Estimating Shell-Index in a Graph with Local Information

Akrati Saxena*
akrati.saxena@iitrpr.ac.in

S. R. S. Iyengar*
sudarshan@iitrpr.ac.in

*Department of Computer Science and Engineering,
Indian Institute of Technology Ropar, India

Abstract

For network scientists, it has always been an interesting problem to identify the influential nodes in the given network. K-shell decomposition method is a widely used method which assigns a shell-index value to each node based on its influential power. The k-shell method requires the global information of the network to compute the shell-index of a node that is not feasible for large-scale real-world dynamic networks. In the present work, we propose a method to estimate the shell-index of a node using local information. We further use hill-climbing based approach to hit the top-ranked nodes in a small number of steps.

1 Introduction

Hierarchical organization of networks gives birth to the core-periphery structure. The concept of the core-periphery structure was first proposed by Borgatti and Everett [1] in 2000. The core is a densely connected nucleus of the network that is surrounded by sparsely connected periphery nodes. Core nodes are highly connected with each other and also highly connected with periphery nodes. Authors proposed an algorithm to detect core-periphery structure based on the adjacency matrix division, such that the number of edges is maximum among core nodes that is preceded by the number of edges between core and periphery, and periphery nodes have very less number of

edges among themselves. The complexity of the proposed matrix division method is very high, and so, it is not feasible for large-size real-world networks. Carmi et al. used k-shell decomposition algorithm to identify core-periphery structure in Internet network and classify the network into three categories [2].

The K-shell decomposition method was proposed by Seidman and is widely used to identify core nodes in unweighted networks [3]. This is a well-known method in social network analysis to identify the tightly knit group of influential core nodes. This algorithm works by recursively pruning the nodes from lower degree to higher degree. First, we recursively remove all nodes of degree 1, until there is no node of degree 1. All these nodes are assigned shell-index $k_s = 1$. In the next step, we remove all the nodes of degree 2 if any. While pruning the degree 2 nodes, new nodes having degree 2 or less can be generated and will also be removed in the same iteration. All these nodes will be assigned shell-index $k_s = 2$. Similarly, nodes of degree 3, 4, 5,... are pruned step by step. When we remove nodes of degree k, if there appears any node of degree less than k, it will also be removed in the same step. All these nodes are assigned shell-index k. This method thus divides the entire network into shells and assigns a shell-index to each node. The shell-index increases as we move from the periphery to core and the higher shell-index represents the higher coreness. The innermost shell has the highest shell-index k_{max} and is called nucleus of the network. Vladimir Batagelj et al. proposed an order $O(m)$ algorithm to compute the coreness of all nodes, where m is total number of edges in the network [4].

Many studies have shown that core nodes are highly influential nodes in a network. In the entire paper, by influential node we mean the node having the highest spreading power. In this work, the spreading/influential power of the nodes is computed using SIR spreading model; please refer section 3.2 for more information. Kitsak et al. [5] showed that if the infection is started from a core node, it will spread faster than if it is started from any periphery node. Saxena et al. showed the importance of core nodes in information diffusion on the networks having meso-scale structures [6]. The results show that the information becomes viral once it hits the core nodes and it is spread into multiple communities through the core. Pei and Maske studied the correlation of various centrality measures with the influential power of the nodes on an online social community called LiveJournal and observed that shell-index is an effective centrality measure to identify the most influential nodes [7]. They further extend their study for different types of networks

like Twitter, Facebook, and scientific publishing in the American Physical Society, and observe the similar results [8].

Shell-index is a good centrality measure to denote the influential power of the node, but it has its disadvantages. Firstly, to compute the shell-index of a node using k-shell decomposition method, we need the entire network as it is a global centrality measure. With time the size of real-world networks is growing very fast. It is not feasible to collect the entire network to identify the influential power of a node. So, we need faster methods that can estimate the influential power of a node using local neighborhood information. Secondly, shell-index assigns the same index values to many nodes which actually might have different influential power [9–11].

Lu et al. showed the relationship between degree, h-index [18], and coreness of a node [19]. They show that the h-index family of a node converges to the coreness of the node. In this work, we show that shell-index value of a node can be estimated using its $h^2 - index$ that can be computed using local neighborhood information of the node. We study the correlation of shell-index and $h^2 - index$ with the spreading power of a node and observe that $h^2 - index$ have good correlation with the spreading power and can be used in real-world networks to identify the influential nodes. We further show that $h^2 - index$ has better monotonicity than the shell-index.

In most of the real world applications, we need to identify highly influential nodes to spread the information without having the global information of the network. Gupta et al. [20] proposed Hill-Climbing approach based on the shell-index of the nodes to hit the code nodes. The proposed algorithms cannot be applied in real-world applications as the shell-index is a global parameter and not available locally. We modify the proposed algorithms for $h^2 - index$ and the results show that the highest $h^2 - index$ nodes can be identified in very less number of steps. We further observe that the subgraph having all the highest $h^2 - index$ nodes is connected and we can identify all top-ranked nodes once a node having the highest $h^2 - index$ is found. The detailed algorithm and results are discussed in Section 5.

The rest of this paper is organized as follows. In Section 2 we discuss the related literature. Section 3 covers the preliminary definitions. In Section 4, we discuss the estimation of shell-index using $h^2 - index$ and experimental results. In Section 5, we discuss hill-climbing based approachs to identify the top-ranked nodes using local information and their simulation on real-world networks. Section 7 concludes the paper followed by the future directions.

2 Related Work

In recent years, the problem of identifying influential nodes has attracted researchers from different areas like computer science, epidemiology, biology, statistics, etc. Researchers have defined various centrality measures to compute the importance of a node based on its characteristics and the given application context. These centrality measures can be categorized as local centrality measures and global centrality measures. Centrality measures that can be computed using local neighborhood information of the node are called local centrality measures like degree centrality [21], h-index [18], semi-local centrality [22] etc. The centrality measures that require the entire network for their computation are called global centrality measures like closeness [23], betweenness [24], eigenvector [25], shell-index [3], PageRank [26] etc. The computational complexity of global centrality measures is very high, and it depends on the network size.

The use of a specific centrality measure to identify important nodes is highly application dependent. Recently Kitsak et al. showed that shell-index is highly correlated with the spreading power of a node [5]. These results have been supported by many other studies that show that the core nodes are highly influential and the information flows into multiple communities through the core [6, 27].

K-shell method assigns the same index values to many nodes which actually might have different influential power. Zeng et al. modified k-shell decomposition method and proposed a mixed degree decomposition (MDD) method which considers both the residual degree and the exhausted degree of the nodes while assigning them index values [11]. Liu et al. proposed an improved ranking method that considers both the k-shell value of the node and its distance with the highest k-shell value nodes [12]. The proposed method computes the shortest distance of all nodes with the highest k-shell nodes, so, it has high computational complexity. Liu et al. showed that in real-world networks some core-like groups are formed which are not true-core [13]. The nodes in these groups are tightly connected with each other but have very few links outside. Based on this observation, authors filtered out redundant links which have low diffusion power but supports non-pure core groups to be formed, and then apply k-shell decomposition methods. Authors show that the coreness computed on this new graph is a better metric of influential power and it is highly correlated with spreading power computed using SIR model in the original graph.

Researchers also have proposed hybrid centrality measures by combining the k-shell with other existing centrality measures. Hou et al. introduced the concept of all-around score to find influential nodes [14]. All around score of a node can be defined as, $Score = \sqrt{\|d\|^2 + \|C_B\|^2 + \|k_s\|^2}$, where d is the degree, C_B is the betweenness centrality, and k_s is the shell-index of the node. The degree takes care of local connectivity of the node, betweenness takes care of shortest paths that represent global information, and k-shell represents the position of the node with respect to the center. The total time complexity of the complete process is $O(nm)$, as it depends on the complexity of betweenness centrality that has the highest complexity. Basaras et al. proposed a hybrid centrality measure based on degree and shell-index and showed that it works better than the traditional shell-index [15]. Bae and Kim proposed a method where the centrality value of a node is computed based on the shell-index value of its neighbors; it thus considers both degree and shell-index value of the nodes [16]. The results show that the proposed method outperforms other methods in the scale-free networks with community structure. Tatti and Gionis proposed a graph decomposition method that considers both the connectivity as well as the local density while the k-shell decomposition method only considers the connectivity of the nodes [17]. The running time of the proposed algorithm is $O(|V|^2|E|)$. They further proposed a linear-time algorithm that computes a factor-2 approximation of the optimal decomposition value. All the proposed centrality measures have better monotonicity but all these measures require global information of the network to be computed, and so, they are not favorable in large-scale networks.

Initially, k-shell decomposition method was defined for unweighted undirected networks, but recently it has been extended to different types of networks. Garas et al. extended k-shell method to identify core-periphery structure in weighted networks [28]. They define the weighted degree that considers both the degree as well as the weights of the connected edges. Then the weighted degree is used while applying the k-shell decomposition method. Eidsaa et al. also proposed a method to identify core-periphery structure in weighted networks where they only consider the strength of the nodes while pruning them in each iteration [29]. The strength of a node is defined as, $s_i = \sum_{j \epsilon \Gamma(i)} W_{ij}$, where W_{ij} denotes the weight of an edge connecting nodes i and j. Wei et al. proposed an edge-weighting k-shell method where they consider both the degree as well as the edge-weights and the edge weight is

computed by adding the degree of its two end points [30]. The importance of both of these parameters can be set using a tuning parameter which varies from 0 to 1. If it is set to 0, then the complete importance is given to edge-weights, and if it is set to 1, then the complete importance is given to the degree of the node.

Real-world networks are highly dynamic, and it will not be feasible to recompute the shell-index of each node for every single change in the network. Li et al. proposed a method to update the shell-index value of the affected nodes whenever a new edge is added or deleted from the network [31]. Jakma et al. proposed first continuous, distributed method to compute shell-index in dynamic graphs [32]. Pechlivanidou et al. proposed a distributed algorithm based on MapReduce to compute the k-shell of the graph [33]. Montresor et al. proposed an algorithm to compute k-shell in live distributed systems [34]. They further show that the execution time of the algorithm is bounded by $1 + \sum_{u \in V}[d(u) - k_s(u)]$ and it gives 80 percent reduction in execution time on the considered real-world networks. Dasari et al. proposed a k-shell decomposition algorithm called ParK that reduces the number of random access calls and the size of working set while computing the shell-index in larger graphs [35]. They further proposed a faster algorithm which involves parallel execution to compute the k-shell in larger graphs. Sariyuce et al. proposed the first incremental k-core decomposition algorithms for streaming networks [36]. They show that the proposed method has million times speed-up than the original k-shell method on a network having 16 million nodes.

K-shell method has been widely used in literature to study different networks. Catini et al. used shell-index to identify clusters in PubMed scientific publications [37]. To identify the clusters, a graph is created where the nodes are the publications, and there is an edge between two nodes if the distance between the locations of the corresponding researchers is less than the threshold. In the experiments, authors have taken the threshold 1 km. Based on the k-shell decomposition authors categorize the cities into mono-core and multicore. Later on, the journal impact factors are used to quantify the quality of research of each core. Results show that k-shell decomposition method can be used to identify the research hub clusters.

Core-periphery structure has been studied in the wide variety of networks like financial network [38, 39], human-brain network [40, 41], nervous system of C. elegans worm [42], Blog-networks [43], collaboration network [44, 45], protein interaction networks [46], Communication network of software development team [47–50], Hollywood collaboration network [51], language net-

work [52], youtube social interaction network [53], metabolic networks [54] etc. Researchers have proposed evolving models to generate synthetic networks having core-periphery structure [55, 56]. Karwa et al. proposed a method to generate all graphs for a given shell-index sequence [57].

With time the size of real-world networks is increasing exponentially, so, it is not feasible to collect the entire network to study its global properties. Researchers have focussed to propose fast and efficient methods to identify the influential nodes and their ranking in the given network [58]. Gupta et al. proposed local information based methods to hit the core nodes quickly [20]. Saxena et al. have proposed fast methods to estimate the degree rank of a node using power-law degree distribution [59, 60] and sampling techniques [61, 62]. In [63], authors proposed heuristic methods to fast estimate the closeness rank of a node. Once the shell-index value is estimated using local information, we aim to propose methods to directly estimate the influential rank of a node without computing the influential value of all nodes. The proposed idea is discussed in detail under Future Directions.

3 Preliminaries

3.1 H-Index

The h-index of a node u ($h - index(u)$) is h if h of its neighbors have degree at least h and there is no subset of $h+1$ neighbors where each node belonging to that subset have the degree at least $h + 1$.

The $h^2 - index$ of a node u ($h^2 - index(u)$) is computed by taking its $h - index$ based on the $h - index$ of its neighbors and not their degrees.

Note: The h-index of a list l ($h - index(l)$) is h if h is the highest number such that h entries of the list are equal to or greater than h.

3.2 SIR Model

We use the Susceptible-Infected-Recovered (SIR) spreading model to simulate the spreading process on real-world networks and to compute the spreading power of each node. In SIR model a node can be in three possible states: 1. S (susceptible), 2. I (infected), and 3. R (recovered). Initially all nodes in the susceptible mode except a node which is infected and spread the infection in the network. An infected node will infect each of its susceptible neighbor

with probability λ. If the neighbor gets infected, its status is changed to Infected. Once an infected node contacts all of its neighbors to infect them, its status is changed to recovered state with probability μ. For generality we set $\mu = 1$. Recovered nodes will neither be infected anymore nor infect others, and they remain in the R state until the spreading stops.

Initially, we infect a single node, and all other nodes are susceptible. Then the infection starts spreading from the seed node to the others through links. The spreading process stops when there is no infected node in the network. The number of recovered nodes is considered the spreading power or spreading capability of the original node.

We execute the SIR model 100 times from each node and take the average of the spreading power to compute the final spreading power of the node. The infection probability is taken as $\lambda > \lambda_c$, where $\lambda_c = \langle d \rangle / (\langle d^2 \rangle - \langle d \rangle)$ is the epidemic threshold computed using mean-field theory, where d represents degree of the node [64].

4 Section 1: Shell-Index Estimation

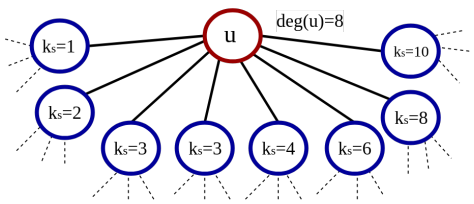
The shell-index of a node is the global centrality measure and require entire network for its computation that is not feasible for large-scale dynamic networks. In this section, we discuss the estimation of shell-index using local neighborhood information.

Theorem 1. *The shell-index of a node u can be computed as $k_s(u) = h - index(k_s(v)|\forall v \in ngh(u))$, where $ngh(u)$ is the set of the neighbors of node u.*

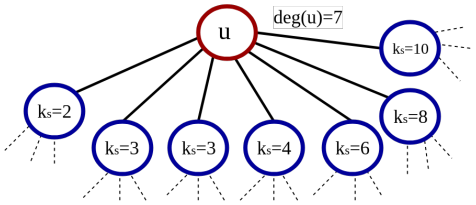
Proof. Let's assume that h-index of $(k_s(v)|\forall v \in ngh(u))$ is h then there are at least h nodes having shell-index equal to or greater than h as per the definition of h-index.

Now, we will see how the shell-index of node u will be decided in k-shell decomposition method. In k-shell decomposition method, in i_{th} iteration all those nodes are removed who have exactly i connections with the nodes having the shell-index i or greater that i. Thus, the node u will be removed in h_{th} iteration if h of its neighbors have shell-index h or greater than h. This is nothing but the $h - index$ of node u based on the shell-indices of its neighbors as defined above. □

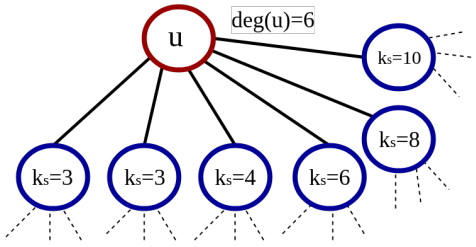
Next, we explain Theorem 1 using examples. The first example is shown in Figure 1(a) where node u has 8 neighbors having shell-indices 1, 2, 3, 3, 4, 6, 8, and 10. Now, we will see how the shell-index of node u will be determined during the k-shell decomposition method. All the iterations are shown in Figure 1. In the 1_{st} iteration, first of its neighbor will be removed and node u will be left with seven connections with the nodes having the shell-indices greater than 1, so, the node u will not be removed in the first iteration. In the 2_{nd} iteration, its second neighbor will be removed as it has shell-index 2, but still, the node u has six connections with the higher shells, so, it will not be removed. In the third iteration, its 3rd and 4th neighbors will be removed as both of these neighbors have shell-index 3. The node u still has four connections with the higher shells, so it will not be removed. In the 4th iteration, 5th of its neighbor having shell-index 4 will be removed, and now the node u has only three connections with the higher shells, so, as per the k-shell decomposition method, node u will also be removed in 4th iteration. So, the shell-index of node u is 4 that is nothing but the $h-index$ of the shell-indices of its neighbors.



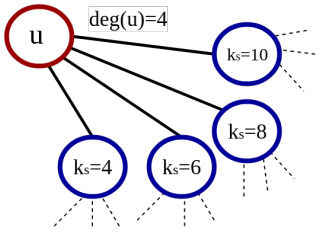
(a) Initial subgraph having all connections of node u



(b) After Iteration 1



(c) After Iteration 2



(d) After Iteration 3

Figure 1: Example 1: Estimate shell-index of node u while applying k-shell decomposition algorithm

Similarly, in Figure 2, node u has degree 7, and the shell-indices of its

neighbors are 1, 2, 2, 3, 3, 3, and 5. Now in the 1st iteration of k-shell decomposition method, 1st of its neighbor will be removed. Node u still have 6 connections with the higher shells, so, it will not be removed. In the 2nd iteration, 2nd and 3rd of its neighbors will be removed and node u still have 4 connections with the higher shells, so, it will not be removed. In the 3rd iteration, 4th, 5th, and 6th of its neighbors will be removed, and, now node u have only one connection with the higher shell, so it will also be removed in the same iteration. Thus, it has shell-index 3.

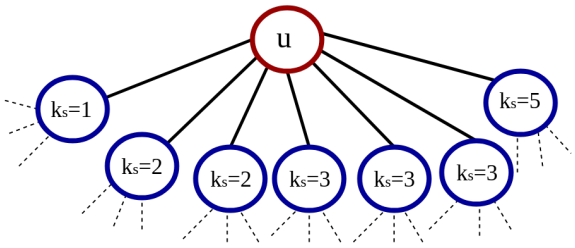


Figure 2: Example 2: A subgraph of a network to explain shell-index Computation using shell-indices of neighbors

In Figure 3, node u has degree 6 and the shell-indexes of its neighbors are 10, 11, 11, 13, 15, and 25. In 1st, 2nd,..., 5th iteration, the node u will not be removed as it has 6 connections with the higher shells. In the 6th iteration, node u will be removed as it has degree 6, so, the shell-index of node u is 6, i.e., the h-index of the shell-indices of its neighbors.

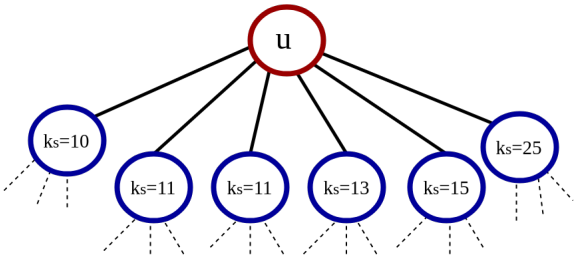


Figure 3: Example 3: A subgraph of a network to explain shell-index Computation using shell-indices of neighbors

Shell-Index Estimation

Using Theorem 1, the shell-index of a node can be estimated if the shell-indices of its neighbors are known. But in real life applications, the shell-indices of the neighbors will not be known. We know that the shell-index of a node is bounded above by its degree, $k_s(u) \leq d(u)$. So, to estimate the shell-index of node u, we can consider the degrees of its neighbors in place of their shell-indices. This is nothing but the $h-index$ of node u as defined in Section 3.1. To further improve the estimation, we consider the $h-index$ of its neighbors as $k_s(v) \leq h-index(v) \leq d(v)$ and this is nothing but the $h^2-index$ of the node. Thus the shell-index of a node can be estimated using its $h^2-index$. Next, we will study the performance of the proposed estimator in real-world networks.

Pseudo-code is given in Algorithm 1, where $ngh(u)$ is the list of the neighbors of node u. The proposed estimator can be further improved if we compute the $h^3-index$ of the node but in the results section, we show that $h^2-index$ is itself a good estimator. It can be computed faster and requires less neighborhood information than to compute the $h^3-index$ of the node.

Algorithm 1 $Compute\_h^2-index(G,u)$

Take a list ngh_hindex, $ngh\_hindex = [\,]$
for each v in $ngh(u)$ **do**
  add $h-index(v)$ in ngh_hindex
end for
return $h-index(ngh\_hindex)$

4.1 Results and Discussion

We study the performance of $h^2-index$ versus shell-index on the following real-world networks.

Datasets

1. **Astro-Ph:** This is the co-authorship network of ArXiv's Astrophysics (Astro-ph) publications where authors are represented by nodes, and there is an edge between two nodes if the corresponding authors have published together [65]. It contains 14845 nodes and 119652 edges.

2. **Buzznet:** Buzznet is a subgraph extracted from a social networking site that is used to share photo, journal, and video [66]. It consists of 101163 nodes and 2763066 edges.

3. **Cond-Mat:** This is the co-authorship network of ArXiv's condensed matter section [65]. This dataset covers all papers from January 1993 to April 2003 (124 months). It contains 13861 nodes and 44619 edges.

4. **DBLP:** This is a coauthorship network extracted from DBLP computer science bibliography, where edge denotes that the autors have common publications [67]. This network contains 317080 nodes and 1049866 edges.

5. **Digg:** This friendship network was extracted from Digg website in 2009 [68]. It contains 261489 nodes and 1536577 edges.

6. **Enron:** This network is the email communication network of the employees of Enron organization from 1999 to 2003 [69]. Nodes of the network are email addresses and there is an edge between two nodes if they have communicated at least once. The dataset has 84384 nodes and 295889 edges.

7. **Facebook:** This dataset is an induced subgraph of Facebook [70], where users are represented by nodes and friendships are represented by edges. It contains 63392 nodes and 816831 edges.

8. **FB-Wall:** This is a network where nodes are Facebook users and there is an edge between two users if any one of them post on the Facebook-wall of other user [70]. It contains 43953 nodes and 182384 edges.

9. **Foursquare:** Foursquare is a location-based social networking software for mobile devices that can be accessed using GPS. This dataset is an induced subgraph of friendships of Foursquare [66]. It contains 639014 nodes and 3214985 edges.

10. **Gowalla:** This friendship network is extracted from a location based social network called, Gowalla [71]. This was used to shared the locations among its users. It contains 196591 nodes and 950327 edges.

The experimental results are shown in Table 1. First, we compute the monotonicity of shell-index and $h^2 - index$. Ideally, if a node has influential

power different from other nodes, it should be assigned a unique index value. In k-shell decomposition method, all the nodes which are pruned at one level are assigned the same shell-index value. Researchers have shown that they have different influential power and should have been assigned different values [9–11]. So, a good centrality measure will assign the same value to fewer nodes and assign more unique values. This characteristic of the measure can be computed using the monotonicity [16]. This is defined as,

$$M(R) = \left(1 - \frac{\sum_{r \in R} n_r(n_r - 1)}{n(n-1)}\right)^2$$

where R denotes the ranking values of all the nodes based on any given centrality measure, n represents the size of R i.e. the number of nodes in our case, and n_r represents the number of nodes having rank r. If all nodes have the same rank, the monotonicity (M) of the ranking is 0, and it is not a valid ranking measure. If each node has a unique rank then the monotonicity (M) is 1, and it shows that the given ranking measure is perfect. The results in Table 1 show that the monotonicity of $h^2 - index$ is either same or slightly better than the shell-index.

Next, we study the correlation of shell-index and $h^2 - index$ with the spreading power of the node. The spreading power of a node is computed by executing the SIR model (SIR model is defined in Section 3.2) 100 times and taking the average number of infected nodes. In experiments, the infection probability λ is taken as $\lambda = \langle d \rangle / (\langle d^2 \rangle - \langle d \rangle) + 0.01$. To study the correlation, we compute Kendall's Tau (τ), Pearson (r), and Spearman (ρ) correlation coefficients. The results in Table 1 show that the correlation of $h^2 - index$ with the spreading power is either as good as the correlation of shell-index with the spreading power or better.

Next, we study how the correlation of shell-index and $h^2 - index$ with the spreading power changes as we vary the infection probability. The results are shown for Astro-physics collaboration network and Facebook wall social interaction network in Figure 4. The epidemic threshold value (λ_c) for Astro-Ph and FB-Wall network is 0.02 and 0.04 respectively, so, the considered range of the infection probability is taken $0.05 - 0.14$, i.e., greater than the λ_c for both the networks. The results show that $h^2 - index$ has good correlation with varying infection probabilities. We also observe that the correlation of both the centrality measures decreases as we increase the infection probability that is not desirable. It helps to conclude that the pro-

Table 1: Performance of Shell-Index (k_s) and $h^2 - index$ using monotonicity and SIR spreading model

Network	Ref	Nodes	Edges	Monotonicity		k_s vs. SIR			$h^2 - index$ vs. SIR		
				k_s	$h^2 - Index$	Kendall	Pearson	Spearman	Kendall	Pearson	Spearman
Astro-Ph	[65]	14845	119652	0.89	0.89	0.51	0.67	0.67	0.52	0.67	0.68
Buzznet	[66]	101163	2763066	0.93	0.93	0.21	0.28	0.30	0.21	0.28	0.30
Cond-Mat	[65]	13861	44619	0.75	0.76	0.55	0.69	0.69	0.56	0.70	0.70
DBLP	[67]	317080	1049866	0.74	0.75	0.49	0.61	0.61	0.49	0.62	0.62
Digg	[68]	261489	1536577	0.45	0.45	0.48	0.60	0.59	0.48	0.60	0.59
Enron	[69]	84384	295889	0.30	0.30	0.45	0.58	0.55	0.45	0.58	0.55
Facebook	[70]	63392	816831	0.91	0.91	0.49	0.64	0.65	0.49	0.64	0.65
Fb-Wall	[70]	43953	182384	0.76	0.77	0.62	0.75	0.76	0.62	0.75	0.76
Foursquare	[66]	639014	3214985	0.50	0.50	0.54	0.66	0.65	0.54	0.66	0.65
Gowalla	[71]	196591	950327	0.73	0.74	0.59	0.71	0.72	0.59	0.71	0.72

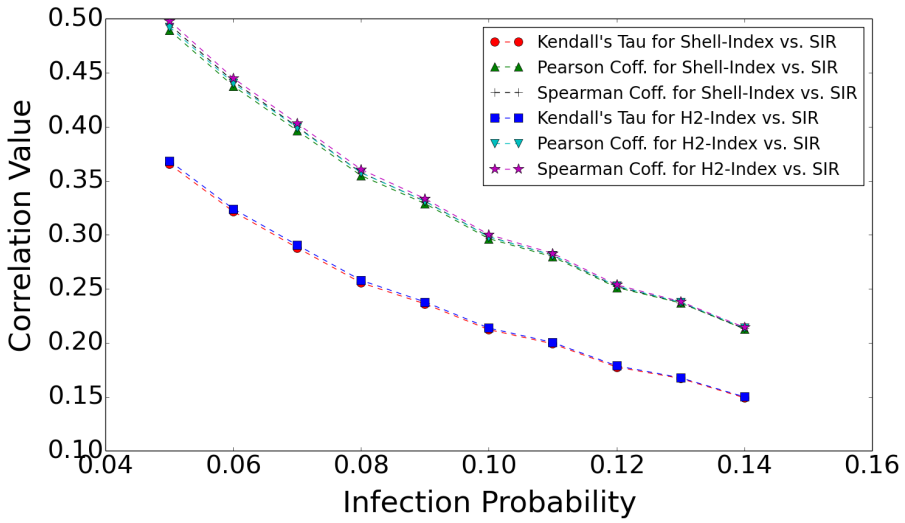
posed centrality measure can be further improved using other available local information of the node.

Thus we observe that $h^2 - index$ is a better centrality measure than the shell-index in real-world networks. $h^2 - index$ has an advantage over shell-index as it is a local centrality measure and can be computed for a node using local neighborhood information without collecting the entire network.

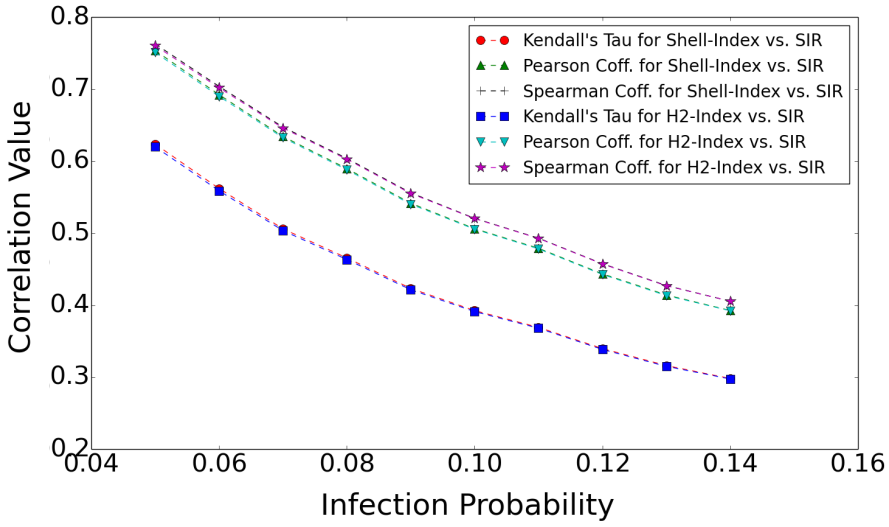
5 Section 2: Hill-Climbing based approach to Identify Top-Ranked Nodes

In many real-life applications, we need to identify top influential nodes to spread the information faster. For example, if a marketing company wants to provide free samples of their products, they would like to find out the influential people in the network who can help them to spread the word faster about their product. Similarly, if someone wants to spread a virus using Internet network, she would like to find out and infect the node having the highest spreading power.

With time, the size of real-world networks is growing very fast, and it is not feasible to collect the entire network to identify the top influential nodes. So, we need to propose methods to identify these nodes using local information without having the global information. By the global information, we mean the network properties which need the entire network to be computed like network size, average degree, the highest $h^2 - index$ of the network, and so on. Researchers have proposed various sampling-based methods to estimate global properties of the network like network size [72], global clustering



(a) Astro-Ph Collaboration Network



(b) FB-Wall Social Interaction Network

Figure 4: Shell-Index and $h^2 - index$ correlation with spreading power for varying infection probability a. Astro-Ph and b. FB-Wall network

coefficient [73], average degree [74], degree distribution of the network [75], and so on.

In this work, we have shown that the influential power of a node can be computed using its $h^2-index$, i.e. a local measure. For a given node, even if we compute its $h^2-index$ locally, we do not know how influential this node is in the entire network, and whether this node belongs to the highest influential nodes or not. In [20], Gupta et al. proposed Hill-Climbing based methods that can be used to hit the highest shell-index nodes faster in a network. In the proposed method, the random walker starts from a periphery node and move to one of its neighbor having the highest shell-index until a top node is found. The proposed method cannot be applied in practice as the shell-index is a global centrality measure, but the results show that using the proposed Hill-Climbing based random walk, the most influential nodes can be hit in a very small number of steps.

We modify the proposed methods to identify the highest $h^2-index$ node in the network. In rest of the paper, the *top-ranked nodes* refer to the nodes having the highest $h^2-index$ in the network. The algorithm is given in Algorithm 2. The input of the algorithm are G, u, k, and $maxindex$ where G is the given network, u is the seed node from which the crawler starts crawling the network, k is the repeat-count that shows how many times the crawler will restart the walk from a randomly chosen neighbor of the current node if it is stuck to the local maxima, and $maxindex$ is the maximum $h^2-index$ in the given network. A node is called local maxima if its $h^2-index$ is higher than all of its neighbors. The nodes having the highest $h^2-index$ in the network are called global maxima. For the clarification, a local maxima can also be the global maxima. In Algorithm 2, $ngh(u)$ represents the set of all the neighbors of node u. $randomchoice(list)$ function returns a random element from the given list.

The algorithm works in the following manner. The crawler starts from the given node u and it moves to one of its neighbors that has not been visited before and has the highest $h^2-index$. The crawler keeps moving until it hits the local maxima. If this node has the highest $h^2-index$, the algorithm exits, else the crawler jumps to one of its non-visited neighbors uniformly at random and the repeat-count is increased by one. The same procedure is repeated until the highest $h^2-index$ node is identified or the repeat-count reaches its maximum value. If the algorithm reaches to maximum repeat-count without finding out the highest $h^2-index$, it returns "The algorithm is failed to find out the top-ranked node."

Algorithm 2 $IndexBasedHillClimbing(G, u, k, maxindex)$

Take a list $visited\_nodes$ and $visited\_nodes = [\,]$
$num\_of\_steps = 0$
$repeat\_count = 0$
$current\_node = u$
$next\_node = u$
add u in $visited\_nodes$
$flag = True$
while $flag == True$ **do**
  for *each* v *in* $ngh(current\_node)$ **do**
    if $h^2 - index(v) \geq h^2 - index(next\_node)$ *and* $v \notin visited\_nodes$ **then**
      $next\_node = v$
    end if
  end for
  if $next\_node == current\_node$ **then**
    if $h^2 - index(next\_node) == maxindex$ **then**
      $flag = False$
    else if $repeat\_count < k$ **then**
      $next\_node = randomchoice_{v \notin visited\_nodes} ngh(current\_node)$
      $repeat\_count = repeat\_count + 1$
    else
      Print "The algorithm is failed to find out the top-ranked node."
      $flag = False$
    end if
  end if
  if $flag == True$ **then**
    add $next\_node$ in $visited\_nodes$
    $current\_node = next\_node$
    $num\_of\_steps = num\_of\_steps + 1$
  end if
end while
return $h^2 - index(current\_node)$

The proposed method is further modified as shown in Algorithm 3 named $IndexAndDegreeBasedHillClimbing(G, u, k, maxindex)$. In this method, the crawler will move to one of the highest degree nodes among the non-visited neighbors having the highest $h^2 - index$. Except this, there is one more change; once the algorithm stuck to local maxima, the crawler moves to one of its non-visited neighbors having the highest degree. The highest degree node has a high probability to be connected with the top-ranked nodes, and so, this algorithm will work faster and better than the first one.

Discussion

We implement the proposed methods on real-world networks, and the results are shown in Table 2. In the experiments, the algorithm is executed from all non top-ranked nodes (the nodes not having the highest $h^2 - index$), and the number of steps taken to hit a top-ranked node are averaged to compute the average number of steps. In experiments, the value of repeat-count (k) is set to 50. In Table 2, the average and the standard deviation of the number of steps are shown in $Avg(steps)$ and $Std(steps)$ columns. The $Avg(count)$ shows the average of the number of repeat-count that algorithm takes when it is stuck to a local maxima. The $Algo\ Failed(\%)$ shows how many times the algorithm is not succeeded to hit the top-ranked node in the given repeat-count. The algorithm might succeed if we increase the value of repeat-count.

The results show that on an average a top-ranked node can be reached in very few steps (3-213 in the considered datasets), and the average value of the repeat-count is 0-10. The algorithm is failed in very few cases for the repeat-count 50. In the case of $IndexAndDegreeBasedHillClimbing(G, u, k, maxindex)$ Algorithm, the average number of steps and the average repeat-count is reduced but the probability to fail the algorithm is increased in the given repeat-count as the crawler always moves to higher degree nodes and ends up following the same path that leads to stuck to the local maxima. In DBLP network the probability of failure is much higher when we apply degree based hill-climbing approach as the algorithm is mostly stuck to a local maxima due to following the same route and not able to hit the global maxima in the given repeat-count. This highly depends on the network structure and not on its size.

When we apply this algorithm in practice, we do not know the value of the highest $h^2 - index$ ($maxindex$), so, the algorithm needs to be repeated

Algorithm 3 *IndexAndDegreeBasedHillClimbing*$(G, u, k, maxindex)$

Take a list *visited_nodes* and *visited_nodes* = [ ]
$num\_of\_steps = 0$
$repeat\_count = 0$
$current\_node = u$
$next\_node = u$
add u in *visited_nodes*
$flag = True$
while $flag == True$ **do**
  for *each* v *in* $ngh(current\_node)$ **do**
    if $h^2 - index(v) \geq h^2 - index(next\_node)$ *and* $v \notin visited\_nodes$ **then**
      $next\_node = v$
    end if
  end for
  for *each* v *in* $ngh(current\_node)$ **do**
    if $h^2 - index(v) == h^2 - index(next\_node)$ *and* $deg(v) \geq deg(next\_node)$ *and* $v \notin visited\_nodes$ **then**
      $next\_node = v$
    end if
  end for
  if $next\_node == current\_node$ **then**
    if $h^2 - index(next\_node) == maxindex$ **then**
      $flag = False$
    else if $repeat\_count < k$ **then**
      for *each* v *in* $ngh(current\_node)$ **do**
        if $deg(v) \geq deg(next\_node)$ *and* $v \notin visited\_nodes$ **then**
          $next\_node = v$
        end if
      end for
      $repeat\_count = repeat\_count + 1$
    else
      Print "The algorithm is failed to find out the top-ranked node."
      $flag = False$
    end if
  end if
  if $flag == True$ **then**
    add *next_node* in *visited_nodes*
    $current\_node = next\_node$
    $num\_of\_steps = num\_of\_steps + 1$
  end if
end while
return $h^2 - index(current\_node)$

few more times when it is stuck to a local maxima to make sure that the node returned by the algorithm has the actual global maxima and not the local maxima. The results show that a smaller value of $repeat - count$ will suffice the purpose. The methods can be started from few randomly chosen nodes to avoid the local maxima and hit the top-ranked nodes with a high probability.

Table 2: Results for $IndexBasedHillClimbing(G, u, k, maxindex)$ and $IndexAndDegreeBasedHillClimbing(G, u, k, maxindex)$ Algorithms

Network	Nodes	$IndexBasedHillClimbing(G, u, k, maxindex)$				$IndexAndDegreeBasedHillClimbing(G, u, k, maxindex)$			
		Avg (*steps*)	*Std* (*steps*)	*Avg* (*count*)	*Algo Failed*(%)	*Avg* (*steps*)	*Std* (*steps*)	*Avg* (*count*)	*Algo Failed*(%)
Astro-Ph	14845	8.66	9.82	0.37	1.21	5.70	1.40	0.00	3.27
Buzznet	101163	3.20	0.69	0.00	0.00	3.20	0.69	0.00	0.00
Cond-Mat	13861	15.55	15.46	2.27	1.39	9.95	4.88	0.18	11.58
DBLP	317080	213.13	84.89	9.79	5.45	118.76	5.34	0.00	79.60
Digg	261489	4.09	1.50	0.02	0.03	4.03	0.85	0.00	0.15
Enron	84384	5.08	0.97	0.00	0.00	5.07	0.93	0.00	0.04
Facebook	63392	6.06	1.90	0.30	0.00	5.99	1.80	0.31	0.02
FB-Wall	43953	9.04	4.48	0.59	0.00	8.65	4.12	0.48	0.04
Foursquare	639014	10.84	0.38	0.00	0.00	10.84	0.38	0.00	0.00
Gowalla	196591	4.33	4.17	0.15	0.02	3.92	1.32	0.03	0.22

The proposed methods can be further improved by only computing the $h^2 - index$ of the neighbors that can be considered for the next step of the crawler instead of computing the $h^2 - index$ of all of its neighbors. A node having the degree lower than the $h^2 - index$ of the current node cannot have the $h^2 - index$ higher than the current node, so, all such neighbors can be discarded. This further fastens up the proposed method.

We further observe that in real-world networks, the subgraph of all top-ranked nodes (the highest $h^2 - index$ nodes) is connected. So, once we hit one top-ranked node, all top-ranked nodes can be identified. All these nodes can be used to spread the information faster in the network.

6 Section 3: Rank Estimation of a Node

In many real-life applications, users are interested to know the relative importance of the node that can be measured using the rank of the node. The classical method to compute the rank of a node, will compute the index value of all the nodes and will compare them to get the rank of the interested node. The time complexity of the proposed method is very high as it computes the

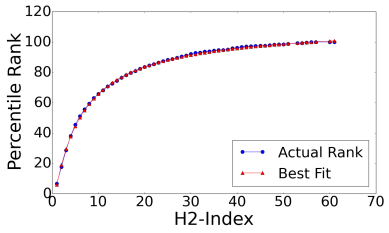
centrality value for all the nodes and its space complexity is also high as it requires the entire network.

We observe that percentile rank versus $h^2-index$ follows a unique pattern in all real-world networks as shown in Figure 5. The percentile rank of a node is computed as $PercentileRank(u) = \frac{n-R(u)+1}{n} * 100$, where n is the network size and $R(u)$ is the rank of node u based on its $h^2 - index$ value in the given network. We study this curve and find that 4-parameters logistic equation better fits the curve. The equation of the curve is given as $PercentileRank(u) = n + \frac{1-n}{1+\left(\frac{h^2-index(u)}{C_{mid}}\right)^p}$, where C_{mid} is a constant, and p denotes slope of the logistic curve at the middle point (also called hill's slope). The plots are shown in Figure 5, where black colored circles show the actual percentile rank of the nodes and the red colored triangles show best fit for the 4-parameter logistic equation. The best fit curve is plotted using scaled Levenberg-Marquardt algorithm [76] with 1000 iterations and 0.0001 tolerance. Once we estimate the parameters of the equation, the rank of a node can be computed in $O(1)$ time without computing the index values of all the nodes. A similar approach has been used to estimate the closeness rank of a node in $O(m)$ time [77].

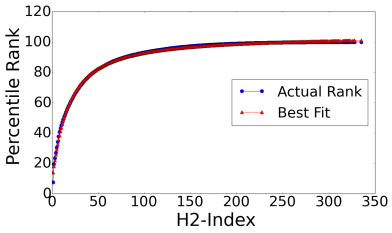
In Table 3, we show the average error and standard deviation in the estimated rank using the best fit curve. In future, we will propose methods to estimate the parameters of the best fit curve using the local information of the network. The parameters of the equation can be estimated if we know two points of the curve.

Table 3: Results for Rank Estimation using Best-fit Curve

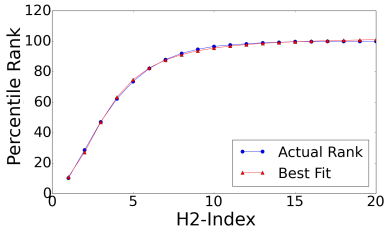
Network	Avg. Error	Std. Dev.
astro-ph	0.3585786572	0.2799515285
buzznet	0.4882475847	0.4604406318
condmat	0.6445420458	0.3868780324
dblp	0.1757178001	0.1434536392
digg	0.3560320321	0.2871690926
enron	0.5703748527	0.3135462032
fb	6.5857493125	5.6214277999
fb-wall	0.6480736627	0.433816658
foursquare	1.5378579372	1.4598262741
gowalla	0.1891852588	0.1160113617



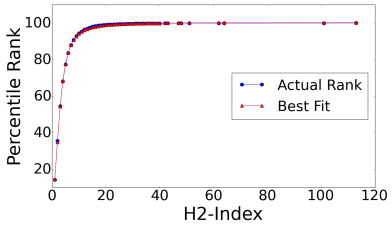
(a) Astro-Ph



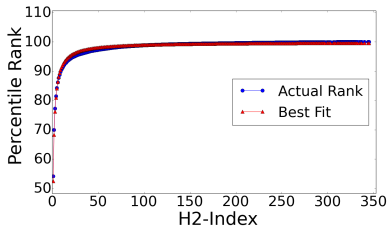
(b) Buzznet



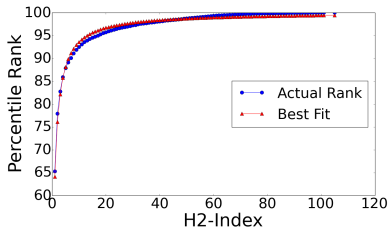
(c) Cond-Mat



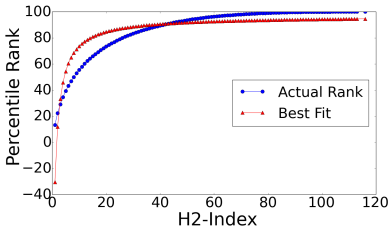
(d) DBLP



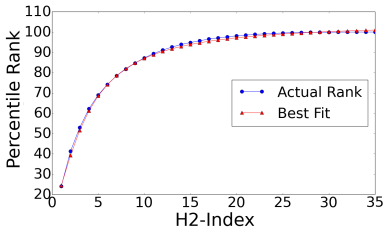
(e) Digg



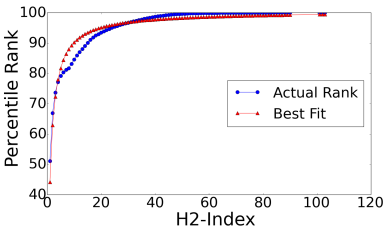
(f) Enron



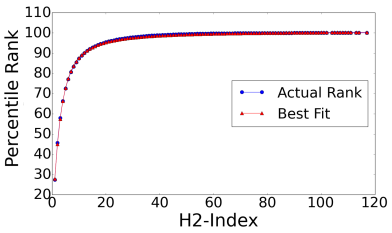
(g) Facebook



(h) FB-Wall



(i) Foursquare



(j) Gowalla

Figure 5: Percentile Rank versus $h^2 - index$

7 Conclusion

In this work, we have shown that $h^2-index$ is a good estimation of shell-index and can be used to identify influential nodes in real-world networks. We also observed that $h^2-index$ has better monotonicity than the shell-index and has better correlation with the spreading power of the nodes. Next, we used Hill-Climbing based methods to hit the top-ranked nodes in the given network. The results show that a top-ranked node can be found in very less number of steps using local information. In very few cases, the walk is trapped at the local maxima and does not hit the top-ranked node. We further observed that all top-ranked nodes form a connected subgraph and once we find one top-ranked node, all the most influential nodes can be identified. In future, we would like to propose mathematical bound to compute the required number of steps to hit a top-ranked node in the given scale-free real-world network. Next, we discuss a heuristic method to estimate the rank of the node. The estimation of the parameters of heuristic solution is left as a future task.

Acknowledgement: Authors would like to thank IIT Ropar HPC committee for providing the resources for performing the experiments.

References

[1] Stephen P Borgatti and Martin G Everett. Models of core/periphery structures. *Social networks*, 21(4):375–395, 2000.

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

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

[4] Vladimir Batagelj and Matjaž Zaveršnik. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification*, 5(2):129–145, 2011.

[5] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888–893, 2010.

[6] Akrati Saxena, SRS Iyengar, and Yayati Gupta. Understanding spreading patterns on social networks based on network topology. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, pages 1616–1617. IEEE, 2015.

[7] Sen Pei and Hernán A Makse. Spreading dynamics in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(12):P12002, 2013.

[8] Sen Pei, Lev Muchnik, José S Andrade Jr, Zhiming Zheng, and Hernán A Makse. Searching for superspreaders of information in real-world social media. *Scientific reports*, 4:5547, 2014.

[9] Ahmad Zareie and Amir Sheikhahmadi. A hierarchical approach for influential node ranking in complex social networks. *Expert Systems with Applications*, 93:200–211, 2018.

[10] Zhixiao Wang, Ya Zhao, Jingke Xi, and Changjiang Du. Fast ranking influential nodes in complex networks using a k-shell iteration factor. *Physica A: Statistical Mechanics and its Applications*, 461:171–181, 2016.

[11] An Zeng and Cheng-Jun Zhang. Ranking spreaders by decomposing complex networks. *Physics Letters A*, 377(14):1031–1035, 2013.

[12] Jian-Guo Liu, Zhuo-Ming Ren, and Qiang Guo. Ranking the spreading influence in complex networks. *Physica A: Statistical Mechanics and its Applications*, 392(18):4154–4159, 2013.

[13] Ying Liu, Ming Tang, Tao Zhou, and Younghae Do. Improving the accuracy of the k-shell method by removing redundant links: From a perspective of spreading dynamics. *Scientific reports*, 5:13172, 2015.

[14] Bonan Hou, Yiping Yao, and Dongsheng Liao. Identifying all-around nodes for spreading dynamics in complex networks. *Physica A: Statistical Mechanics and its Applications*, 391(15):4012–4017, 2012.

[15] Pavlos Basaras, Dimitrios Katsaros, and Leandros Tassiulas. Detecting influential spreaders in complex, dynamic networks. *Computer*, 46(4):0024–29, 2013.

[16] Joonhyun Bae and Sangwook Kim. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. *Physica A: Statistical Mechanics and its Applications*, 395:549–559, 2014.

[17] Nikolaj Tatti and Aristides Gionis. Density-friendly graph decomposition. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1089–1099. International World Wide Web Conferences Steering Committee, 2015.

[18] Jorge E Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences of the United States of America*, pages 16569–16572, 2005.

[19] Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H Eugene Stanley. The h-index of a network node and its relation to degree and coreness. *Nature communications*, 7, 2016.

[20] Yayati Gupta, Debarati Das, and SRS Iyengar. Pseudo-cores: The terminus of an intelligent viral meme's trajectory. In *Complex Networks VII*, pages 213–226. Springer, 2016.

[21] Marvin E Shaw. Some effects of unequal distribution of information upon group performance in various communication nets. *Journal of abnormal and social psychology*, 49(4):547–553, 1954.

[22] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica a: Statistical mechanics and its applications*, 391(4):1777–1787, 2012.

[23] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[24] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[25] Karen Stephenson and Marvin Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37, 1989.

[26] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine in: Seventh international world-wide web conference (www 1998), april 14-18, 1998, brisbane, australia. *Brisbane, Australia*, 1998.

[27] Yayati Gupta, Akrati Saxena, Debarati Das, and SRS Iyengar. Modeling memetics using edge diversity. In *Complex Networks VII*, pages 187–198. Springer, 2016.

[28] Antonios Garas, Frank Schweitzer, and Shlomo Havlin. A k-shell decomposition method for weighted networks. *New Journal of Physics*, 14(8):083030, 2012.

[29] Marius Eidsaa and Eivind Almaas. s-core network decomposition: A generalization of k-core analysis to weighted networks. *Physical Review E*, 88(6):062819, 2013.

[30] Bo Wei, Jie Liu, Daijun Wei, Cai Gao, and Yong Deng. Weighted k-shell decomposition for complex networks based on potential edge weights. *Physica A: Statistical Mechanics and its Applications*, 420:277–283, 2015.

[31] Rong-Hua Li, Jeffrey Xu Yu, and Rui Mao. Efficient core maintenance in large dynamic graphs. *Knowledge and Data Engineering, IEEE Transactions on*, 26(10):2453–2465, 2014.

[32] Paul Jakma, Marcin Orczyk, Colin S Perkins, and Marwan Fayed. Distributed k-core decomposition of dynamic graphs. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pages 39–40. ACM, 2012.

[33] Katerina Pechlivanidou, Dimitrios Katsaros, and Leandros Tassiulas. Mapreduce-based distributed k-shell decomposition for online social networks. In *Services (SERVICES), 2014 IEEE World Congress on*, pages 30–37. IEEE, 2014.

[34] Alberto Montresor, Francesco De Pellegrini, and Daniele Miorandi. Distributed k-core decomposition. *Parallel and Distributed Systems, IEEE Transactions on*, 24(2):288–300, 2013.

[35] Naga Shailaja Dasari, Ranjan Desh, and Mohammad Zubair. Park: An efficient algorithm for k-core decomposition on multicore processors. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 9–16. IEEE, 2014.

[36] Ahmet Erdem Saríyüce, Buğra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V Çatalyürek. Streaming algorithms for k-core decomposition. *Proceedings of the VLDB Endowment*, 6(6):433–444, 2013.

[37] Roberto Catini, Dmytro Karamshuk, Orion Penner, and Massimo Riccaboni. Identifying geographic clusters: A network analytic approach. *Research policy*, 44(9):1749–1762, 2015.

[38] Daniel Fricke and Thomas Lux. Core–periphery structure in the overnight money market: evidence from the e-mid trading platform. *Computational Economics*, 45(3):359–395, 2015.

[39] Paolo Barucca and Fabrizio Lillo. Disentangling bipartite and core-periphery structure in financial networks. *Chaos, Solitons & Fractals*, 88:244–253, 2016.

[40] Danielle S Bassett, Nicholas F Wymbs, M Puck Rombach, Mason A Porter, Peter J Mucha, and Scott T Grafton. Task-based core-periphery organization of human brain dynamics. *PLoS computational biology*, 9(9):e1003171, 2013.

[41] Hae-Jeong Park and Karl Friston. Structural and functional brain networks: from connections to cognition. *Science*, 342(6158):1238411, 2013.

[42] Nivedita Chatterjee and Sitabhra Sinha. Understanding the mind of a worm: hierarchical network structure underlying nervous system function in c. elegans. *Progress in brain research*, 168:145–153, 2007.

[43] Darko Obradovic and Stephan Baumann. A journey to the core of the blogosphere. In *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, pages 1–6. IEEE, 2009.

[44] Loet Leydesdorff, Caroline Wagner, Han Woo Park, and Jonathan Adams. International collaboration in science: The global map and the network. *arXiv preprint arXiv:1301.0801*, 2013.

[45] Clark Hu and Pradeep Racherla. Visual representation of knowledge networks: A social network analysis of hospitality research domain. *International Journal of Hospitality Management*, 27(2):302–312, 2008.

[46] Feng Luo, Bo Li, Xiu-Feng Wan, and Richard H Scheuermann. Core and periphery structures in protein interaction networks. In *Bmc Bioinformatics*, volume 10, page S8. BioMed Central, 2009.

[47] Kevin Crowston, Kangning Wei, Qing Li, and James Howison. Core and periphery in free/libre and open source software team communications. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 6, pages 118a–118a. IEEE, 2006.

[48] Chintan Amrit and Jos Van Hillegersberg. Exploring the impact of socio-technical core-periphery structures in open source software development. *journal of information technology*, 25(2):216–229, 2010.

[49] Pankaj Setia, Balaji Rajagopalan, Vallabh Sambamurthy, and Roger Calantone. How peripheral developers contribute to open-source software development. *Information Systems Research*, 23(1):144–163, 2012.

[50] Marcelo Cataldo and James D Herbsleb. Communication networks in geographically distributed software development. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 579–588. ACM, 2008.

[51] Gino Cattani, Simone Ferriani, and Paul D Allison. Insiders, outsiders, and the struggle for consecration in cultural fields: A core-periphery perspective. *American Sociological Review*, 79(2):258–281, 2014.

[52] Evelina Fedorenko and Sharon L Thompson-Schill. Reworking the language network. *Trends in cognitive sciences*, 18(3):120–126, 2014.

[53] John C Paolillo. Structure and network in the youtube core. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 156–156. IEEE, 2008.

[54] Jing Zhao, Guo-Hui Ding, Lin Tao, Hong Yu, Zhong-Hao Yu, Jian-Hua Luo, Zhi-Wei Cao, and Yi-Xue Li. Modular co-evolution of metabolic networks. *BMC bioinformatics*, 8(1):311, 2007.

[55] Akrati Saxena and SRS Iyengar. Evolving models for meso-scale structures. In *Communication Systems and Networks (COMSNETS), 2016 8th International Conference on*, pages 1–8. IEEE, 2016.

[56] Oludare Adeniji, David S Cohick, Ralucca Gera, Victor G Castro, and Akrati Saxena. A generative model for the layers of terrorist networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 690–697. ACM, 2017.

[57] Vishesh Karwa, Michael J Pelsmajer, Sonja Petrović, Despina Stasi, Dane Wilburne, et al. Statistical models for cores decomposition of an undirected random graph. *Electronic Journal of Statistics*, 11(1):1949–1982, 2017.

[58] Akrati Saxena and SRS Iyengar. Global rank estimation. *arXiv preprint arXiv:1710.11341*, 2017.

[59] Akrati Saxena, Vaibhav Malik, and SRS Iyengar. Rank me thou shalln't compare me. *arXiv preprint arXiv:1511.09050*, 2015.

[60] Akrati Saxena, Vaibhav Malik, and SRS Iyengar. Estimating the degree centrality ranking of a node. *arXiv preprint arXiv:1511.05732*, 2015.

[61] Akrati Saxena, Ralucca Gera, and SRS Iyengar. Observe locally rank globally. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 139–144. ACM, 2017.

[62] Akrati Saxena, Ralucca Gera, and SRS Iyengar. Degree ranking using local information. *arXiv preprint arXiv:1706.01205*, 2017.

[63] Akrati Saxena, Ralucca Gera, and SRS Iyengar. Fast estimation of closeness centrality ranking. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 80–85. ACM, 2017.

[64] Claudio Castellano and Romualdo Pastor-Satorras. Thresholds for epidemic spreading in networks. *Physical review letters*, 105(21):218701, 2010.

[65] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[66] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.

[67] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[68] Tad Hogg and Kristina Lerman. Social dynamics of digg. *EPJ Data Science*, 1(1):1–26, 2012.

[69] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Machine learning: ECML 2004*, pages 217–226. Springer, 2004.

[70] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42. ACM, 2009.

[71] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[72] Maciej Kurant, Carter T Butts, and Athina Markopoulou. Graph size estimation. *arXiv preprint arXiv:1210.0460*, 2012.

[73] Stephen J Hardiman and Liran Katzir. Estimating clustering coefficients and size of social networks via random walk. In *Proceedings of the 22nd international conference on World Wide Web*, pages 539–550. International World Wide Web Conferences Steering Committee, 2013.

[74] Anirban Dasgupta, Ravi Kumar, and Tamas Sarlos. On estimating the average degree. In *Proceedings of the 23rd international conference on World wide web*, pages 795–806. ACM, 2014.

[75] Bruno Ribeiro and Don Towsley. On the estimation accuracy of degree distributions from graph sampling. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 5240–5247. IEEE, 2012.

[76] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[77] Akrati Saxena, Ralucca Gera, and SRS Iyengar. A faster method to estimate closeness centrality ranking. *arXiv preprint arXiv:1706.02083*, 2017.