Identifying Influential Spreaders in Complex Networks Using Neighborhood Network Structure

Ziyi Zhang, Rong Yan*, Wei Yuan
College of Computer Science, Inner Mongolia University
Inner Mongolia Key Laboratory of Mongolian Information Processing Technology
National & Local Joint Engineering Research Center of Intelligent Information Processing Technology for Mongolian
Hohhot 010021, China
Email: csyanr@imu.edu.cn

***Abstract*—Identifying influential spreaders is a hot topic in complex network research. While centrality-based algorithms are easy to implement, they often have lower accuracy. Topology-based algorithms are effective for identifying network center influential spreaders but may not perform well in identifying peripheral influential spreaders. To address these problems, we propose a Neighborhood Structure Centrality (NSC) algorithm, which utilizes structural embedding and clustering to collect various network structural information and calculates node influence based on both node and neighborhood structural information. We compare the NSC algorithm with twelve baseline algorithms on six public datasets and four synthetic network datasets and demonstrate its higher accuracy.**

***Index Terms*—complex network, influential spreaders identification, network structure.**

I. Introduction

At present, researchers have conducted extensive studies on evaluating node influence, which can be mainly divided into two categories according to application scenarios: identifying influential spreaders and influence maximization. This paper focuses on identifying influential spreaders. We evaluated the influence of nodes by analyzing network structure and node characteristics, and ranked nodes according to their influence values, considering the top-ranked nodes as influential spreaders. In recent years, identifying influential spreaders has received extensive attention [1, 2]. The following provides an overview of the current state of research on algorithms for identifying influential spreaders.

In general, identifying influential spreaders algorithms can be roughly divided into two categories, including centrality-based algorithm and topology-based algorithm. Centrality-based algorithm also includes local centrality-based algorithms (LCAs) [3, 4], global centrality-based algorithms (GCAs) [5, 6], and semi-global centrality-based algorithms (SCAs) [7–9]. In general, centrality-based algorithms solely considering the features of nodes and neighbors will lead to ignoring differences in network position and structure, thereby affecting the accuracy and effectiveness of the algorithm. Meanwhile, much works focus on using the network topology to identify the influential spreaders, including k-shell algorithm [10], k-shell-based improved algorithms (KIAs) [11–13], k-shell-based hybrid algorithms (KHAs) [14–16], local global algorithms (LGAs) [17, 18], and structure-based graph neural network algorithms (SGNNs) [19, 20]. Overall, topology-based algorithms ignore high-influence nodes at the periphery of the network, which leads to the influential spreaders aggregation phenomenon. In addition, some hybrid algorithms use multiple topological attributes to evaluate the node influence, but they have to set parameters to adjust the weight of the attributes. Graph neural network algorithms use small network training models to predict all networks, which makes the models cannot learn enough network features. In brief, the recognition results of these algorithms are not stable enough.

In this paper, we identify influential spreaders by network topology. The main contributions of this paper are:

- Presents a new algorithm that uses the neighborhood structure to identify influential spreaders, alleviating the position limitation problem of topological algorithms.
- Taking structural embedding and clustering to collect comprehensive structural information, which alleviates the limited information problem of centrality-based algorithms and improves their accuracy.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 introduces our proposed algorithm. Section 4 reports the results and the analysis. Finally, in Section 5 we conclude the paper and present future work.

II. Related Work

In this section, we will briefly introduce related work. Firstly, we introduce the definition of the network and the two algorithms used in this paper. Secondly, we summarize several popular identifying influential spreaders algorithms.

A. Basic definition

Let an undirected unweighted network $G(V, E)$ is composed of $N = |V|$ nodes and $M = |E|$ edges, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of nodes in the network and $E = \{e_1, e_2, \ldots, e_m\}$ represents the set of edges.

In this paper, we use the GraphWave and KMeans++ algorithms. GraphWave [21] is a structural embedding algorithm

DOI reference number: 10.18293/SEKE2023-054

that utilizes spectral analysis to convert a network into a signal and learns node neighborhood structural features through spectral analysis techniques. This algorithm can capture nodes with similar local structures throughout the network, and generate node representations using these structures. KMeans++ [22] is a clustering algorithm that can partition nodes into k prespecified categories. Nodes within each category have high similarity, while the similarity between nodes from different categories is low.

B. Identifying influential spreaders algorithms

Kitsak et al. [10] proposed the k-shell algorithm, which categorizes nodes into different shells to evaluate their influence. The advantage of k-shell algorithm is its high efficiency and good recognition results. However, its disadvantages include position limitations and monotonicity problems, meaning that it cannot accurately evaluate the influence of nodes located at the network periphery and cannot distinguish the influence of nodes within the same shell. In recent years, many efforts have been devoted to addressing these problems, such as NC+ [12], G+ [13], KSH [15], and WKSD [16].

NC+ iteratively calculates the sum of neighbors' k-shell indexes to evaluate node influence, which improves the accuracy of the algorithm but exacerbates the position limitation problem. G+ refers to the gravity formula of physics, which regards the k-shell index value of the node as the mass, and the path length from the node to the neighbor as the distance to calculate the influence of the node. G+ achieves higher accuracy but still has a position limitation problem. KSH evaluates the influence of nodes based on k-shell power, degree, and distance. While it alleviates the position limitation problem, it sets a tunable parameter that may lead to unstable results. WKSD evaluates node influence using the k-shell index and degree. It is effective on incompletely connected networks, but the inclusion of two tunable parameters may lead to unstable results.

In addition to the related algorithms of k-shell, neural networks are also used to identify influential spreaders. RCNN [19] extracts each node's neighborhood subnetwork and constructs a micro-feature matrix, then it uses CNN to predict the influential spreaders. Based on RCNN, M-RCNNN [20] adds the community attributes and macro attributes to construct a three-channel node feature matrix, and automatically learns the weights of the three attributes to predict the influential spreaders. Although these two algorithms can automatically adjust parameters to address the instability problem, their accuracy is relatively low when applied to large-scale networks.

III. Our Algorithm

In this section, we introduce our proposed NSC algorithm in detail. Firstly, we give some basic notation definitions as shown in Table I. NSC algorithm includes three stages: (1) structure embedding, (2) clustering and cluster index allocation, and (3) node influence calculation.

TABLE I: Notation definitions.

Notation	Explanation
d_i	Degree of node v_i, $v_i \in V$ in G
k	Number of clusters
L_{uv}	The distance between nodes u and v
C_i	The i-th cluster
KC_i	The cluster index value of node v_i
$neigh\_inf(v)$	The influence of node v neighbors
$node\_inf(v)$	The influence of node v

A. First stage: structure embedding

In this stage, we try to capture the topology information of the network by using GraphWave algorithm [21]. GraphWave algorithm uses low-dimensional embedding to represent the node, where nodes with similar neighborhood structures have similar embeddings. Generally, GraphWave algorithm is divided into three steps: (1) spectral graph wavelet diffusion, (2) node mapping, and (3) sampling. Once the network structure embedding is completed, nodes with similar structures are closer, while nodes with different structures are farther apart from each other.

B. Second stage: clustering and cluster index allocation

In this stage, we designed a clustering index to distinguish structural importance, and to improve the efficiency of the algorithm, we chose the KMeans++ algorithm [22] for clustering. In this paper, we assume that the node influence within each cluster is similar. Firstly, we obtain k clusters by clustering, each cluster corresponds to one structure and ultimately captures the network of k structures. Next, we use the cluster index to represent the node importance within the cluster. In large-scale networks, the embedding vectors of nodes clustering boundaries are not apparent, which can cause incorrect clustering results for some nodes. Therefore, it is necessary to fine-tune the clustering. We first calculate the average degree $avg(C_i)$ of each cluster C_i, and sort all clusters $C=\{C_1, C_2, \cdots, C_k\}$ in ascending order by $avg(C_i)$. Then, the nodes whose degree is greater than $avg(C_i)$ in the cluster C_i are exchanged with the nodes whose degree is less than $avg(C_{i+1})$ in the set C_{i+1}. The number of exchange nodes is 30% of the number of nodes to be adjusted in smaller clusters. After this processing, we recalculate $avg(C_i)$ and sort C in ascending order, and give the index of cluster C_i as KC_i, the larger $avg(C_i)$ the higher the value of KC_i, $KC_i \epsilon [1, k]$.

Algorithm. 1 describes the details of this stage. Where the input is the embedding vector X of nodes and the number of clusters k. Line 1 is to cluster X. Lines 2-4 are to calculate the average degree of each cluster. lines 5-7 are to fine-tune the nodes in the clusters and sort the clusters. Lines 8-13 are to assign cluster indexes to each cluster.

C. Third stage: node influence calculation

In this stage, we calculate the node influence and then sort the nodes according to their influence within the network. In the experiment, we find that the clustering stage will produce deviation in large-scale networks or high-density networks.

Algorithm 1: Clustering and cluster index allocation

Input: Structural embedding X, Number of clusters k.
Output: $Rank[C, Score_{kc}]$.

KMeans++ clustering to get a set of k clusters C
for *each cluster* $C_i \in C$ **do**
    $d_i = average\_degree(C_i)$
end
Sort C in ascending order by d_i
Fine-tuning the nodes in the cluster
Recalculate d_i and resort C in ascending order by d_i
Set $score = 1$
for *each cluster* $C_i \in C$ **do**
    The index value of the i-th cluster $KC_i = score$
    Append C_i and KC_i to $Rank[C, Score_{kc}]$
    $score+ = 1$
end

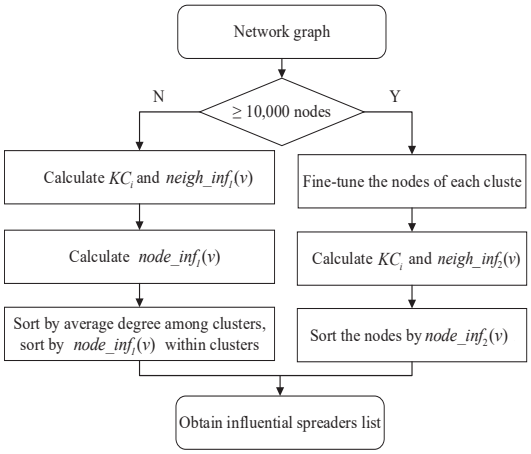


Fig. 1: The procedure of node influence calculation.

Thus, we design different node influence calculation strategies according to the network size. On the one hand, for small networks, we sort the nodes within each cluster first and then sort the clusters based on their influence. On the other hand, for larger networks, we directly sort all nodes according to their influence. Figure. 1 shows the procedure of node influence calculation. Node influence includes the influence of the node itself as well as the influence of its neighbors.

For networks with less than 10,000 nodes, we consider node influence as a combination of its degree and the influence of its nearest neighbors. Represent neighborhood influence using the cluster index value of neighbors. If the cluster index value of a node is high, it indicates that the local structure where it is located has a significant influence. Then, the influence of node v and the influence of node v's neighborhood are defined in Eq.(1) and Eq.(2) respectively.

$$node\_inf_1(v) = d_v + neigh\_inf_1(v) \tag{1}$$

$$neigh\_inf_1(v) = \sum_{j \in \eta(v)} KC_j \tag{2}$$

where $\eta(v)$ represents all nodes within the nearest neighborhood of node v. d_v is the degree of node v, KC_j is the cluster index value of node j.

Algorithm 2: Node influence calculation

Input: Network graph $G(V, E)$, $Rank[C, Score_{kc}]$.
Output: $Rank[V]$.

1 **if** *Node number* $|V| < 10{,}000$ **then**
2 **for** *each nodes* $v_i \in V$ **do**
3 $node\_inf_1(i) = d(v_i)$
4 **for** $v_j \in neighbors(v_i, 1)$ **do**
5 $node\_inf_1(v_i) += KC(v_j)$
6 **end**
7 **end**
8 **for** $C_i \in C$ **do**
9 Sort C_i in descending order with $node\_inf_1(v_i)$
10 **end**
11 $Rank[V] = C$
12 **end**
13 **else**
14 **for** *each nodes* $v_i \in V$ **do**
15 $neigh\_inf(v_i)$ = 0
16 **for** $v_j \in neighbors(v_i, r)$ **do**
17 $neigh\_inf_2(v_i)$ += $\frac{KC(w) * d(v_j)}{L(v_i, v_j)}$
18 **end**
19 **end**
20 **for** *each nodes* $v_i \in V$ **do**
21 $node\_inf(v_i)$=0
22 **for** $v_j \in neighbors(v_i, 1)$ **do**
23 $node\_inf_2(v_i)$ += $neigh\_inf(v_j)$
24 **end**
25 **end**
26 $Rank[V]$ = Resorting all nodes by $node\_inf_2(v_i)$.
27 **end**

For the network with more than 10,000 nodes, we define the influence of node v and the influence of node v's neighbors as shown in Eq.(3) and Eq.(4) respectively.

$$node\_inf_2(v) = \sum_{w \in \eta(v)} neigh\_inf_2(w) \tag{3}$$

$$neigh\_inf_2(w) = \sum_{j \in \eta_r(w)} \frac{KC_w * d_j}{L_{wj}^2} \tag{4}$$

where L_{vj} is the distance from node v to node j, $\eta_r(v)$ denotes the set of all nodes within r hops of node v, and $\eta(v)$ denotes the nearest neighbor node set of node v. In the experiment, we set r is 3.

Algorithm. 2 describes the details of this stage. The input of the algorithm is the network G and the clusters with their index, and the output is the influence sequence. Lines 1-12 compute the node influence for networks with less than 10,000 nodes, while lines 13-29 compute the node influence for networks with more than 10,000 nodes. The calculation process for both parts is similar. Lines 1-3 and 14-19 compute the self-influence of nodes, while lines 4-7 and 20-25 compute the influence of the neighbors of the nodes. Lines 8-11 and 26 sort the nodes.

D. Computing complexity

Given a network with N nodes, E edges, R neighbors within 3 hops of each node, k clusters, and 2-dimensional node embedding vectors. The time complexity of the network embedding phase is $O(S|E|)$, where S is the order Chebyshev polynomial approximation. The time complexity of the clustering and cluster index allocation phases is $O(2Nk + k)$, and the node influence calculation and sorting phases are

$O(NR + NlogN)$. Overall, the total time complexity of the NSC algorithm is $O(S|E| + 2Nk + k + NR + NlogN)$.

IV. Experiments

We conduct experiments on six real-world networks and four synthetic networks. In the experiment, we compare twelve baseline algorithms to verify the effectiveness of our proposed NSC.

A. Baselines

Degree centrality (DC) [3]: It evaluates the influence of a node by using the number of its nearest neighbors. **Betweenness centrality (BC)** [6]: It uses the number of shortest paths through the node to evaluate node influence. **Eigenvector centrality (EC)** [7]: It evaluates the influence of a node based on the importance of its neighbors. **K-shell (KS)** [10]: It evaluates the influence of a node based on its position in the network. **PageRank** [8]: It evaluates the influence of a node based on the probability of the node being visited during random walks. **Ksum** [23]: It evaluates the node influence by calculating the sum of the neighbor's degree. **Neighborhood coreness centrality (NC+)** [12]: It evaluates the influence of nodes by calculating the sum of the neighbor's k-shell. **Gravity index centrality (G+)** [13]: It refers to the gravity formula, where the k-shell of nodes is taken as the mass, and the shortest path length between nodes is taken as the distance to evaluate node influence. **Global and local structure (GLS)** [17]: It evaluates the influence of nodes by combining local information with global information. **Neighborhood entropy centrality (NEC)** [9]: It evaluates the node influence by calculating the sum of the entropies of the node's neighbors. **RCNN** [19]: It evaluates node influence by generating a neighborhood feature matrix for each node and applying a convolutional neural network. **Multi-channel RCNN (M-RCNN)** [20]: It evaluates node influence by generating three-channel node representations and using convolutional neural networks.

B. Dataset

We use six real-world networks and four synthetic networks for the experiments, including Jazz [24], USAir [25], Hamster[26], Power [27], PGP [28], Sex [29], and BA [30]. BA network synthesized by Python's networkx library. Table II gives the detailed description of each network (N denotes the number of nodes, M denotes the number of edges, β_{th} and β denote the infection threshold and infection rate of the network respectively. D is the diameter of the network, $<K>$ is the average degree of the network, and k is the number of clusters in NSC).

TABLE II: The description of all networks.

Network	N	M	β_{th}	β	D	$<K>$	k
Jazz	198	2,742	0.0266	0.04	6	27.697	5
USAir	332	2,126	0.0231	0.04	6	12.807	6
Hamster	2,426	16,631	0.0241	0.04	10	13.711	7
Power	4,941	6,594	0.3483	0.40	46	2.669	2
PGP	10,680	24,316	0.0559	0.06	24	4.554	7
Sex	15,810	38,540	0.0366	0.04	17	4.875	9
BA_1	1,000	9,900	0.0285	0.03	4	20	9
BA_2	1,000	14,775	0.0206	0.03	3	30	13
BA_3	2,000	19,900	0.0267	0.03	4	20	8
BA_4	2,000	29,775	0.0189	0.02	4	30	12

C. Evaluation criteria

1) Kendall correlation coefficient τ [31] : $\tau \in$[-1,1] is widely used to measure the correlation between two lists. After calculating the influence of each node using the SIR model [32], the accuracy of the algorithm is evaluated by comparing the ranking produced by the algorithm with the actual ranking generated by the SIR model. τ is mathematically defined as Eq.(5):

$$\tau = \frac{N_c - N_d}{N(N-1)/2} \tag{5}$$

where N_c and N_d denote the number of consistent and inconsistent pairs respectively, and N is the number of nodes in the list. Generally, if the value of τ is bigger, the algorithm considers effective.

2) Improvement percentage $\eta(\%)$ [33, 34]: To verify the improvement of NSC over the baseline algorithms, we conduct experiments using the η function (improve τ ratio), and η is mathematically defined as Eq.(6)

$$\eta(\%) = \begin{cases} \frac{\tau_{C(l)} - \tau_l}{\tau_l} * 100, & \tau_l > 0 \\ \frac{\tau_{C(l)} - \tau_l}{-\tau_l} * 100, & \tau_l < 0 \\ 0, & \tau_l = 0 \end{cases} \tag{6}$$

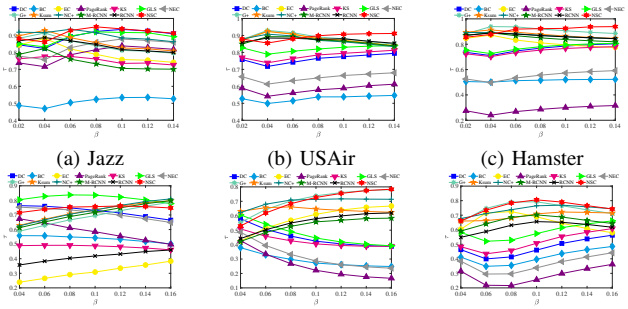
where $\tau_{C(l)}$ represents the τ value of NSC, and τ_l represents the τ value of other algorithms. $\eta(\%)>0$ means that the algorithm has been improved, $\eta(\%)<0$ means that NSC is worse than other algorithms, and $\eta(\%)=0$ means that NSC is not improved compared with other algorithms.

D. Experimental setup

In the process of using the SIR model to obtain the actual influence of nodes, 1,000 simulations were run on networks with less than 10,000 nodes, and 100 simulations were run on networks with more than 10,000 nodes. For the different structures of each network, we set the appropriate number of clusters k for NSC, and the specific values are shown in Table 2. The parameter L for the M-RCNN algorithm is set to 4 in the Power network and 28 in the other networks. The parameter L of RCNN is set to 28 in all networks. RCNN and M-RCNN use BA networks for training, so these two algorithms are not examined in the effectiveness experiment on synthetic networks.

E. Results and analysis

1) Effectiveness experiment on real-world network: We compared our proposed NSC with the twelve baseline algorithms for τ values, as shown in Fig. 2. It can be seen that NSC can achieve high correlation in most networks. Compared to the two structure-based neural network algorithms, the NSC performance was also more significant. Specifically, for Jazz,



(a) Jazz (b) USAir (c) Hamster

(d) Power (e) PGP (f) Sex

Fig. 2: Comparison results on six real networks.

USAir, Hamster, and Sex networks, NSC achieves the highest τ value. For larger networks like PGP and Sex, the value of τ for NSC is significantly higher than most algorithms.

Table III show the average τ value for each algorithm. In Jazz, NSC achieves the highest τ value of 0.9143. It is 1.21% better than the second-placed NC+ and 40.3% better than the worst BC. In USAir, the NSC is 0.5% higher than the second-placed G+ and 36.2% higher than the worst BC. In Hamster, the τ=0.9108 of the NSC is the highest. It is 0.19% better than the second-placed G+ and 40.3% better than the worst PageRank. In Power, the NSC is 6.81% lower than the first-place GLS and 1.91% higher than the third-place NC+.

In conclusion, NSC outperforms other baselines. It shows that the combination of topology and centrality algorithm is effective, and it can obtain more accurate node influence ranking.

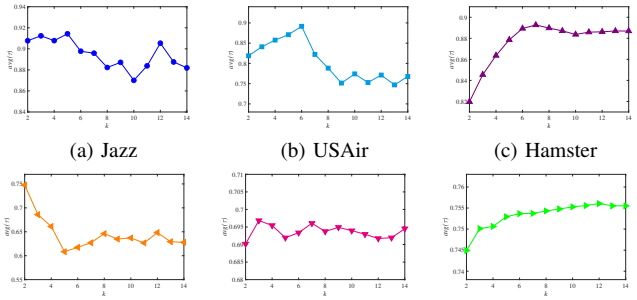
TABLE III: The average τ comparisons on six real networks.

Algorithm \ Network	Jazz	USAir	Hamster	Power	PGP	Sex
$\tau(DC)$	0.8938	0.7624	0.7638	0.7244	0.4506	0.4772
$\tau(BC)$	0.5112	0.5293	0.5143	0.5373	0.2913	0.4136
$\tau(EC)$	0.7984	0.8787	0.8163	0.3113	0.5829	0.6522
$\tau(PageRank)$	0.7917	0.5823	0.2850	0.5849	0.2556	0.2855
$\tau(KS)$	0.7533	0.7811	0.7480	0.4826	0.4242	0.5177
$\tau(GLS)$	0.8896	0.8215	0.7746	**0.8157**	0.4704	0.5885
$\tau(Nec)$	0.8329	0.6526	0.5515	0.7104	0.3202	0.3640
$\tau(G+)$	0.8456	0.8865	0.9089	0.6937	**0.7075**	0.7425
$\tau(Ksum)$	0.8610	0.8783	0.8779	0.7267	0.6425	0.6952
$\tau(NC+)$	0.9022	0.8713	0.8700	0.7285	0.6957	0.7373
$\tau(M-RCNN)$	0.7447	0.8667	0.8516	0.7138	0.5319	0.6592
$\tau(RCNN)$	0.8423	0.8721	0.8657	0.4147	0.5581	0.6181
$\tau(NSC)$	**0.9143**	**0.8916**	**0.9108**	0.7476	0.6959	**0.7537**

2) Cluster number k analysis: In this experiment, we explore the impact of a different number of clusters k on the τ value of NSC. Also, we provide guidance on choosing the value of k for different networks, as shown in Fig. 3. Except for Power, we can observe that NSC achieves the highest average τ value when k is around 6 in most networks. On a small network like Jazz, the average τ reaches the highest value when k is 5. On medium-sized networks such as Hamster, NSC achieves the best results when k is 6. For large networks such as PGP, the best results can be achieved when $k\geq 7$. Power is different from other networks, the average τ value of the NSC decreases with the increase of the k value. This is because all nodes in the Power network have a

TABLE IV: The average τ comparisons on BA networks.

Algorithm \ Network	BA_1	BA_2	BA_3	BA_4
$\tau(DC)$	0.8048	0.8635	0.8146	0.8692
$\tau(BC)$	0.7705	0.8051	0.7742	0.8210
$\tau(EC)$	0.6880	0.7206	0.6594	0.6744
$\tau(PageRank)$	0.7854	0.8538	0.7793	0.8529
$\tau(GLS)$	0.8307	0.8416	0.8338	0.8385
$\tau(Nec)$	0.8153	0.8643	0.8005	0.8623
$\tau(G+)$	**0.8637**	0.8731	**0.8607**	0.8772
$\tau(Ksum)$	0.6873	0.7219	0.6769	0.6836
$\tau(NC+)$	0.6873	0.7219	0.6769	0.6836
$\tau(NSC)$	0.8595	**0.8782**	0.8566	**0.8794**

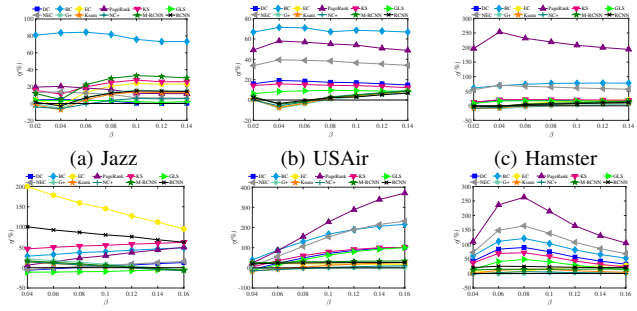


(a) Jazz (b) USAir (c) Hamster

(d) Power (e) PGP (f) Sex

Fig. 3: τ comparisons on different cluster k of NSC.

similar neighborhood structure. Therefore dividing the nodes into more clusters, which will degrade the performance of the algorithm. In summary, we recommend that the value of k should be set between 5 and 13. For small-scale networks set a smaller value of k and for large-scale or dense networks set a larger value of k. When the network structure is similar, k is set to 2.



(a) Jazz (b) USAir (c) Hamster

(d) Power (e) PGP (f) Sex

Fig. 4: Improvement percentage $\eta(\%)$ of NSC relative to twelve algorithms on τ value.

3) Improvement percentage $\eta(\%)$: The experimental results are shown in Fig. 4. In most networks, the NSC algorithm has obvious improvement over other algorithms. The percent improvement $\eta(\%)$ for BC, PageRank, and NEC algorithms is the highest. In Jazz, USAir, Hamster, PGP, and Sex networks, the value of $\eta(\%)$ is greater than zero when β>0.06. Especially in Sex, τ_{NSC} is relative to τ_{PR}, τ_{NEC}, τ_{BC}, τ_{DC} and τ_{KS}

exceeds 50%.

4) Effectiveness experiment on synthetic network: To further verify the effectiveness of NSC, we conduct experiments on BA networks. Since the nodes in the BA networks have at least $<K>$ edges, the shell values of all nodes after k-shell decomposition are the same, and the influence of nodes cannot be distinguished, so k-shell is not considered in this experiment. To verify the robustness of NSC on network scale and density, we set different node numbers and average degrees for the network. Table IV shows the average τ value of each algorithm in different scale BA networks. We observe that the NSC average τ on BA_1 and BA_3 is 0.4% lower than G+, but higher than other algorithms. In other networks, NSC outperforms other algorithms.

V. Conclusion and Future Work

In this paper, we propose an influential spreaders identification algorithm NSC. We compare NSC with 12 state-of-the-art baseline algorithms on 6 real networks and 4 synthetic networks, the results show that our proposed algorithm can obtain higher τ values than other algorithms in most networks. Specifically, compared with the 10 centrality algorithms, our algorithm is more stable when the network size and β value change. Meanwhile, compared with the 2 neural network algorithms based on neighborhood structure, NSC complies higher recognition accuracy, especially in larger networks. In future work, we will optimize the algorithm to automatically cluster based on network size, clustering coefficient, and structural difference metrics, so as to make it more widely applicable.

Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grant No. 61866029) and the Self-project Program of Engineering Research Center of Ecological Big Data, Ministry of Education.

References

[1] L. Caruccio, D. Desiato, and G. Polese, "Fake account identification in social networks," in *2018 IEEE international conference on big data (big data)*, 2018, pp. 5078–5085.

[2] F. Cerruto, S. Cirillo, D. Desiato, S. M. Gambardella, and G. Polese, "Social network data analysis to highlight privacy threats in sharing data," *Journal of Big Data*, vol. 9, no. 1, p. 19, 2022.

[3] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *Journal of mathematical sociology*, vol. 2, no. 1, pp. 113–120, 1972.

[4] L. Lü, T. Zhou, Q.-M. Zhang, and H. E. Stanley, "The h-index of a network node and its relation to degree and coreness," *Nature communications*, vol. 7, no. 1, pp. 1–7, 2016.

[5] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, no. 4, pp. 581–603, 1966.

[6] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[7] P. Bonacich and P. Lloyd, "Eigenvector-like measures of centrality for asymmetric relations," *Social networks*, vol. 23, no. 3, pp. 191–201, 2001.

[8] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[9] L. Qiu, J. Zhang, X. Tian, and S. Zhang, "Identifying influential nodes in complex networks based on neighborhood entropy centrality," *The Computer Journal*, vol. 64, no. 10, pp. 1465–1476, 2021.

[10] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, "Identification of influential spreaders in complex networks," *Nature physics*, vol. 6, no. 11, pp. 888–893, 2010.

[11] A. Zeng and C.-J. Zhang, "Ranking spreaders by decomposing complex networks," *Physics Letters A*, vol. 377, no. 14, pp. 1031–1035, 2013.

[12] J. Bae and S. Kim, "Identifying and ranking influential spreaders in complex networks by neighborhood coreness," *Physica A: Statistical Mechanics and its Applications*, vol. 395, pp. 549–559, 2014.

[13] L.-l. Ma, C. Ma, H.-F. Zhang, and B.-H. Wang, "Identifying influential spreaders in complex networks based on gravity formula," *Physica A: Statistical Mechanics and its Applications*, vol. 451, pp. 205–212, 2016.

[14] A. Sheikhahmadi and M. A. Nematbakhsh, "Identification of multi-spreader users in social networks for viral marketing," *Journal of Information Science*, vol. 43, no. 3, pp. 412–423, 2017.

[15] A. Namtirtha, A. Dutta, and B. Dutta, "Identifying influential spreaders in complex networks based on kshell hybrid method," *Physica A: Statistical Mechanics and its Applications*, vol. 499, pp. 310–324, 2018.

[16] A. "Namtirtha, A. Dutta, and B. Dutta, "Weighted kshell degree neighborhood: A new method for identifying the influential spreaders from a variety of complex network connectivity structures," *Expert Systems with Applications*, vol. 139, p. 112859, 2020.

[17] J. Sheng, J. Dai, B. Wang, G. Duan, J. Long, J. Zhang, K. Guan, S. Hu, L. Chen, and W. Guan, "Identifying influential nodes in complex networks based on global and local structure," *Physica A: Statistical Mechanics and its Applications*, vol. 541, p. 123262, 2020.

[18] A. Ullah, B. Wang, J. Sheng, J. Long, N. Khan, and Z. Sun, "Identifying vital nodes from local and global perspectives in complex networks," *Expert Systems with Applications*, vol. 186, p. 115778, 2021.

[19] E.-Y. Yu, Y.-P. Wang, Y. Fu, D.-B. Chen, and M. Xie, "Identifying critical nodes in complex networks via graph convolutional networks," *Knowledge-Based Systems*, vol. 198, p. 105893, 2020.

[20] Y. Ou, Q. Guo, J.-L. Xing, and J.-G. Liu, "Identification of spreading influence nodes via multi-level structural attributes based on the graph convolutional network," *Expert Systems with Applications*, p. 117515, 2022.

[21] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1320–1329.

[22] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[23] S. Pei, L. Muchnik, J. S. Andrade Jr, Z. Zheng, and H. A. Makse, "Searching for superspreaders of information in real-world social media," *Scientific reports*, vol. 4, no. 1, pp. 1–12, 2014.

[24] P. M. Gleiser and L. Danon, "Community structure in jazz," *Advances in complex systems*, vol. 6, no. 04, pp. 565–573, 2003.

[25] V. Batagelj and A. Mrvar, "Pajek-program for large network analysis," *Connections*, vol. 21, no. 2, pp. 47–57, 1998.

[26] J. Kunegis, "Konect: the koblenz network collection," in *Proceedings of the 22nd international conference on world wide web*, 2013, pp. 1343–1350.

[27] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[28] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Physical review E*, vol. 70, no. 5, p. 056122, 2004.

[29] L. E. Rocha, F. Liljeros, and P. Holme, "Simulated epidemics in an empirical spatiotemporal network of 50,185 sexual contacts," *PLoS computational biology*, vol. 7, no. 3, p. e1001109, 2011.

[30] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[31] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[32] Y. Moreno, R. Pastor-Satorras, and A. Vespignani, "Epidemic outbreaks in complex heterogeneous networks," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 26, no. 4, pp. 521–529, 2002.

[33] Y. Liu, M. Tang, T. Zhou, and Y. Do, "Identify influential spreaders in complex networks, the role of neighborhood," *Physica A: Statistical Mechanics and its Applications*, vol. 452, pp. 289–298, 2016.

[34] J. Wang, X. Hou, K. Li, and Y. Ding, "A novel weight neighborhood centrality algorithm for identifying influential spreaders in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 475, pp. 88–105, 2017.