Quantifying Node-Based Core Resilience

Jakir Hossain[1](✉), Sucheta Soundarajan[2], and Ahmet Erdem Sarıyüce[1]

[1] University at Buffalo, Buffalo, NY 14260, USA
{mh267,erdem}@buffalo.edu

[2] Syracuse University, Syracuse, NY 13244, USA
susounda@syr.edu

Abstract. Core decomposition is an efficient building block for various graph analysis tasks such as dense subgraph discovery and identifying influential nodes. One crucial weakness of the core decomposition is its sensitivity to changes in the graph: inserting or removing a few edges can drastically change the core structure of a graph. Hence, it is essential to characterize, quantify, and, if possible, improve the resilience of the core structure of a given graph in global and local levels. Previous works mostly considered the core resilience of the entire graph or important subgraphs in it. In this work, we study node-based core resilience measures upon edge removals and insertions. We first show that a previously proposed measure, Core Strength, does not correctly capture the core resilience of a node upon edge removals. Next, we introduce the concept of dependency graph to capture the impact of neighbor nodes (for edge removal) and probable future neighbor nodes (for edge insertion) on the core number of a given node. Accordingly, we define Removal Strength and Insertion Strength measures to capture the resilience of an individual node upon removing and inserting an edge, respectively. As naive computation of those measures is costly, we provide efficient heuristics built on key observations about the core structure. We consider two key applications, finding critical edges and identifying influential spreaders, to demonstrate the usefulness of our new measures on various real-world networks and against several baselines. We also show that our heuristic algorithms are more efficient than the naive approaches.

1 Introduction

The k-cores are proposed as the seedbeds in which cohesive subsets of nodes can be found [36]. A k-core is defined as the maximal connected subgraph in which every vertex has at least k neighbors in the subgraph. Each node is assigned a core number which denotes the maximum k for which the node is a part of k-core. Thanks to its linear time complexity, k-cores are used as a standard tool in various applications at downstream graph analytics. Examples include the analysis of internet topology [11], predicting protein interactions [2], identifying influential spreaders [20], and community detection [3,6,16,21].

Despite its widespread use, k-cores are known to have a weak resilience against a few changes in the graph [1,22]. Inserting or removing a few edges

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14171, pp. 259–276, 2023.
https://doi.org/10.1007/978-3-031-43418-1_16

can drastically change the core structure of a graph. In applications where noise is common or the studied graph has uncertain parts, core decomposition is not reliable. For example, networks are constructed as a result of indirect measurements in various applications, such as the Internet router/AS level graphs by traceroutes [15], biological networks by experimental correlations [35], and social media based networks by limited samples via the APIs [5]. It is essential to characterize, quantify, and, if possible, improve the resilience of the core structure of a given graph in such applications at global and local levels.

In previous works, the resilience of k-core is studied under node or edge removal to improve users' involvement in social networks [29,42,45], bolster connections to protect a social network from unraveling [9] and determining the edges that should be monitored for attacks on technological networks [22]. Those studies only consider the core structure of the entire graph or a few important subgraphs (e.g., maximum k-cores). **There is no holistic study to quantify the node-based core resilience for any given node in the graph upon edge removals and edge insertions.** Considering the query-driven scenarios in uncertain or noisy networks where the properties of the nodes are important, such as identifying influential nodes in spreading processes [26] or information diffusion [27] and finding critical nodes/edges [31], it is crucial to measure the resilience of core numbers against edge removals as well as insertions.

In this work, we study node-based core resilience measures upon edge removals and insertions. We first demonstrate that a previously proposed node-based measure, Core Strength [22], is inaccurate at capturing the changes in the core number upon edge removals. Next, we propose the concept of dependency graph which captures the impact of neighbor nodes (for removal case) and probable future neighbor nodes (for insertion case) on the core number of a given node. In the dependency graph for removal, one-way dependency relationships between neighboring nodes help to identify the resilience of core numbers. Likewise, in the insertion case, we discover the one-way dependency relationships to quantify the likelihood of a change in the core number. Using the dependency graphs, we define a pair of Removal Strength and Insertion Strength measures for each node. Calculating those node strengths for big graphs in a naive way is computationally intensive. For edge removal, we use the equal edges [46] and k-corona [17] properties to design RSC algorithm. For insertion, we design ISC algorithm based on the number of connections a node has with the same or higher core number. As node-level aspects of a graph are important in many real-world applications, we consider two applications to demonstrate the benefit of our new measures: finding the most critical edges to remove/insert such that the number of nodes that changes their initial core numbers is maximized [14,45,46] and identifying influential spreaders [13,28,40]. For both applications we compare our node-based measures against several state-of-the-art baselines.

Our contributions can be summarized as follows:

- We point out that the Core Strength definition (by [22]) is incorrect and provide counterexamples as well as empirical results to show its unreliability.

- To quantify the node resilience upon edge removal and edge insertion, we use the concept of dependency graphs. Accordingly, we introduce a pair of Removal Strength and Insertion Strength measures.
- We design RSC and ISC algorithms to compute the new node resilience measures for removal and insertion.
- We consider two motivating applications to examine the effectiveness of those metrics: finding critical edges and identifying influential spreaders.
- We evaluate our measures and algorithms on real-world networks. We demonstrate the efficiency and effectiveness of our techniques against several baselines on the two applications mentioned above.

2 Background

In this work, we consider $G = (V, E)$ as an undirected and unweighted graph, where V and E represent the set of nodes and edges in G, respectively. We use $\bar{E}$ to denote the complement of E, i.e., $\bar{E} = \{(u, v)|u \in V, v \in V, (u, v) \notin E\}$. We use $N(u, G)$ to represent the set of neighbors of u in G and $\Gamma(u, G)$ to denote the distance-2 neighbors of u. Let $S \subseteq G$ be a subgraph of G. We use $deg(u, S)$ to denote the degree of u in S. In some cases we consider a directed graph G' in which $deg^{-}(u, G')$ and $deg^{+}(u, G')$ denotes the in-degree and out-degree of u in G', respectively. In our notations, we omit G when it is obvious.

The k-core, denoted by $C_k(G)$, is the maximal connected subgraph $S \subseteq G$ where every vertex has at least k connections in S, i.e., $deg(u, S) \geq k \; \forall u \in G$. The core number of a vertex is the largest k value for which a k-core contains the vertex. Here, $K(u, G)$ denotes the core number of u in G, and $K_{.}(G)$ is the core vector, which is the core numbers of all vertices in G. The maximum k-core(s) of a graph are the (non-empty) k-cores with largest value of k. The k-shell of a graph is the set of nodes with core number k [11] and a subcore is a connected subgraph of nodes with core number k [32]. The k-cores (for all k) are computed by recursively removing vertices with degree less than k and their adjacent edges, while assigning core numbers during the process, which takes $O(|E|)$ time [8].

We define the subset of neighbors of a node u based on the relative core numbers: $\Delta_{<}(u, G)$ denotes the neighbors with smaller core numbers, i.e., $\{v : v \in N(u, G) \wedge K(v, G) < K(u, G)\}$ and $\Delta_{=}(u, G)$ is the neighbors with equal core numbers. Similarly, $\Delta_{>}(u, G)$ and $\Delta_{\geq}(u, G)$ are defined.

3 Related Work

Network resilience is the capability of a network to maintain or restore its function under faults. Characterizing the resilience of a network is important for critical systems such as power grids and transportation systems [24]. Characterization of the resilience is made with respect to various graph characteristics, such as components and paths [34]. One interesting direction in this context is the resilience of the core structure. Core decomposition is one of the most widely used graph algorithms thanks to its linear complexity [36]. However, it is quite sensitive to changes in the graph and there are a few studies to characterize and improve its robustness [1,10,14,22,44]. Here we summarize the literature on core resilience and explore its significance in two motivating applications: finding critical edges and identifying influential spreaders.

Table 1. Comparison of previous works on core resilience and our work.

	[1]	[22]	[10]	[14]	[44]	Our work
Graph structure	Max cores	Entire graph	Entire graph	Entire graph	k-shells	Core number
Edge insertion	Yes	No	No	No	Yes	Yes
Edge removal	Yes	Yes	No	No	Yes	Yes

Core Resilience. Adiga and Vullikanti found that the stability of maximum k-cores under noise and sampling perturbations does not degrade in a monotonic way [1]. Laishram et al. defined the core resilience of a graph as the correlation between the core number rankings of the top r% nodes before and after p% edges or nodes are removed at random [22]. As computing this is costly, they proposed Core Strength and Core Influence measures as proxy to quantify the resilience of a node's core number upon node or edge deletions. Burleson-Lesser et al. modeled network robustness by using the histogram of core numbers [10] and found that ecological and financial networks with U-shaped histograms are resilient to node deletion attacks. Dey et al. defined a graph's stability based on changes in each node's core number upon node removals and studied identifying critical nodes to delete to maximize the number of nodes falling from their initial cores [14]. More recently, Zhou et al. studied attack strategies to change the core numbers of the nodes by rewiring edges [44]. Unlike those studies, we focus on node-based core resilience and consider both removal and insertion. For a given node, we quantify the resilience of its core number upon edge insertion and removals. Table 1 compares our work and previous studies on core resilience.

Finding Critical Edges. A related line of work has proposed problems to minimize and maximize the size of a k-core by inserting/removing nodes/edges. For the removal, the motivation is often to find critical nodes/edges that should be kept in the graph to avoid unraveling in social networks or be watched against targeted attacks in infrastructure networks [29,42,43,46]. In the context of core resilience, such nodes/edges are the weak structures with low resilience against removal and are suitable for targeted attacks. Regarding the insertion, the objective is to find new edges that can increase the user engagement [38,45] or incentivize existing nodes to stay engaged so that other nodes are kept engaged as well [9,23,41]. In the scope of core resilience, such nodes/edges are the critical graph structures that are most vulnerable to increases in core numbers or core sizes. All those works consider a specific k-core and study targeted attacks to change the core structure with a minimal number of edge/node changes. In this work, we focus on the core numbers and use our new node-based core resilience measures to select a limited number of edges so as to maximize the number of nodes whose core numbers change (see Sect. 5.2).

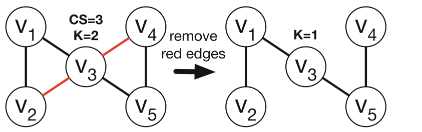
Identifying Influential Spreaders. Another application that core numbers are heavily used is identifying influential spreaders [20]. Influential spreaders are the nodes that determine how information spreads over the network or how a virus is propagated [4,18]. SIR (Susceptible-Infected-Recovered) model is a classical tool to measure the influence of a given set of nodes [37]. In the SIR model, a set of initially infected nodes are chosen which will spread the disease at each time step, t. The fraction of infected nodes, denoted by $S(t)$, is used to measure the spread after t iterations. Kitsak et al. demonstrated that the most efficient spreaders are located in the highest k-shells [20]. Wang et al. discovered that greedily choosing multiple spreaders may result in some of them being too close to each other and hence their influence overlaps [39]. They proposed the IKS algorithm to select nodes from different k-shells based on the highest node information entropy, which outperforms the other centrality or core-number based measures. Considering the success of core-based measures, we use node-based core resilience as a proxy to identify influential spreaders (see Sect. 5.3).

4 Node-Based Core Resilience

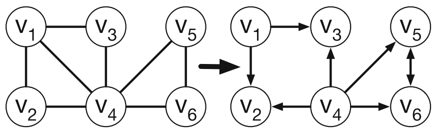
Earlier studies mostly defined core resilience measures for the entire graph. One exception is the core strength (CS) definition by Laishram et al. [22], which aims to measure the resilience of a node's core number upon edge removals and is defined as $CS(u, G) = |\Delta_{\geq}(u, G)| - K(u, G) + 1$. They claim that in order to decrease $K(u, G)$, at least $CS(u, G)$ connections from u must be removed. Here we show that this claim is not true by a simple counterexample and give empirical evidence to show how frequently it fails in practice.

Consider the toy graph in Fig. 1a. $CS(v_3)$ is 3, which means that at least three edges of v_3 should be removed to decrease its core number, according to Laishram et al. [22]. However, if we remove only (v_3, v_2) and (v_3, v_4), $K(v_3)$ decreases to 1. Note that removing two edges does not always decrease $K(v_3)$, e.g.,. deleting (v_3, v_1) and (v_3, v_2) does not affect $K(v_3)$. Depending on the edges being removed, other nodes may have their core numbers changed too, and this cascading effect may result in decreasing the core number of the vertex of interest. Hence, not only the count but also the position of the removed edges matters in quantifying the node-based core resilience.

One question is how likely to see such structures, where removing less than $CS(u)$ edges decreases $K(u)$, in real-world networks. We perform a simple experiment to check this. We consider the nodes in the maximum k-cores with a CS of at least two. For each node, we remove one of its edges and observe its core number changes. We repeat this for each edge of a node. Removing even a single edge is sufficient around 10% of the time to decrease the core number (as shown in [19]). Hence, the CS definition also fails frequently in practice.



(a) A counterexample: removing less than $CS(v_3)$ edges decreases $K(v_3)$.



(b) A toy graph (left) and its removal dependency graph (right)

Fig. 1. Illustrative examples

As the CS definition is inaccurate in capturing the core resilience of a node upon edge removals, we define a new measure, Removal Strength, to compute the likelihood of a node's core number change (Sect. 4.1). Moreover, we propose a new measure, Insertion Strength, to assess the stability of a node's core number after an edge insertion (Sect. 4.2).

4.1 Resilience Against Edge Removal

We capture a node's core resilience against edge removals by analyzing its dependency on neighbor nodes. We focus on single edge removal, with a conjecture that multiple edge removals can be approximated by considering multiple single edge removals.

We define that the node u is dependent on node v, denoted as a relationship $v \rightarrow u$, if $K(u)$ decrements after removing the edge (u, v). For a given graph $G = (V, E)$, we define the *removal dependency graph*, denoted by $G^{rd} = (V, E^{rd})$, as a directed graph such that an edge $(u, v) \in E^{rd}$ if $(u, v) \in E$ and $K(v, G \setminus (u, v)) < K(v, G)$. We give an example in Fig. 1b. For the toy graph on the left, the corresponding removal dependency graph (G^{rd}) is given on the right. In the G^{rd}, v_2 has two in-neighbors (v_1 and v_4) which means it is dependent on v_1 and v_4. For an edge in G, if neither node is dependent on the other, then no edge will appear in the G^{rd}, such as (v_1, v_4). If each node is dependent on another, then there are two edges in both directions, as for (v_5, v_6).

In-degree and out-degree of a node in the removal dependency graph give important insights about its core resilience. A node with a large in-degree is dependent on many of its neighbors, hence removing a nearby edge could reduce its core number, implying a lower core resilience. We define *In-Degree Removal Strength* of a node u, $RS_{ID}(u)$, to quantify the resilience of u to retain its core number upon edge removal(s): $RS_{ID}(u) = \frac{1}{deg^{-}(u, G^{rd})}$. The higher a node's out-degree in the dependency graph, the more strength it has to change the other nodes' core numbers. We define *Out-Degree Removal Strength* of a node u, $RS_{OD}(u)$, to quantify the strength of u to change the core number of other nodes: $RS_{OD}(u) = deg^{+}(u, G^{rd})$.

4.1.1 Removal Strength Computation

A naive way to compute the removal dependency graph is to run incremental core decomposition algorithm for every single edge removal [32,33], which will be costly. Here we propose efficient heuristics by using key observations about the core structure.

We define node $u \in G$ as **vulnerable** if $K(u, G) = |\Delta_{\geq}(u, G)|$. For a vulnerable node u, the set of edges (u, v) where $v \in \Delta_{\geq}(u, G)$ is called the **sensitive edges of** u (also called as equal edges in [46]).

Lemma 1. *If a sensitive edge (u, v) of a vulnerable node u is removed, then $K(u, G)$ will decrease.*

Proof. Proofs of Lemmas 1–7 are available in the extended version [19].

Sensitive edges of a vulnerable node provide a way to group certain edges whose removal yields the same core vector, as first shown in [46].

Lemma 2. *For a vulnerable node u, removing any sensitive edge yields the same core vector, i.e., $K_{\cdot}(G \setminus \{(u, v_1)\}) = K_{\cdot}(G \setminus \{(u, v_2)\})$ where both edges are sensitive.*

According to [17], k**-corona** is a maximal connected subgraph of vulnerable vertices with the same core number, k. Formally, $S \subseteq G$ is a k-corona if $\forall u \in S$, $K(u, G) = k$ and u is a vulnerable vertex. We define k**-corona adjacent edge set**, $KAES(S)$, as the union of the sensitive edges of the vulnerable nodes in a k-corona S, i.e., $\bigcup_{u \in S} \{(u, v) | v \in \Delta_{\geq}(u, G)\}$.

Lemma 3. *When an edge (u, v) is removed from the graph, there will be a change in the core numbers if and only if the removed edge (u, v) is part of a $KAES$.*

Lemma 4. *For a k-corona S, removing any edge in $KAES(S)$ yields the same core vector, i.e., $K_{\cdot}(G \backslash \{(u, v)\}) = K_{\cdot}(G \backslash \{(x, y)\})$ for $(u, v), (x, y) \in KAES(S)$.*

We define the subset of nodes whose core numbers change after removing a single edge as **Core Changed Nodes (*CCN*)**. According to Lemma 4, for a k-corona S, if we choose any edge $(u, v) \in KAES(S)$ to delete, then we will get the same core vector. Hence we denote the set of nodes whose core numbers change after removing any edge in a $KAES(S)$ as $CCN_{KAES(S)}$.

Instead of examining every single edge in a graph, we can utilize $CCN_{KAES(S)}$ to efficiently detect the changes in the core numbers of nodes. Assume w.l.o.g. that $K(u) \leq K(v)$. If u is a vulnerable node, then, by Lemma 1, deleting an edge (u, v) will decrement the core number of u. If u is not a vulnerable node, we need to look at the properties of both u and v. If v is also not a vulnerable node, then the edge $(u, v) \notin KAES$, and there will be no changes in $K(u)$ or $K(v)$ (by Lemma 3). However, if the node v is vulnerable, we need to consider the following two cases to determine the changes in $K(v)$:

Case 1: $K(u) = K(v)$. Here, (u, v) is a sensitive edge, and deleting (u, v) will decrement $K(v)$ (Lemma 1). In this case, $K(u)$ will also decrement if it becomes affected by the changes in $K(v)$. This information is actually captured by the CCN set. If two nodes are in a same CCN, a change in the core number of one node affects the core number of the other node. Hence, if u and v are in the same $CCN_{KAES(S)}$ for any k-corona S, then their core numbers depend on each other.

Case 2: $K(u) < K(v)$. In this case, $K(v)$ will not change, as shown by [32]. Regarding $K(u)$, as u is not a vulnerable node, it has at least $K(u)+1$ neighbors in its k-core. Since v is not in the k-core of u, there will still be at least $K(u)$ neighbors in u's k-core in $G \setminus \{(u,v\}$, and thus $K(u)$ will not change either.

4.1.2 Removal Strength Algorithm

Building on the definitions and observations above, we propose RSC algorithm (Algorithm 1) to compute RS_{ID} and RS_{OD} for each node in a given graph. At the beginning, we find the k-core(s) of a graph using the classical peeling based algorithm proposed by Batagelj et al. [7] (line 4). Then using the BFS traversal, we compute the set of k-coronas ($\mathcal{S}$) in every k-core subgraph (line 6). Since removing an edge $(u,v) \notin KAES$ does not affect the core number of any node (by Lemma 3), we only consider the edges $(u,v) \in KAES$ in each k-core. For each k-corona S, we find the $KAES$ (line 8). Thanks to Lemma 4, we remove only one edge in $KAES(S)$ and compute $CCN_{KAES(S)}$ for a $KAES(S)$ (line 9) by using the incremental core decomposition algorithm from [33]. Next, we

Algorithm 1: RSC: Removal Strength Computation $(G(V,E))$

```
1  Input: $G\ (V,E)$: graph
2  Output: $RS_{ID}$, $RS_{OD}$: in and out-degree removal strength, respectively
3  $G^{rd}\ (V,E^{'}) \leftarrow$ empty graph                    // removal dependency graph
4  Compute all the $k$-cores of $G$, $C_k(G)$, and put in $\mathcal{C}$
5  foreach $k$-core $C_k(G) \in \mathcal{C}$ do
6      Compute all $k$-coronas in $C_k(G)$ and put in $\mathcal{S}$
7      foreach $k$-corona $S \in \mathcal{S}$ do
8          Find $KAES(S)$
9          Delete any single edge $e \in KAES(S)$, compute
           $CCN_{KAES(S)}$                                       // by [33]
10 foreach $u \in V$ do
11     if $K(u) = |\Delta_{\geq}(u,G)|$ then
12         foreach $v \in N_u$ do
13             if $K(u) \leq K(v)$ then
14                 $E^{'}$.push((v, u))        // $K(u)$ will decrement, by Lemma 1
15     else
16         foreach $v$ in $N_u$ do
17             if $K(v) = K(u)$ & $K(v) = |\Delta_{\geq}(v,G)|$ &
18             $u$ and $v$ are in a same $CCN_{KAES(S)}$ then
19                 $E^{'}$.push((v, u))        // $K(u)$ will decrement, by Case 1
20 foreach $u$ in $V$ do
21     $RS_{ID}(u) \leftarrow \frac{1}{deg^{-}(u,G^{rd})}$, $RS_{OD}(u) \leftarrow deg^{+}(u,G^{rd})$
22 Return $RS_{ID}$, $RS_{OD}$
```

use Lemma 1 and the two observations (Case 1 and 2) to quickly determine whether the core numbers will change after each edge removal (lines 10–19). At the end, we calculate and return the in-degree and out-degree removal strengths of all the nodes by using G^{rd} (lines 20–22).

Time and Space Complexity. Line 4, as well as lines 10–19 takes $O(|E|)$ time. In the worst case, lines 5–9 takes $O(|V| \cdot |E|)$ time—if each node is a k-corona, one edge is removed per node, hence $|V|$ edge removals in total (line 9) and each edge removal takes $O(|E|)$ time per [33]. Overall time complexity is $O(|V| \cdot |E|)$, but this is a loose bound as the number of k-coronas is significantly less than $|V|$ in real-world networks (even for large networks we observe the number of k-coronas to be small, requiring us to remove fewer edges, as shown in Table 2 column 4 (% Gain)). In addition to graph $(O(|E|))$, we store $|\Delta_{\geq}(u, G)|$ values $(O(|V|))$, component ids to bookkeep $CCN_{KAES(S)}$ for each node $(O(|V|))$, and RS_{ID}, RS_{OD} values $(O(|V|))$. Overall space complexity is $O(|E|)$.

4.2 Resilience Against Edge Insertion

We now characterize the resilience of a node's core number against edge insertions. We again focus on the impact of a single edge change, consider the changes in a node's core number based on new links it forms with other nodes, and model the resilience accordingly. Regarding the set of edge insertions, it is impractical and unrealistic to think about all possible new links between any pair of unconnected nodes, namely $\bar{E}$. It is impractical because real-world networks are sparse, i.e., $|E| << \binom{|V|}{2}$, which implies $|\bar{E}| >> |E|$. It is unrealistic as it is unexpected that a link will form between two nodes if they have no common neighbors, i.e., if they are not distance-2 neighbors [25,30]. Even the number of non-neighbor node pairs with at least one common neighbor is too large to be considered, reaching up to $100 \cdot |E|$ for some real-world networks. Furthermore, those node pairs are not located homogeneously in the graph; some nodes (mostly low-degree) have very few (or no) distance-2 neighbors, hence it is not clear how to define insertion core resilience for those (see [19] for statistics).

To address these issues, we consider a fixed number (b) of edge insertions for each node and construct the *insertion candidate graph*, G^{ic}, accordingly. Here, we fix $b = 5$ as it is close to the average degrees of the networks used in experiments and no significant advantage is observed for larger b values. For any node $u \in G$, and its distance-2 neighbors $\Gamma(u)$, we consider the below cases to select the edges and add to G^{ic}:

– If $|\Gamma(u)| > b$, choose b random edges (u, v) such that $v \in \Gamma(u)$.
– Else, choose all (u, v) edges such that $v \in \Gamma(u)$ and choose $b - |\Gamma(u)|$ random (u, w) edge(s) such that $w \in V$ (and $w \notin \Gamma(u)$).

Note that b ensures a lower bound on the degree of a node in G^{ic}, there can be nodes with larger degree due to random edges coming from the other nodes.

We define the dependency relationships between nodes by using the insertion candidate graph, G^{ic}. For each edge $(u, v) \in G^{ic}$, we check how the core numbers of u and v change when (u, v) is inserted to G. u is said to be dependent on node v, denoted as a relationship $v \to u$, if $K(u)$ increases after inserting the edge (u, v). For a given graph $G = (V, E)$ and $G^{ic} = (V, E^{ic})$, we define *insertion dependency graph*, $G^{id} = (V, E^{id})$, as a directed graph such that an edge $(u, v) \in E^{id}$ if $(u, v) \in E^{ic}$ and $K(v, G \cup (u, v)) > K(v, G)$. Here, G^{id} is always a subgraph of G^{ic}. We give an example in Fig. 2. For the toy graph on the left, corresponding insertion candidate graph is given in the middle (for $b = 2$). All the nodes except v_2 has at least two distance-2 neighbors. To ensure v_2 has two edges, we randomly select a node, v_5, and put an edge between them. Straight edges in the candidate graph are the edges due to the distance-2 neighborhood (the if condition above) and the dashed edge is the random edge (from the else condition). The corresponding insertion dependency graph is shown on the right. For example, inserting (v_3, v_5) edge would increase $K(v_5)$ and does not impact $K(v_3)$, hence $(v_3 \to v_5)$ is put.

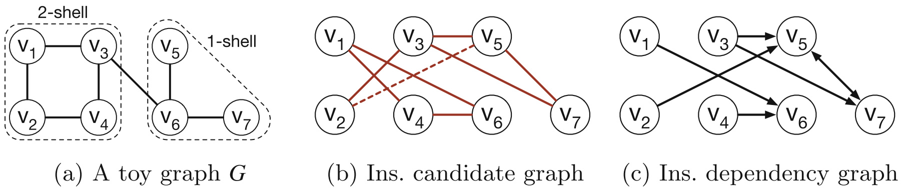


Fig. 2. Examples for insertion candidate ($b = 2$) **and dependency graphs.**

A node with a large in-degree is dependent on many of its distance-2 (or random) neighbors, implying a lower core resilience. We define *In-Degree Insertion Strength* of a node u, IS_{ID}, to measure the node's ability to preserve its core number after edge insertion: $IS_{ID}(u) = \frac{1}{deg^{-}(u, G^{id})}$. A node with a large-out degree implies the ability to increase the core numbers of other nodes. We define *Out-Degree Insertion Strength* of a node u, IS_{OD}, to measure the strength of a node to impact the nodes around it: $IS_{OD}(u) = deg^{+}(u, G^{id})$.

4.2.1 Insertion Strength Computation

A naive computation of the insertion dependency graph is to run incremental core decomposition algorithm for every single edge insertion [32,33], which is costly. Here we consider four lemmas that help to determine the core number changes without running the incremental algorithm.

Lemma 5. *For a node u such that $K(u, G) = |\Delta_{>}(u, G)|$, adding a new edge (u, v) s.t. $K(v, G) > K(u, G)$ will increment $K(u)$, i.e., $K(u, G \cup (u, v)) = K(u, G) + 1$.*

Lemma 6. *For two non-neighbor nodes u and v, $(u, v) \notin E$, such that $K(u, G) = K(v, G) = k$, if $|\Delta_{>}(u, G)| = K(u, G)$ and $|\Delta_{>}(v, G)| = K(v, G)$, then adding a new edge (u, v) will increment $K(u)$ and $K(v)$.*

Lemma 7. *Consider a node $u \in G$ such that $|\Delta_{>}(u, G)| = K(u, G) - 1$. Say u has a neighbor w for which $K(u, G) = K(w, G)$ and $|\Delta_{>}(w, G)| = K(w, G)$. Adding a new edge (u, v) such that $K(v, G) > K(u, G)$ will increment $K(u)$ and $K(w)$.*

4.2.2 Insertion Strength Algorithm

We use the above lemmas (Lemma 5 to Lemma 7) to design ISC (Insertion Strength Computation) algorithm which creates the insertion dependency graph by determining the changes in the core numbers (pseudocode is in [19]). We start by computing the k-cores by [8]. We consider the edges of G^{ic}, which is given, to build the dependency graph of insertion. To construct the G^{id}, we check whether each edge $e \in G^{ic}$ can be handled by the lemmas given in Sect. 4.2.1. If the conditions in any of the lemmas are satisfied, we can readily determine the dependency graph and tell if inserting the new edge (u, v) will change the $K(u)$ and/or $K(v)$. If the edge does not fit to any of the lemmas, we use the incremental core decomposition algorithm [33] to determine the new core numbers. For each of the cases, if there is any increase in $K(u)$ and/or $K(v)$, we update the G^{id} by inserting directed edges based on the core number changes. At the end, we calculate and return the insertion strength measures of each node $u \in G$ by using the G^{id}.

Time and Space Complexity. In the worst case, incremental core decomposition algorithm [33], which takes $O(|E|)$, is run for each edge in G^{ic}. There is $O(b \cdot |V|)$ edges in G^{ic} where b is a constant. In total, the time complexity is $O(|V| \cdot |E|)$. However, this is a loose bound as we show runtime results in Sect. 5. In addition to graph ($O(|E|)$), we store $|\Delta_{>}(u, G)|$ values ($O(|V|)$) and IS_{ID}, IS_{OD} values ($O(|V|)$). Overall space complexity is $O(|E|)$.

Table 2. Statistics for the networks (first two columns) and runtime results for edge removal and edge insertion (in seconds). %Gain denotes the savings how much less edges are processed by our algorithm than the naive approach for edge removal. $\frac{|E^{ic}|}{|E|}$ denotes the ratio of the number of edges in the insertion candidate graph to the actual graph. Sp. is the speedup of our algorithms against the naive approaches.

			Removal				Insertion			
Graph	$\|V\|$	$\|E\|$	% Gain	Naive (s)	RSC (s)	Sp.	$\frac{\|E^{ic}\|}{\|E\|}$	Naive (s)	ISC (s)	Sp.
as19971108	3015	5156	50.2 %	4.88	2.93	1.67×	2.79	1.10	0.75	1.46×
as19990309	4759	8896	54.4 %	12.38	6.35	1.95×	2.58	1.70	1.31	1.30×
bio-dmela	7393	25569	79.4 %	56.96	12.85	4.43×	1.40	38.26	26.60	1.44×
ca-CondMat	21363	91286	89.0 %	575.82	67.19	8.57×	1.11	475.15	377.32	1.26×
ca-Erdos992	5094	7515	39.1 %	10.93	7.47	1.46×	3.11	7.48	5.23	1.43×
ca-GrQc	4158	13422	84.4 %	17.82	3.47	5.14×	1.39	32.41	26.66	1.22×
inf-openflights	2939	15677	86.8 %	15.67	2.49	6.29×	0.90	2.64	2.28	1.16×
inf-power	4941	6594	63.8 %	9.36	3.96	2.36×	2.76	504.87	486.43	1.04×
jazz	198	2742	97.8 %	0.43	0.06	7.17×	0.35	1.00	0.94	1.06×
p2p-Gnutella08	6301	20777	80.3 %	40.83	8.99	4.54×	1.45	951.50	918.05	1.04×
p2p-Gnutella09	8114	26013	78.8 %	63.81	14.51	4.40×	1.49	769.12	713.63	1.08×
soc-hamsterster	2426	16630	93.6 %	13.03	1.27	10.26×	0.72	4.82	4.22	1.14×
soc-wiki-Vote	889	2914	81.2 %	0.88	0.29	3.03×	1.46	1.45	1.09	1.33×
tech-routers-rf	2113	6632	77.8 %	4.26	1.30	3.28×	1.50	2.46	1.98	1.25×
tech-WHOIS	7476	56943	89.9 %	128.82	14.51	8.88×	0.65	6.59	5.60	1.18×
USAir97	332	2461	91.2 %	0.28	0.08	3.50×	0.65	0.12	0.09	1.46×
web-spam	4767	37375	91.5 %	52.48	5.28	9.94×	0.63	7.05	5.17	1.36×

5 Experimental Evaluation

We conduct experiments on real-world networks of various types and sizes to evaluate the efficiency and effectiveness of our node-strength measures. Table 2 (first three columns) shows the statistics of the networks, obtained from SNAP[1] and Network Repository[2]. All experiments are performed on a Linux operating system (v. 3.10.0-1127) running on a machine with Intel(R) Xeon(R) Gold 6130 CPU processor at 2.10 GHz with 192 GB memory. We implemented our algorithms in Python 3.6.8. **Our implementation is publicly available**[3].

Since we consider random edge selections to construct G^{ic} and calculate IS_{ID} and IS_{OD}, we repeat insertion experiments 10 times to account for randomness and report the average strength measure for each node. Note that the standard deviation in those computations is quite low, e.g., in `inf-openflights` graph, the standard deviation is less than .18 for most nodes where more than half of the nodes have zero (or close to zero) standard deviation (details are in [19]).

[1] http://snap.stanford.edu/.
[2] http://networkrepository.com/.
[3] https://github.com/erdemUB/ECMLPKDD23.

5.1 Runtime Results

We first compare the runtime performances of our RSC and ISC algorithms against the naive strategy which simply runs incremental core decomposition algorithms for each edge removal and edge insertion. One important note is that the three approaches (Subcore, Purecore, and Traversal) proposed in [32] give different behaviors in our removal and insertion experiments. Although the Traversal algorithm is shown to be the best in [32] for both single edge removal and insertion, we observe that the Subcore algorithm can be made faster for edge insertion in our experiments. The key is to precompute all the subcores in each k-core and reuse when handling edge insertions. We use this pre-calculation technique and Subcore algorithm in our edge insertion experiments, whereas the Traversal algorithm is used in our edge removal experiments.

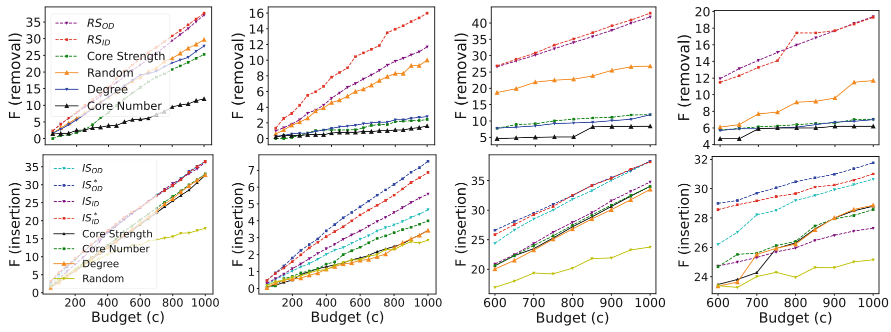
Table 2 gives the results. For the edge removal, we are able to remove 78.2% less edges, on average, when compared to the naive approach (fourth column in Table 2). This translates to 5.11× faster runtime on average. For edge insertion, our algorithm well utilizes the lemmas in Sect. 4.2.1 and gives 1.25× faster computation when compared to the naive approach.

5.2 Finding Critical Edges

Here, we compare our node resilience measures to several baselines for finding critical edges in edge removal and insertion scenarios. We use four baselines: Random, Core Number, Degree, Core Strength. Each method identifies a limited number (c) of critical edges to maximize the impact on the core numbers of affected nodes. For Random, we repeat experiments 50 times and take the average. For Core Number, Degree, and Core Strength; the score of each edge is determined by the sum of its end points' values and c edges with the highest score are considered. We assess each method by the percentage of nodes affected, F, (decreased or increased from the initial core number) by the removal or insertion of the budget number of edges. For all experiments, we vary the budget (c) from 50 to 1000 and evaluate our results. For better visualization, we show the results from budget 600 to 1000 in Fig. 3(c) and Fig. 3(d).

5.2.1 Edge Removal Experiments.

We use RS_{ID} and RS_{OD} to select c critical edges to remove from the graph. For our measures, the score of each edge is set as the sum of its endpoints' RS_{ID} or RS_{OD} values. For RS_{ID}, we choose c edges with the lowest scores as a node with lower RS_{ID} is more likely to change its core number on edge removal, whereas, for RS_{OD}, we select c edges with the highest score as a node with larger RS_{OD} affect other nodes' core numbers more. We also pay attention to not selecting no more than one edge from any $KAES(S)$, as removing any edges in $KAES(S)$ produces the same core vector for a k-corona S by Lemma 4. For Random, we choose c random edges from the graph. Figure 3 (top row) shows the results for four graphs (results for other graphs are in [19]). Both RS_{ID} and RS_{OD} outperform the baselines. RS_{ID} is slightly better than RS_{OD} in some graphs and significantly better in a few.



(a) as19971108 (b) ca-CondMat (c) inf-openflights (d) p2p-Gnutella08

Fig. 3. Finding critical edges by our methods and baselines for edge removal (top row) and edge insertion (bottom row).

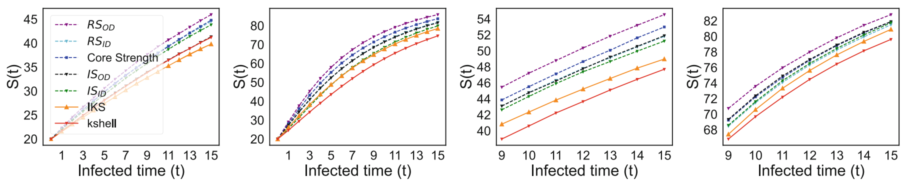
5.2.2 Edge Insertion Experiments

We use IS_{OD} and IS_{ID} to select c critical edges to insert to the graph. We first consider all the non-neighbor node pairs who share at least two common neighbors, called as candidate set, and then select a subset of size c edges of the candidate set by using the baselines or our methods. When inserting an edge (u, v), if core number of both u and v increased by a previous insertion, we skip this edge. For our measures, we define the score of each candidate edge (u, v) as $\max(IS_{ID}(u), IS_{ID}(v))$ (likewise for IS_{OD}), then select the c edges with lowest scores to insert. Here, we consider maximum endpoint strength as the edge score, unlike the edge removal case where we considered sum, to keep the scores more regularized because the space of edge insertions is larger and can yield very large edge scores if the sum is applied. We choose the edges with the lowest score as they are the least resilient for incrementing core numbers. For Random, we choose c random edges from the candidate set. Figure 3 (bottom row) gives the results for four networks (rest are in [19]). Overall, IS_{ID} and IS_{OD} consistently outperform the baselines.

We also define a simple variant of our measures to handle the clique-like structures in which core numbers are difficult to increase. We consider the propagation effect of neighbor nodes by summing up the strength of a node with its neighbors' strengths. We define Neighbor Sum variants as $IS^*_{ID}(u) = IS_{ID}(u) + \sum_{v \in N(u)} IS_{ID}(v)$ (likewise for IS^*_{OD}). As above we define the score of each candidate edge the maximum strength of its endpoints and then select the c edges with the lowest scores to insert. As shown in Fig. 3 (bottom row), IS^*_{ID} and IS^*_{OD} significantly outperform all the other methods in `ca-CondMat`, which is a co-authorship network formed by cliques of authors on a paper.

5.3 Identifying Influential Spreaders

In this section, we consider the problem of identifying influential spreaders in the SIR model. We use our node resilience measures as well as three baselines to choose 20% nodes in a given graph as the initially infected node set. For our measures ($RS_{ID}, RS_{OD}, IS_{ID}, IS_{OD}$), we choose the node with largest strength from the highest k-shell, then do the same for the next highest shell (($k-1$)-shell), and so on until the 1-shell. Then we repeat this process until 20% of the nodes are chosen. Ties are broken randomly. As the highest strength nodes are more resilient upon graph changes, they are more important for influence maximization than others. Regarding the baselines, we choose the methods that rely on core numbers—the k-shell strategy [20], the IKS method [39], and the Core Strength measure (the nodes with the largest values)—to select the 20% initially infected node set. To ensure a smooth transmission in the SIR model, we fix S→I probability $\mu = 0.01$ and set the value of I→R probability β to be a little bit bigger than $\beta_{min} = \langle k \rangle / \langle k^2 \rangle$ [12], where $\langle k \rangle$ and $\langle k^2 \rangle$ are the first and the second moment of the degree distribution, as done in [39] (exact β values are in [19]). For each method, we run the model 50 times and take the average. We consider the percentage of affected nodes at time t, denoted as $S(t)$, to evaluate the spreading effect of the initially infected node set.



(a) as19971108 (b) ca-CondMat (c) inf-openflights (d) p2p-Gnutella08

Fig. 4. Identifying influential spreaders by our measures and baselines.

Figure 4 shows $S(t)$ as a function of $t \in [0, 15]$ for four networks (results for other graphs are in [19]). As t increases, $S(t)$ rises and eventually reaches a steady value. Overall, our node strength measures outperform the k-shell and IKS strategies. Core Strength measure shows superior performance than some of our measures but RS_{OD} consistently outperforms all the methods. The reason for this behavior is that the nodes with large RS_{ID} do not always have large core numbers whereas the nodes with large RS_{OD} are consistently in highest k-cores.

6 Conclusions and Future Work

In this paper, we studied the problem of node-based core resilience upon edge removals and edge insertions. We first showed that the Core Strength [22] does not correctly capture the core resilience of a node upon edge removals. Then we introduced the concept of dependency graph to capture the impact of neighbor nodes (for removal) and probable future neighbor nodes (for insertion) on the core number of a given node. We defined node strengths in dependency graphs based on in- and out-degrees and introduced efficient heuristics to compute those. Experiments show that our heuristics are faster than the naive approaches and our strength measures outperform the existing baselines on two key applications, finding critical edges and identifying influential spreaders. For future work, we plan to speed up the computation of insertion strength measures and also consider more realistic scenarios to construct the G^{ic}[4].

Acknowledgments. Hossain and Sarıyüce were supported by NSF Award #1910063 and used resources of the Center for Computational Research at the University at Buffalo. Soundarajan was supported by NSF Award #1908048.

Ethical Statement. Our contribution is algorithmic in nature, building on previously proposed concepts. We work on public datasets. We do not foresee any ethical implications of our work.

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 Inf. **14**, 498–499 (2003)
3. Andersen, R., Chellapilla, K.: Finding dense subgraphs with size bounds. In: Avrachenkov, K., Donato, D., Litvak, N. (eds.) WAW 2009. LNCS, vol. 5427, pp. 25–37. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-95995-3_3
4. Anderson, R.M., May, R.M.: Infectious Diseases of Humans: Dynamics and Control. Oxford University Press, Oxford (1992)
5. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on twitter. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 65–74 (2011)
6. Balasundaram, B., Butenko, S., Hicks, I.V.: Clique relaxations in social network analysis: the maximum k-plex problem. Oper. Res. **59**(1), 133–142 (2011)
7. Batagelj, V., Zaversnik, M.: An O(m) algorithm for cores decomposition of networks. corr. arXiv preprint cs.DS/0310049 37 (2003)
8. Batagelj, V., Zaversnik, M.: Fast algorithms for determining (generalized) core groups in social networks. Adv. Data Anal. Classif. **5**(2), 129–145 (2011)
9. Bhawalkar, K., Kleinberg, J., Lewi, K., Roughgarden, T., Sharma, A.: Preventing unraveling in social networks: the anchored k-core problem. SIAM J. Disc. Math. **29**(3), 1452–1475 (2015)
10. Burleson-Lesser, K., Morone, F., Tomassone, M.S., Makse, H.A.: K-core robustness in ecological and financial networks. Sci. Rep. **10**(1), 1–14 (2020)
11. Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., Shir, E.: A model of internet topology using k-shell decomposition. Proc. Natl. Acad. Sci. **104**(27), 11150–11154 (2007)

[4] https://ubir.buffalo.edu/xmlui/handle/10477/79221.

12. Castellano, C., Pastor-Satorras, R.: Thresholds for epidemic spreading in networks. Phys. Rev. Lett. **105**(21), 218701 (2010)
13. Chen, D., Lü, L., Shang, M.S., Zhang, Y.C., Zhou, T.: Identifying influential nodes in complex networks. Physica A: Stat. Mech. Appl. **391**(4), 1777–1787 (2012)
14. Dey, P., Maity, S.K., Medya, S., Silva, A.: Network robustness via global k-cores. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pp. 438–446 (2021)
15. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. ACM SIGCOMM Comput. Commun. Rev. **29**(4), 251–262 (1999)
16. Giatsidis, C., Thilikos, D.M., Vazirgiannis, M.: D-cores: measuring collaboration of directed graphs based on degeneracy. Knowl. Inf. Syst. **35**(2), 311–343 (2013)
17. Goltsev, A.V., Dorogovtsev, S.N., Mendes, J.F.F.: k-core (bootstrap) percolation on complex networks: critical phenomena and nonlocal effects. Phys. Rev. E **73**(5), 056101 (2006)
18. Guille, A., Hacid, H., Favre, C., Zighed, D.A.: Information diffusion in online social networks: a survey. ACM Sigmod Rec. **42**(2), 17–28 (2013)
19. Hossain, J., Soundarajan, S., Sarıyüce, A.E.: Quantifying node-based core resilience. arXiv preprint arXiv:2306.12038 (2023)
20. Kitsak, M., et al.: Identification of influential spreaders in complex networks. Nat. Phys. **6**(11), 888–893 (2010)
21. Kortsarz, G., Peleg, D.: Generating sparse 2-spanners. J. Algor. **17**(2), 222–236 (1994)
22. Laishram, R., Sariyüce, A.E., Eliassi-Rad, T., Pinar, A., Soundarajan, S.: Measuring and improving the core resilience of networks. In: Proceedings of the 2018 World Wide Web Conference, pp. 609–618 (2018)
23. Laishram, R., Sariyuce, A.E., Eliassi-Rad, T., Pinar, A., Soundarajan, S.: Residual core maximization: an efficient algorithm for maximizing the size of the k-core. In: Proceedings of the 2020 SIAM International Conference on Data Mining, pp. 325–333. SIAM (2020)
24. Lewis, T.G.: The many faces of resilience. Commun. ACM **66**(1), 56–61 (2022)
25. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: Proceedings of the Twelfth International Conference on Information and Knowledge Management, pp. 556–559 (2003)
26. Lin, J.H., Guo, Q., Dong, W.Z., Tang, L.Y., Liu, J.G.: Identifying the node spreading influence with largest k-core values. Phys. Lett. A **378**(45), 3279–3284 (2014)
27. Liu, C., Zhang, Z.K.: Information spreading on dynamic social networks. Commun. Nonlinear Sci. Numer. Simul. **19**(4), 896–904 (2014)
28. Medo, M., Zhang, Y.C., Zhou, T.: Adaptive model for recommendation of news. EPL (Europhys. Lett.) **88**(3), 38005 (2009)
29. Medya, S., Ma, T., Silva, A., Singh, A.: A game theoretic approach for core resilience. In: International Joint Conferences on Artificial Intelligence Organization (2020)
30. Newman, M.E.: Clustering and preferential attachment in growing networks. Phys. Rev. E **64**(2), 025102 (2001)
31. Purevsuren, D., Cui, G.: Efficient heuristic algorithm for identifying critical nodes in planar networks. Comput. Oper. Res. **106**, 143–153 (2019)
32. Sariyüce, A.E., Gedik, B., Jacques-Silva, G., Wu, K.L., Çatalyürek, Ü.V.: Streaming algorithms for k-core decomposition. Proc. VLDB Endow. **6**(6), 433–444 (2013)
33. Sarıyüce, A.E., Gedik, B., Jacques-Silva, G., Wu, K.L., Çatalyürek, Ü.V.: Incremental k-core decomposition: algorithms and evaluation. VLDB J. **25**(3), 425–447 (2016)

34. Schaeffer, S.E., Valdés, V., Figols, J., Bachmann, I., Morales, F., Bustos-Jiménez, J.: Characterization of robustness and resilience in graphs: a mini-review. J. Complex Netw. **9**(2), cnab018 (2021)
35. Schwab, D.J., Bruinsma, R.F., Feldman, J.L., Levine, A.J.: Rhythmogenic neuronal networks, emergent leaders, and k-cores. Phys. Rev. E **82**(5), 051911 (2010)
36. Seidman, S.B.: Network structure and minimum degree. Social Netw. **5**(3), 269–287 (1983)
37. Sharkey, K.J.: Deterministic epidemic models on contact networks: correlations and unbiological terms. Theor. Popul. Biol. **79**(4), 115–129 (2011)
38. Sun, X., Huang, X., Jin, D.: Fast algorithms for core maximization on large graphs. Proc. VLDB Endow. **15**(7), 1350–1362 (2022)
39. Wang, M., Li, W., Guo, Y., Peng, X., Li, Y.: Identifying influential spreaders in complex networks based on improved k-shell method. Physica A: Stat. Mech. Appl. **554**, 124229 (2020)
40. Zareie, A., Sheikhahmadi, A.: A hierarchical approach for influential node ranking in complex social networks. Expert Syst. Appl. **93**, 200–211 (2018)
41. Zhang, F., Zhang, W., Zhang, Y., Qin, L., Lin, X.: OLAK: an efficient algorithm to prevent unraveling in social networks. Proc. VLDB Endow. **10**(6), 649–660 (2017)
42. Zhang, F., Zhang, Y., Qin, L., Zhang, W., Lin, X.: Finding critical users for social network engagement: the collapsed k-core problem. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
43. Zhao, K., Zhang, Z., Rong, Y., Yu, J.X., Huang, J.: Finding critical users in social communities via graph convolutions. IEEE Trans. Knowl. Data Eng. **35**(1), 456–468 (2023)
44. Zhou, B., Lv, Y., Mao, Y., Wang, J., Yu, S., Xuan, Q.: The robustness of graph k-shell structure under adversarial attacks. IEEE Trans. Circ. Syst. II: Express Briefs **69**(3), 1797–1801 (2021)
45. Zhou, Z., Zhang, F., Lin, X., Zhang, W., Chen, C.: K-core maximization: an edge addition approach. In: IJCAI, pp. 4867–4873 (2019)
46. Zhu, W., Chen, C., Wang, X., Lin, X.: K-core minimization: an edge manipulation approach. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1667–1670 (2018)