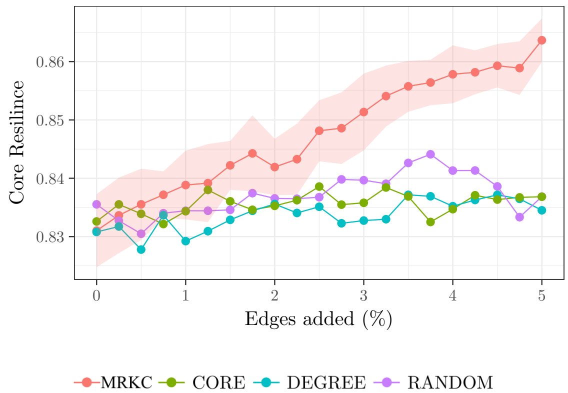


Figure 1: Adding edges to increase the resilience of the k-cores of the TECH_Router network against deleted edges. The MRKC method is shown in red, and outperforms the baseline methods.

computer, or other technological networks to evaluate the strength of the network before an attack occurs.

Based on these features, we propose an algorithm, called *Maximize Resilience of k-Core* (MRKC), to determine which edges should be added to a network to improve its resilience, under the constraint that the nodes' core numbers do not change. This has important applications in complex networks such as technological networks, where nodes can drop randomly and without warning, and we wish to improve the resilience of the network while preserving its overall core structure. We show that MRKC effectively inserts edges to make the network more resilient against such attacks. We observe that MRKC improves the core resilience by 11.1% and 19.7% against edge deletion and node deletion, respectively, whereas the best baseline can improve it by only 2% and 3%. Figure 1 presents the comparison of MRKC and baseline methods regarding the core resilience improvements against edge deletions in the TECH_Router network. We present our results in more details, and for other networks, in Section 5.3.

Our contributions can be summarized as follows:

(1) We propose the core resilience measure for characterizing the resilience of a network's core structure when nodes and edges are dropped.
(2) We introduce two simple node-based measures, core strength and core influence, to quickly and efficiently predict a network's core resilience.
(3) We present a novel algorithm for adding edges to a network to increase its core resilience, while keeping the core numbers unchanged.
(4) We perform experiments on a variety of real-world network datasets, and demonstrate that our algorithm outperforms a set of baseline methods at the task of increasing a network's core resilience.

2 RELATED WORK

In this section, we describe previous literature on core decomposition and evaluation of the k-core's resilience in real-world networks.

Core decomposition: Erdős and Hajnal [11] described the first k-core related concept in 1966, defining the degeneracy of the graph as the maximum core number of a vertex in the graph. Matula introduced the min-max theorem [17] for the same concept, but in the context of graph coloring. Roughly simultaneously, Seidman [21] and Matula and Beck [16] defined the k-core subgraph as the maximal connected subgraph where each vertex has at least degree k. Seidman stated that k-cores are good *seedbeds* that can be used to find further dense substructures, but did not provide a principled algorithm for finding k-cores [21]. Matula and Beck [16], on the other hand, give algorithms for finding the core numbers of vertices, and also finding all the k-cores of a graph (and their hierarchy) by using these core numbers, since there can be multiple k-cores for the same k value.

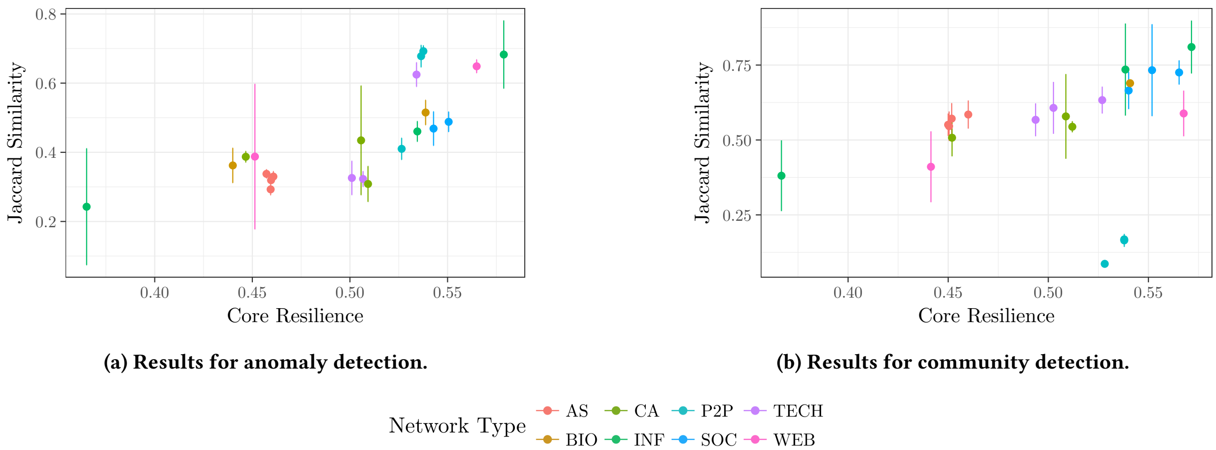
Batagelj and Zaversnik introduced an efficient implementation that uses the bucket data structure to find the core numbers of vertices [6]. In contrast to previous work [16, 21], they defined the k-core as a possibly disconnected subgraph. Core decomposition has attracted a great deal of interest in the recent years, finding use in applications such as visualization [3] and analysis of the internet topology [5]. Thanks to the the practical benefit and linear complexity of the k-core decomposition, there has been a great deal of recent work in adapting k-core algorithms for different data types or setups. Cheng *et al.* [9] introduced the first external-memory algorithm, and Wen *et al.* [23] and Khaouid *et al.* [15] provided further improvements in this direction. Giatsidis *et al.* adapted the k-core decomposition for weighted [13] and directed [12] graphs.

To handle the dynamic nature of the real-world data, Sariyuce *et al.* [20] introduced the first streaming algorithms to maintain the k-core decomposition of a graph upon edge insertions and removals. Motivated by the incomplete and uncertain nature of the real network data, O'Brien and Sullivan [18] proposed new methods to locally estimate core numbers (K values) of vertices when the entire graph is not known, and Bonchi *et al.* [7] showed how to efficiently perform the k-core decomposition on uncertain graphs, which has existence probabilities on the edges.

Core resilience: There are only a few works that study the sensitivity of the core decomposition. Most closely related to our work is the study by Adiga and Vullikanti, investigating the robustness of the top cores under sampling and in noisy networks [1]. They reported that the success in recovering the top cores under sampling and noise exhibits non-monotonic behavior with the amount of samples and noise. Another related study is by Zhang *et al.* [24], who recently proposed the collapsed k-core problem to find the critical vertices. For a given k value and a budget b, they introduced algorithms to delete b (critical) vertices to get the smallest k-core (in size). In our work, we follow a more general approach and quantify the resilience of the core numbers, and the impact of the neighbor vertices on the stability. In addition, we propose edge insertion heuristics to strengthen the core numbers while preserving the existing core decomposition.

3 CORE RESILIENCE

In many network applications, we may encounter the problem of deleted edges or nodes. For example, in technological networks, edges may be lost due to dropped communication links, and in router networks, nodes might drop due to routers being turned off. It is thus valuable to understand the resilience of the k-core of the



(a) Results for anomaly detection.

(b) Results for community detection.

Figure 2: Similarity between anomalies (Figure 2a) and communities (Figure 2b) found in the full network G and the sample G' for different real-world networks. The x-axis is the Core Resilience ($\mathcal{R}_{50}^{n(0,50)}(G)$) of the different networks against node deletion, and the y-axis is the Jaccard Similarity. As expected, in the networks with high Core Resilience, the results on the sample is more similar to that on the full network in general.

network to missing edges and nodes. In this section, we introduce the concept of *core resilience*, which quantifies the degree to which a network's core structure changes when nodes or edges are deleted uniformly at random.

We define the (r,p)-core resilience of a network G as the rank correlation between the top r% nodes (as ranked by core number) in the original network to that of the network after p% of the the edges or nodes have been removed uniformly at random. We denote the (r,p)-core resilience of a graph G to edge deletion by $\mathcal{R}_r^{e(p)}(G)$, and that due to node deletion by $\mathcal{R}_r^{n(p)}(G)$. We will use $\mathcal{R}_r^{\cdot(p)}$ to refer to (r,p)-core resilience in general. Let $G = \langle V, E\rangle$ be a network, and let G^p represent the network obtained removing p% of the edges (or nodes) from G randomly. Let the top r% nodes (by core numbers) in G be denoted by V_r. Define a set M_r^p such that,

$$M_r^p = \left\{\left(K(u,G), K(u,G^p)\right) : u \in V_r\right\}$$

where $K(u,G)$ is the core number of node u in network G. (If a node has been deleted, its core number in G^p is 0.)

Then, the (r,p)-core resilience of G is given by,

$$\mathcal{R}_r^{\cdot(p)}(G) = \tau_b\left(M_r^p\right) \tag{1}$$

where $\tau_b(\cdot)$ is the Kendall's tau-b rank correlation. (We can replace $\tau_b(\cdot)$ by any other measures of rank correlation.)

While $\mathcal{R}_r^{\cdot(p)}$ gives rich, detailed insight into the core resilience of the different cores of the network at different levels of edges or nodes deletion, in some applications it may be preferable to use a simpler measure. We thus define an aggregate measure, the (r,p_l,p_u)-*core resilience. We define the (r,p_l,p_u)-core resilience of a network as the mean (r,p)-core resilience as we vary p from p_l to p_u.* We denote the (r,p_l,p_u)-core resilience of G by $\mathcal{R}_r^{\cdot(p_l,p_u)}(G)$.

$$\mathcal{R}_r^{\cdot(p_l,p_u)}(G) = \frac{\int_{p_l}^{p_u}\mathcal{R}_r^{\cdot(x)}(G)dx}{p_u - p_l} \tag{2}$$

In practice, we approximate the integral in Equation 2 by a summation with step size 1.

It should be noted that there are a number of graph robustness measures, but the concept of core resilience specifically concerns the k-core structure of the network, and so is not directly related to these existing measures. To verify this we compared the *Natural Connectivity* [14] to the Core Resilience of various real-world networks, and did not observe any significant correlation. Due to space limitations, we do not include these results.

Because it is not always practical to compute the core resilience by Equation 1, it is of great practical interest to determine whether a network will have high or low resilience based on its structural features. In Section 4, we thus address the problem of characterizing the core resilience of a network in terms of quickly-computed structural properties.

3.1 Motivating Applications

The concept of Core Resilience is helpful in applications where the k-core structure of the network under missing edges or nodes is important. In this section we will discuss two such applications, anomaly detection and community detection, which use k-cores on sampled data.

Assume that we have a network $G = \langle V, E\rangle$ and a subgraph $G' = \langle V', E'\rangle$, where G' is the result of random walk on G.

If we perform anomaly detection [22] or community detection [19] on G', how well do the results on G' reflect the true anomalies and communities in G? Because these applications make use of the k-core structures, we expect the results to more closely match that of the original graph if the original graph has high core resilience.

We verify this experimentally on multiple real-world networks, and the sample we use is generated by a random walk with half the number of nodes in the network as the budget.

3.1.1 Anomaly Detection. In this application, we perform anomaly detection on the full network G using the *CORE-A* method proposed in [22] to find the anomalous nodes V_α. This method operates on the intuition that nodes with high core numbers also have high degrees. So for a given node, the difference between the ranking in terms of the degree and core number (referred to as *dmp* in [22]) should be fairly small. However, anomalous nodes (for example, someone in a social network who paid to get more followers) deviate significantly from this pattern. By looking at the *dmp* values of the nodes, the anomalies are identified in the CORE-A algorithm.

We find anomalies in the subgraph G' with the same method, and refer to the set of these anomalies as V'_α. We then use Jaccard Similarity to determine how close the result on G' is to that on G.

$$J_\alpha(V_\alpha, V'_\alpha) = \frac{|V_\alpha \cap V'_\alpha|}{|(V_\alpha \cap V') \cup V'_\alpha|}$$

We present results in Figure 2a. We can observe that the anomalies found in the sample are more similar to those in the full network for networks with high core resilience.

3.1.2 Community Detection. By finding a central region of the network, k-cores can be used to accelerate community detection. We perform community detection using the method proposed in [19] and the Louvain method on the original network G. We denote the communities in G by C. Then, we perform community detection with the same method on G', to get the communities C'.

We compute the similarity between C and C' as the mean Jaccard Similarity between the communities in C' to its best match community in C.

$$J_c(C, C') = \frac{1}{|C'|} \sum_{c \in C'} \frac{|c \cap \beta(c, C)|}{|c \cup (\beta(c, C) \cap V')\,|}$$

where $\beta(x, Y)$ is a function that maps the community x to another community y such that $|x \cap y|$, and there are no other $x' \in X$ that maps to y.

Figure 2b shows the results of these experiments on community detection. In the networks with higher Core Resilience, the nodes that are grouped together in the same community in the sample are more frequently grouped together in the original communities as well. The only exceptions to this are two P2P networks, for which the similarity is low even though they have relatively high core resilience. This is because there are very few communities in the original network, but only a single, giant community. So, $\beta(c, C) = \emptyset$ for most $c \in C'$.

These two applications demonstrate that if we know the Core Resilience of a network, we can use it as an indicator of how much we should expect core-based observations on incomplete data to reflect those on the original.

4 CHARACTERIZING CORE RESILIENCE

Directly computing the (r, p)-core resilience of a network is not practical in many cases, as it requires repeated k-core decomposition. It is thus valuable to characterize the core resilience of the network without directly computing the (n, p)-core resilience (and, as we will see, this characterization allows us to develop an effective algorithm for improving a network's core resilience).

In this section, we propose two node properties based on a network's structure: *Core Strength* and *Core Influence*. The core strength of a node is a measure of how likely its core number will decrease when edges are deleted from the network. The core influence of a node is a measure of the extent to which nodes with lower core numbers depend on that node for their own core numbers. In Sections 4.3 and 4.2, we describe the core influence and core strength properties in more details.

We also define an overall network property, based on the core strength and core influence of the nodes in the network. We describe this in more detail in Section 4.4. We perform experiments on real world networks of various types to show the relationship between these measures and the core resilience of the network.

4.1 Notation

Before describing the Core Influence and Core Strength properties, we first introduce our notations. Let $K(u, G)$ and $\Gamma(u, G)$ represent the core number and set of neighbors of u in G, respectively. We split the neighbors of u into three sets $\Delta_<(u, G)$, $\Delta_=(u, G)$ and $\Delta_>(u, G)$ representing, respectively the neighbors of u with core number less than, equal to, and greater than that of u.

$$\Delta_<(u, G) = \{v : v \in \Gamma(u, G) \land K(v, G) < K(u, G)\}$$
$$\Delta_=(u, G) = \{v : v \in \Gamma(u, G) \land K(v, G) = K(u, G)\}$$
$$\Delta_>(u, G) = \{v : v \in \Gamma(u, G) \land K(v, G) > K(u, G)\}$$
$$\Delta_\geq(u, G) = \Delta_=(u, G) \cup \Delta_>(u, G)$$

We also define a set V_δ of nodes where each node $u \in V_\delta$ has at least one neighbor node, v, with a larger core number, i.e., $K(u, G) < K(v, G)$. That also means the following:

$$V_\delta = \{u : u \in V \land |\Delta_=(u, G)| < K(u, G)\}.$$

4.2 Core Strength

The Core Strength of node u is the minimum number of u's neighbors that need to be disconnected in order for u's core number to decrease. We denote the core strength of u in G by $CS(u, G)$.

For all nodes u in network G, u gets its core number due to connections to $\Delta_\geq(u, G)$. Thus, the Core Strength of node $u \in G$ is given by,

$$CS(u, G) = |\Delta_\geq(u, G)| - K(u, G) + 1. \tag{3}$$

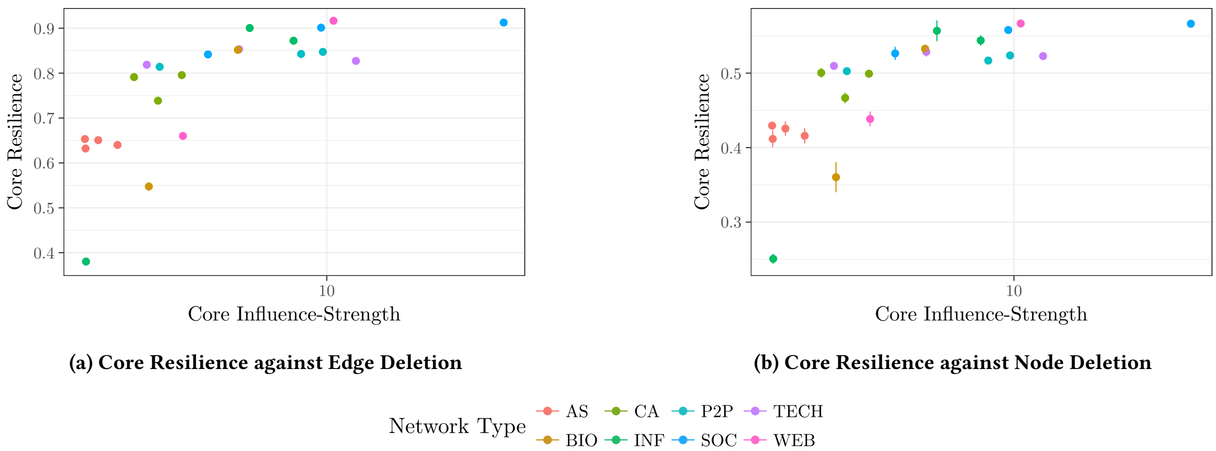
Intuitively, the Core Strength of a node u describes how likely it is to retain its core number when it loses connections. A node with a high core strength has many redundant connections (i.e., many connections to other nodes with equal or higher core number), and so is less likely to drop its core number if its connections are deleted.

Running Time: Given a network $G = \langle V, E \rangle$, computing the Core Strength of all the nodes is possible once the k-core decomposition is performed, which takes $O(|E|)$ time. For each node we need to count the number of neighbors with greater or equal core number, which is also linear in the number of edges, $O(|E|)$. So, the time complexity of computing the core strength of all nodes is $O(|E|)$.

4.3 Core Influence

The Core Influence of a node u in network G is a measure of the extent to which u affects the core numbers of neighbor nodes with lower core numbers.

For a node u, the set of nodes that depend on u for their core numbers is $V_\delta \cap \Delta_<(u, G)$. Consider two nodes $v_0, v_1 \in V_\delta \cap \Delta_<(u, G)$,



(a) Core Resilience against Edge Deletion

(b) Core Resilience against Node Deletion

Figure 3: Core Resilience ($\mathcal{R}_{100}^{\cdot(0,50)}(G)$) against Core Influence-Strength ($CIS_{95}(G)$) for various networks. Figure 3a shows the core resilience against edge deletion vs Core Influence-Strength, and Figure 3b shows the core resilience against node deletion vs Core Influence-Strength. We can observe that the Core Resilience is higher for networks with higher Core Influence-Strength, which is consistent with what we expect.

where v_0 and v_1 need at least m'_0 and m'_1 nodes with higher core numbers for their core numbers respectively, and $\frac{m'_0}{K(v_0,G)} < \frac{m'_1}{K(v_1,G)}$. Then v_1 depends more strongly on $\Delta_>(v_1, G)$, than v_0 on $\Delta_>(v_0, G)$. To account for this, v_1 needs to contribute a greater fraction of its core influence to $\Delta_>(v_1, G)$.

Thus, for $v \in V$, we introduce a weight $\delta(v, G)$ such that v contributes $\delta(v, G) \cdot CI(v, G)$ to $\Delta_>(v_1, G)$.

$$\delta(v, G) = 1 - \frac{|\Delta_=(v, G)|}{K(v, G)}$$

Consider the nodes v_0 and v_1 again, and assume that they have m_0 and m_1 neighbors with higher core numbers, and $m_0 < m_1$. Then, the dependence of v_0 on u is stronger than that of v_1 on u. So, to account for this, we equally divide the CI contribution of a node v equally between all nodes in $\Delta_>(v, G)$.

Now, we mathematically define the core influence of u as

$$CI(u, G) = \sum_{v \in V_\delta \cap \Delta_<(u,G)} \frac{\delta(v, G) \cdot CI(v, G)}{|\Delta_>(v, G)|}. \quad (4)$$

To compute the core influence of all the nodes in G, we initialize all values to 1. (Any positive number can be used.) We then start computing the core influence of the nodes with minimum core number, and proceed till we reach the nodes with maximum core number. Because the core influence of a node is only influenced by that of nodes with lower core numbers, we need only one iteration to compute the core influence of all nodes.[1]

Running Time: To compute the core influence of all nodes in $G = \langle V, E \rangle$, we need to perform k-core decomposition first ($O(|E|)$). Then we need to find $\Delta_=(u, G)$, $\Delta_>(u, G)$ and $\Delta_<(u, G)$ for all nodes u. This can be performed in $O(|E|)$. Then we find the set V_δ in $O(|V|)$. We can assign the core influences of all the nodes (with Equation 4) in $O(|V|)$. So, the overall computation takes $O(|E|)$.

[1]Core influence can also be defined to consider the nodes with equal core numbers, in addition to lower. However, we found that the overall results were similar for both definitions and one iteration is enough for the formulation with lower core numbers.

4.4 Core Influence-Strength

Core Strength and Core Influence describe node level properties. To characterize the network, we need an aggregate measure.

Assume that $CI_f(G)$ is the f percentile of core influence of all nodes in G. Let $S_f(G)$ be the set of nodes in G with core influence equal to or greater than $CI_f(G)$.

$$S_f(G) = \{u : u \in V \wedge CI(u, G) \geq CI_f(G)\}]$$

Then we define the *Core Influence-Strength* as the mean core strength of $S_f(G)$. We denote it by $CIS_f(G)$,

$$CIS_f(G) = \frac{\sum_{u \in S_f(G)} CS(u, G)}{|S_f(G)|}. \quad (5)$$

If a network has high $CIS_f(G)$ for high f, this means that the most influential nodes are unlikely to drop their core number when they lose connections to their neighbors. We expect such networks to have high core resilience. In contrast, the networks for which $CIS_f(G)$ is low are expected to have low core resilience.

4.5 Experiments

To verify that CIS reflects actual core resilience, we perform experiments on 22 real-world networks of different types (Table 1). These networks were downloaded from SNAP[2] and Network Repository[3]. The Core Resilience ($\mathcal{R}_{100}^{\cdot(0,50)}(G)$) vs Core Influence-Strength ($CIS_{95}(G)$) for edge deletion is shown in Figure 3a, and that for node deletion is shown in Figure 3b.

In these figures, each point is the core resilience of a network (with the network type color-coded), and is the result of 10 experiments. We observe that, as expected, the resilience is higher for networks with high Core Influence-Strength. However the relation between Core Influence-Strength and Core Resilience is sub-linear - that is it increases rapidly for low values, but for networks high Core Influence-Strength the difference in Core Resilience is not

[2]https://snap.stanford.edu/
[3]http://networkrepository.com/

Type	Network	$\|V\|$	$\|E\|$	k_{max}
AS	AS_733_19971108†	3015	5196	9
	AS_733_19990309†	4759	8896	12
	Oregon1_010331†	10670	22002	17
	Oregon1_010428†	10886	22493	17
BIO	BIO_Dmela‡	7393	25569	11
	BIO_Yeast_Protein‡	1846	2203	5
CA	CA_GrQc†	5241	14484	43
	CA_HepTh†	9875	25973	31
	CA_Erdos992‡	5094	7515	7
INF	INF_OpenFlights‡	2939	15677	28
	INF_Power‡	4941	6594	5
P2P	P2P_Gnutella08†	6301	20777	10
	P2P_Gnutella09†	8114	26013	10
	P2P_Gnutella25†	22687	54705	5
SOC	SOC_Hamsterster‡	2426	16630	4
	SOC_Advogato‡	5167	39432	5
	SOC_Wiki_Vote‡	889	2914	9
TECH	TECH_Pgp‡	10680	24316	31
	TECH_Routers_rf‡	2113	6632	15
	TECH_WHOIS‡	7476	56943	88
WEB	WEB_Spam‡	4767	37375	35
	WEB_Webbase‡	16062	25593	32

Table 1: In this table, $|V|$ is the number of nodes, $|E|$ is the number of edges, and k_{max} is the degeneracy. These datasets were downloaded from SNAP (denoted by †), and Network Repository (denoted by ‡).

significant. Additionally we observe that the Core Resilience of P2P networks generally have lower Core Resiliences, while that of SOC networks tend to be higher in terms of both edge and node deletion.

5 IMPROVING CORE RESILIENCE

In many types of networks (such as technological networks), edges or nodes might drop randomly. We may thus be interested in adding a fixed number of edges to improve the core resilience of the network, in order to ensure that the network will retain its basic core structure even if nodes or edges are lost.

A simple way to accomplish this would be to add edges so as to increase gaps between k-shells (i.e., by adding intra-shell edges, beginning with the highest shells). However, this would change the distribution of core numbers, which is an important property of the network [4, 10]. We thus add an additional constraint that the core numbers of the nodes should not change. Formally, we consider the following problem:

Given an undirected, unweighted network G and an edge budget b, which b edges should we add to G so that the core resilience of the modified network G' is as high as possible and core numbers are not changed?

5.1 Edge Deletion and Node Deletion

We define the core resilience under two scenarios in which the ranking of the nodes by core number might change: edge deletion and node deletion. Note that node deletion can be treated as a special type of edge deletion, as when a node is deleted, all of its edges are deleted. In this section, we show the relationship between core resilience due to edge deletion and that due to node deletion.

Consider, a graph $G = \langle V, E\rangle$. The (r, p)-core resilience of G is given by $\mathcal{R}_r^{n(p)}(G)$ and $\mathcal{R}_r^{e(p)}(G)$ (by definition) for node deletion and edge deletion, respectively.

Assume that deletion of p nodes results in deletion of p' edges. It is reasonable to assume $p' > p$, since real-world networks rarely have an average degree of one. That is, $\mathcal{R}_r^{e(p')}(G) \approx \mathcal{R}_r^{n(p)}(G)$, and in general $\mathcal{R}_r^{e(p)}(G) \geq \mathcal{R}_r^{e(p')}(G)$. So, $\mathcal{R}_r^{n(p)}(G) \leq \mathcal{R}_r^{e(p)}(G)$.

Now let us consider the (r, p_l, p_u)-core resilience under edge deletion and node deletion.

$$\mathcal{R}_r^{n(p_l,p_u)}(G) - \mathcal{R}_r^{e(p_l,p_u)}(G) = \frac{\int_{p_l}^{p_u}\left(\mathcal{R}_r^{n(x)}(G) - \mathcal{R}_r^{e(x)}(G)\right)dx}{p_u - p_l}$$
$$\mathcal{R}_r^{n(p_l,p_u)}(G) \leq \mathcal{R}_r^{e(p_l,p_u)}(G) \qquad (6)$$

5.2 Proposed Method: MRKC

In this section we address the problem of improving the core resilience of a network by adding a fixed number of edges. Our initial results in Section 4 suggest that edges should be added to bolster the nodes with high Core Influence; i.e., give them higher Core Strength. We propose a new algorithm called *Maximize Resilience of k-core* (MRKC).

Node deletion can be considered a special case of edge deletion, as deleting a node is equivalent to deleting the edges of that node. For this reason, the algorithm for improving the core resilience of a network against edge deletion is the same as for node deletion.

The MRKC algorithm consists of two steps: Generating Candidate Edges and Assigning Edge Priority. We discuss these steps in detail in Sections 5.2.1 and 5.2.2, respectively.

5.2.1 Generating Candidate Edges. Given a network $G = \langle V, E\rangle$, the first step in MRKC is to determine which edges can be added to the network without changing the core number of any node. Let E' be the set of edges that do not exist in G. The size of E' is on the order of $|V|^2$. This is clearly too many edges to check, so we need a method to quickly filter out the edges that would change the core number if added to G.

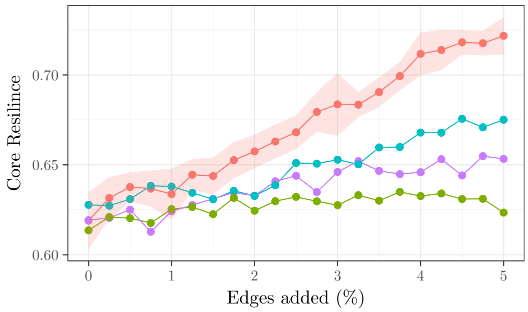
MRKC accomplishes this by adapting the purecore-based method described in [20], which examines the endpoint of each potential edge (the *purecore* of a node u is the set of nodes that have the same core number as u and could be affected by a change in the core number of u).

Let us denote the purecore of node g in graph G by $PC(u, G)$. We split E' into two sets E_{sim} and E_{dif}, such that, $K(u, G) = K(v, G)$ for all $(u, v) \in E_{sim}$; and $K(u, G) \neq K(v, G)$ for all $(u, v) \in E_{dif}$.

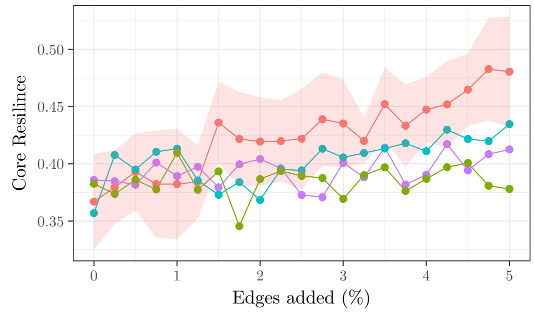
From the set E_{sim}, we generate subsets E_{sim}^i such that:

- $\bigcup E_{sim}^i \equiv E_{sim}$; i.e. is all edges in E_{sim} are in some E_{sim}^i.
- $E_{sim}^i \cap E_{dif}^{j \neq i} \equiv \emptyset$; i.e. all E_{sim}^i are disjoint.
- No two edges in E_{sim}^i are connected via the nodes that have same core number with the endpoints of those edges.

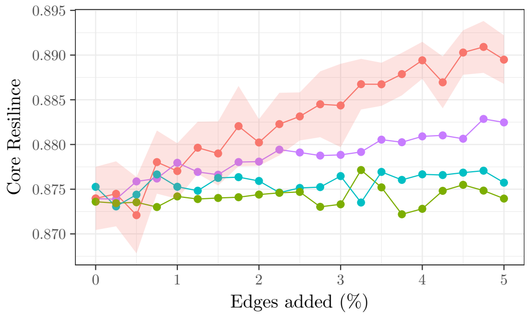
Because all the edges have endpoints that are not in the other's purecore, we can insert E' to G, and if there is a node that changes core number, we can pinpoint which edge in E' caused it. Assume that there are n_{sim} such subsets.



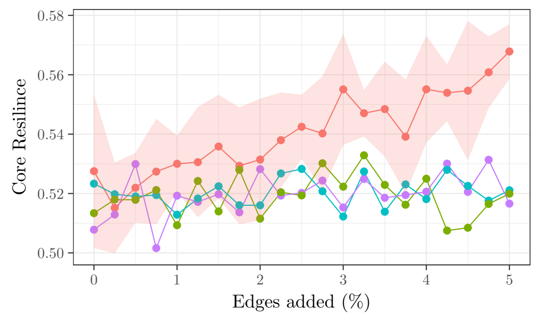
(a) AS_733_1999 (Edge Deletion)



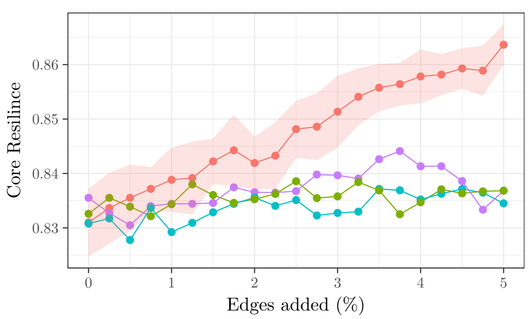
(b) AS_733_1999 (Node Deletion)



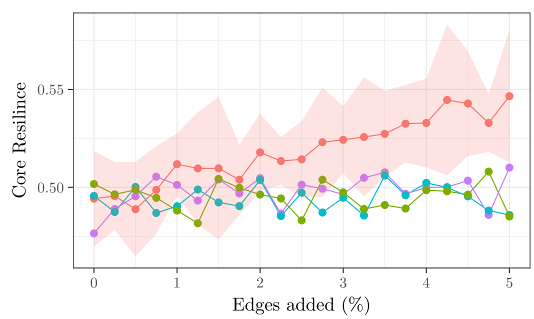
(c) INF_OpenFlights (Edge Deletion)



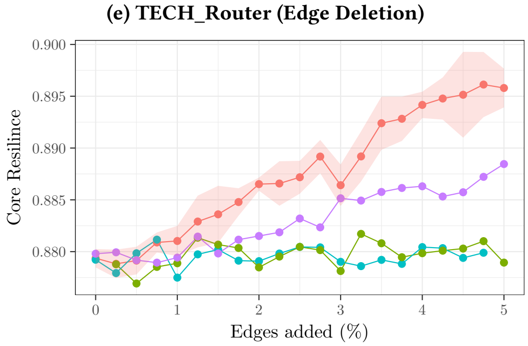
(d) INF_OpenFlights (Node Deletion)



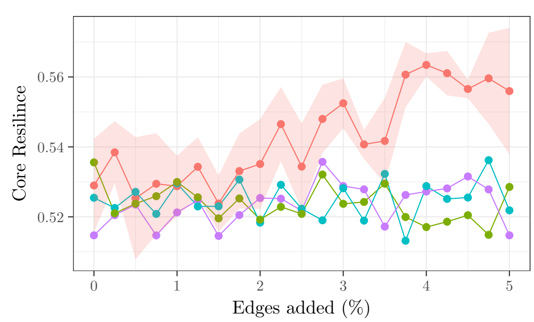
(e) TECH_Router (Edge Deletion)



(f) TECH_Router (Node Deletion)



(g) WEB_Spam (Edge Deletion)



(h) WEB_Spam (Node Deletion)

Figure 4: Change in Core Resilience against percentage of new edges added for different real-world networks. The y-axis is the core resilience and the x-axis is the percentage of new nodes added by the different algorithms. The figures in the left column (Figures 4a,4c,4e,4g) are for edge deletion, and those in the right column (Figure 4b,4d,4f,4h) are for node deletion. In all cases, `MRKC` outperforms the baselines.

Similarly, we split E_{dif} into subsets E^i_{dif} in the same way as E_{sim}, but with additional conditions that if there are two edges in E^i_{dif} that have the same endpoints, the other two nodes cannot have the same core numbers.

Again in this case if on adding E^i_{dif} to G, the core number of any node changes, we can identify which edge in E^i_{dif} caused that. Let us assume that there are n_{dif} such subsets.

Then, instead of checking all $|E'|$ edges one-by-one, we need to check only $n_{sim} + n_{dif}$ times.

We can further speed up the generation of the candidate edges. Assume that $E^i_.$ is the set of nodes currently being tested. Let k_{min} and k_{max} be the minimum and maximum core number of the nodes involved in $E^i_.$. Then, adding the $E^i_.$ can only change the core numbers of nodes u where $k_{min} \geq K(u, G) \geq k_{max}$.

So, instead of running k-core decomposition on the entire network after adding the edges, we can add the edges to the k_{min}-core subgraph of the original network, and run the k-core decomposition on the subgraph. Again because, no node with core number above k_{max} will be affected, we do not need to run the k-core decomposition to completion - we can stop after the k_{max}-core has been found.

5.2.2 Assigning Edge Priority. Once it obtains the set of edges that can be added to the network, MRKC selects which subset of edges to add. To do this, MRKC assigns each edge $(u, v) \in E'$ a priority based its endpoints u and v. As discussed before, the goal is to improve the core strength of the nodes with high core influence. So the priority value for each node u is assigned as $\frac{CI(u)}{CS(u)}$.

There are three cases that needs to be considered based on the core numbers of u and v: (a) $K(u, G) > K(v, G)$, (b) $K(u, G) < K(v, G)$, and (c) $K(u, G) = K(v, G)$.

In the case of $K(u, G) > K(v, G)$, addition of the edge (u, v) will only affect $CI(v, G)$; $CI(u, G)$ will be unaffected. On the other hand, if $K(u, G) = K(v, G)$, both $CI(u, G)$ and $CI(v, G)$ will be affected by addition of (u, v). So, for all $(u, v) \in E'$, MRKC assigns priority as,

$$\rho(u, v) = \begin{cases} \frac{CI(u,G)}{CS(u,G)} & \text{if } K(u, G) < K(v, G) \\ \frac{CI(v,G)}{CS(v,G)} & \text{if } K(u, G) > K(v, G) \\ \frac{CI(u,G)}{CS(u,G)} + \frac{CI(v,G)}{CS(v,G)} & \text{if } K(u, G) = K(v, G) \end{cases} . \quad (7)$$

At each step, MRKC selects the edge with the highest priority and adds it to the network until we reach the budget, i.e., maximum number of edges allowed to be added. The set E' needs to be updated after any edge (u, v) is inserted, but we can make it efficient by checking only for those edges that has an endpoint in $PC(u, G) \cup PC(v, G)$. Updates to core influence and core strength can also be done in similar way.

5.3 Experiments

To evaluate MRKC, we added up to 5% new edges to real-world networks to improve their core resilience.

The networks we used for our experiments are given in Table 2. As mentioned in Section 5.2, adding edges to improve core resilience is applicable to only some type of networks. For example, in social networks, we cannot force people to form connections. However, we included these kind of networks in our experiments for the sake of completeness.

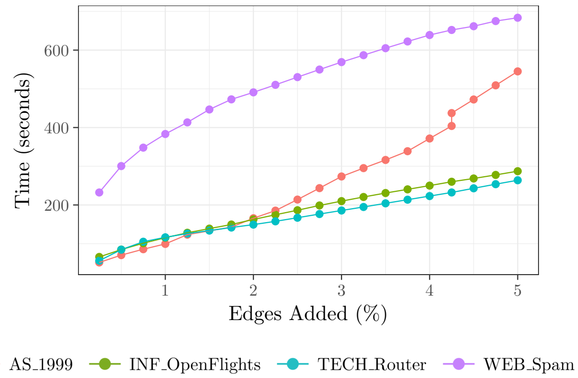


Figure 5: Running time of our method for improving core resilience (MRKC) on different networks. The x-axis is the amount of new edges added (in %), and the y-axis is the time taken to add the edges (in seconds).

For comparison, we consider three baseline methods where the edges in E' are added (1) randomly (RANDOM), (2) in decreasing order of the sum of the degrees of the endpoints (DEGREE), and (3) in decreasing order of the sum of the core numbers of the endpoints (CORE). We run each experiment 10 times, and present the mean values. In Figure 4, we show the comparison of the core resiliences of different networks with edges added by MRKC and the three baselines. The y-axis is the core resilience, and the x-axis is the percentage of edges added. Because of space limitations, we cannot present the plots for all the networks, and so we give them in Table 2 when 5% new edges are added.

We observe that MRKC outperforms all considered baseline methods. In cases where the initial core resilience is low, MRKC can improve it by a large amount (for example in INF_Power, BIO_Yeast). However, if a network already has high core resilience to begin with, MRKC cannot improve it by much (as in INF_OpenFlights, TECH_Whois).

In the case of AS networks, the core resilience (with respect to both edge deletion and node deletion) is low, and after adding the edges by MRKC, the core resilience is increased significantly - up to 17.9% and 25.7% for edge deletion and node deletion respectively. However, for the TECH networks, the core resilience against edge deletion is already high. So on adding edges by MRKC, we could achieve an improvement of only 3.4%.

In the plots shown in Figure 4, we observe that the rate of improvement of MRKC in the case of node deletion is lower than that for edge deletion in the same network. This is because the core resilience due to edge deletion cannot be less than that of node deletion (Equation 6).

Running Time: In Figure 5, we show the time taken to add the new edges according to our method for four networks. The x-axis is the amount of new edges added (in%), and the y-axis is the time taken to add the edges. The values are the means over 10 runs.

MRKC checks for all edges that can be added without changing core number in the first step. This is why we observe in Figure 5 that the plots do not start at the same points. After the initial candidate edges generation, we no longer need to check all the edges - if an

		Edge Deletion ($\mathcal{R}_{50}^{e(0,50)}$)					Node Deletion ($\mathcal{R}_{50}^{n(0,25)}$)				
Type	Network	Original	MRKC	RANDOM	DEGREE	CORE	Original	MRKC	RANDOM	DEGREE	CORE
AS	AS_733_19971108	0.58	**0.65**	0.60	0.61	0.58	0.35	**0.44**	0.40	0.38	0.36
	AS_733_19990309	0.62	**0.72**	0.65	0.67	0.62	0.36	**0.48**	0.41	0.43	0.37
	Oregon1_010331	0.66	**0.78**	0.71	0.72	0.72	0.42	**0.49**	0.45	0.44	0.45
	Oregon1_110428	0.67	**0.79**	0.72	0.72	0.71	0.41	**0.50**	0.46	0.42	0.44
BIO	BIO_Dmela	0.80	**0.84**	0.82	0.83	0.83	0.48	**0.55**	0.49	0.49	0.48
	BIO_Yeast_Protein	0.49	**0.71**	0.55	0.57	0.56	0.34	**0.47**	0.38	0.37	0.37
CA	CA_GrQc	0.75	**0.81**	0.74	0.76	0.74	0.43	**0.51**	0.43	0.42	0.42
	CA_HepTh	0.69	**0.78**	0.71	0.70	0.72	0.40	**0.45**	0.38	0.40	0.41
	CA_Erdos992	0.69	**0.72**	0.70	0.69	0.71	0.44	**0.49**	0.42	0.43	0.43
INF	INF_OpenFlights	0.87	**0.89**	0.88	0.87	0.87	0.51	**0.57**	0.51	0.52	0.51
	INF_Power	0.49	**0.77**	0.36	0.42	0.38	0.29	**0.46**	0.26	0.25	0.27
P2P	P2P_Gnutella08	0.73	**0.79**	0.72	0.75	0.73	0.40	**0.51**	0.43	0.45	0.43
	P2P_Gnutella09	0.71	**0.78**	0.73	0.72	0.73	0.39	**0.50**	0.42	0.45	0.43
	P2P_Gnutella25	0.69	**0.81**	0.71	0.73	0.74	0.39	**0.47**	0.41	0.40	0.41
SOC	SOC_Hamster	0.84	**0.86**	0.85	0.85	0.85	0.50	**0.54**	0.52	0.52	0.50
	SOC_Wiki_Vote	0.76	**0.82**	0.75	0.77	0.77	0.43	**0.51**	0.45	0.45	0.47
	SOC_Advogato	0.88	**0.91**	0.89	0.88	0.89	0.52	**0.61**	0.52	0.50	0.51
TECH	TECH_Ppg	0.81	**0.86**	0.81	0.81	0.82	0.47	**0.53**	0.49	0.50	0.51
	TECH_Router_rf	0.83	**0.86**	0.83	0.83	0.83	0.49	**0.55**	0.51	0.48	0.48
	TECH_Whois	0.89	**0.91**	0.89	0.89	0.89	0.52	**0.65**	0.57	0.59	0.59
WEB	WEB_Spam	0.87	**0.90**	0.88	0.87	0.87	0.51	**0.56**	0.51	0.52	0.52
	WEB_Webbase	0.61	**0.75**	0.60	0.59	0.60	0.38	**0.45**	0.42	0.43	0.44

Table 2: Improvement (in %) in Core Resilience of the top 50% nodes (by core number) on adding 5% new nodes by MRKC, random (RANDOM), highest mean degree (DEGREE) and highest mean core number (CORE) of the endpoints. It can be observed that MRKC outperforms all the baselines.

edge (u, v) is added, we only need to check the purecore of u and v., so the following edge insertions are faster. The only exception is the AS_1999 network, where the runtime increases constantly. This is because there are a large number of nodes with large purecores, so subsequent checks still take a significant amount of time for this network.

6 CONCLUSIONS

In this paper, we discussed the problem of capturing how the k-core structure of a network changes due to deleted edges or nodes. To address this we proposed a measure called *Core Resilience* of a network (Section 3), which measures how much the ordering of the nodes by core number is affected when there are missing edges and nodes.

Computing the core resilience of a large networks is potentially expensive, and so we proposed two node measures based on network structure. The two measures - *Core Strength* and *Core Influence*, can be used together to tell us if a network is likely to have high core resilience or not. We proposed a method called Maximize Resilience of k-core (MRKC) to add edges to a network without changing the core number of any node, such that the core resilience of the resulting network is improved. We tested our method against baselines on multiple real-world networks, and found that it can improve the core resilience against edge deletion by 19% on average, and against node deletion by 19.7% over the original network.

There are several future research directions that we plan to pursue. We observed that in some networks the $\mathcal{R}_r^{\cdot(p)}$ is non-monotonic with respect to p. Why do some networks have this behavior, and which structural properties of the network can be used to predict this behavior? Another direction is to consider specific attack scenario - if there is an attacker which disables the nodes/edges in a targeted manner, how can we extend our work to such cases?

ACKNOWLEDGMENTS

Laishram and Soundarajan were supported by Army Research Office award W911NF-18-1-0047. Eliassi-Rad was supported by NSF-CNS-1314603 and NSF-IIS-1741197. Ali Pinar was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

REFERENCES

[1] Abhijin Adiga and Anil Kumar S. Vullikanti. 2013. *How Robust Is the Core of a Network?* Springer Berlin Heidelberg, Berlin, Heidelberg, 541–556.

[2] Md Altaf-Ul-Amine, Kensaku Nishikata, Toshihiro Korna, Teppei Miyasato, Yoko Shinbo, Md Arifuzzaman, Chieko Wada, Maki Maeda, Taku Oshima, Hirotada Mori, et al. 2003. Prediction of protein functions based on k-cores of protein-protein interaction networks and amino acid sequences. *Genome Informatics* 14 (2003), 498–499.

[3] J. Ignacio Alvarez-Hamelin, Alain Barrat, and Alessandro Vespignani. 2006. Large scale networks fingerprinting and visualization using the k-core decomposition. In *NIPS*. 41–50.

[4] José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. 2005. K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases. *arXiv preprint cs/0511007* (2005).

[5] J. Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. 2008. K-core decomposition of Internet graphs: hierarchies, self-similarity and measurement biases. *Networks and Heterogeneous Media* 3, 2 (2008), 371–293. http://cnet.fi.uba.ar/ignacio.alvarez-hamelin/pdf/NHM_AH_D_B_V_2008.pdf

[6] V. Batagelj and M. Zaversnik. 2003. *An O(m) Algorithm for Cores Decomposition of Networks*. Technical Report cs/0310049. Arxiv.

[7] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. 2014. Core Decomposition of Uncertain Graphs. In *KDD*. 1316–1325.

[8] Shai Carmi, Shlomo Havlin, Scott Kirkpatrick, Yuval Shavitt, and Eran Shir. 2007. A model of Internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences* 104, 27 (2007), 11150–11154.

[9] James Cheng, Yiping Ke, Shumo Chu, and M. Tamer Ozsu. 2011. Efficient Core Decomposition in Massive Networks. In *ICDE*. 51–62.

[10] Sergey N Dorogovtsev, Alexander V Goltsev, and Jose Ferreira F Mendes. 2006. K-core organization of complex networks. *Physical review letters* 96, 4 (2006), 040601.

[11] P. Erdős and A. Hajnal. 1966. On chromatic number of graphs and set-systems. *Acta Mathematica Hungarica* 17 (1966), 61–99.

[12] Christos Giatsidis, Dimitrios M. Thilikos, and Michalis Vazirgiannis. 2011. Evaluating Cooperation in Communities with the k-Core Structure. In *ASONAM*. 87–93.

[13] Christos Giatsidis, Dimitrios M. Thilikos, and Michalis Vazirgiannis. 2013. D-cores: measuring collaboration of directed graphs based on degeneracy. *Knowl. Inf. Syst.* 35, 2 (2013), 311–343. https://doi.org/10.1007/s10115-012-0539-0

[14] WU Jun, Mauricio Barahona, Tan Yue-Jin, and Deng Hong-Zhong. 2010. Natural connectivity of complex networks. *Chinese physics letters* 27, 7 (2010), 078902.

[15] Wissam Khaouid, Marina Barsky, S. Venkatesh, and Alex Thomo. 2015. K-Core Decomposition of Large Networks on a Single PC. *PVLDB* 9, 1 (2015), 13–23. http://www.vldb.org/pvldb/vol9/p13-khaouid.pdf

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

[17] David. W. Matula. 1968. A min-max theorem for graphs with application to graph coloring. *SIAM Rev.* 10, 4 (1968), 481–482.

[18] Michael P. O'Brien and Blair D. Sullivan. 2014. Locally Estimating Core Numbers. In *ICDM*. 460–469.

[19] Chengbin Peng, Tamara G Kolda, and Ali Pinar. 2014. Accelerating community detection by using k-core subgraphs. *arXiv preprint arXiv:1403.2226* (2014).

[20] Ahmet Erdem Sarıyüce, Buğra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ü. V. Çatalyürek. 2013. Streaming Algorithms for K-core Decomposition. *Proc. VLDB Endow.* 6, 6 (April 2013), 433–444. https://doi.org/10.14778/2536336.2536344

[21] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[22] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. 2016. CoreScope: Graph Mining Using k-Core AnalysisâĂŤPatterns, Anomalies and Algorithms. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on.* IEEE, 469–478.

[23] D. Wen, L. Qin, Y. Zhang, X. Lin, and J. Yu. 2016. I/O efficient Core Graph Decomposition at web scale. In *ICDE*. 133–144.

[24] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2017. Finding Critical Users for Social Network Engagement: The Collapsed k-Core Problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 245–251. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14349