Knowledge-Based Systems 198 (2020) 105893

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Identifying critical nodes in complex networks via graph convolutional networks

En-Yu Yu [a], Yue-Ping Wang [b], Yan Fu [a], Duan-Bing Chen [a,c,d,*], Mei Xie [c]

[a] Big Data Research Center, University of Electronic Science and Technology of China, Chengdu 611731, PR China
[b] Science and Technology on Complex Land Systems Simulation Laboratory, Beijing, 100012, PR China
[c] The Center for Digitized Culture and Media, University of Electronic Science and Technology of China, Chengdu 611731, PR China
[d] Union Big Data Tech. Inc., Chengdu 610041, PR China

ARTICLE INFO

Article history:
Received 17 December 2019
Received in revised form 29 February 2020
Accepted 7 April 2020
Available online 13 April 2020

Keywords:
Complex networks
Adjacency matrices
Critical nodes
Graph convolutional networks

ABSTRACT

Critical nodes of complex networks play a crucial role in effective information spreading. There are many methods have been proposed to identify critical nodes in complex networks, ranging from centralities of nodes to diffusion-based processes. Most of them try to find what kind of structure will make the node more influential. In this paper, inspired by the concept of graph convolutional networks(GCNs), we convert the critical node identification problem in complex networks into a regression problem. Considering adjacency matrices of networks and convolutional neural networks(CNNs), a simply yet effectively method named *RCNN* is presented to identify critical nodes with the best spreading ability. In this approach, we can generate feature matrix for each node and use a convolutional neural network to train and predict the influence of nodes. Experimental results on nine synthetic and fifteen real networks show that under Susceptible–Infected–Recovered (SIR) model, *RCNN* outperforms the traditional benchmark methods on identifying critical nodes under spreading dynamic.

1. Introduction

Nowadays, many real-world systems can be described by complex networks, the nodes represent the elements and edges stand for the relationships among different elements. It is well-known that many mechanisms such as cascading, spreading, and synchronizing are highly affected by a few critical nodes [1]. Therefore, identifying critical nodes in complex networks has caused great concern in the research community [2–5]. And the spreading ability of nodes has been applied to solve many problems, such as controlling propagation of information and rumors in social networks, ordering reputation of scientists [6] and identifying leaders in social networks [7].

Identifying critical nodes based on the structure and function of networks is a hot research topic on static networks and plays vital role in controlling the spread of epidemic or negative information [8,9]. Up to now, many methods have been proposed to measure the importance of nodes in static networks. Among of these, degree centrality [10], k-shell [11], H-index [12,13] and semi-local centrality [14] select influential nodes by node's own influences. Based on paths in networks, eccentricity centralities [15], betweenness [16] and closeness [17] have been proposed. And HITs [18], LeaderRank [19] and PageRank [20] are based on eigenvector. Zhang et al. have presented VoteRank [21] to identify a set of spreaders with the best spreading ability. Nodes vote in their neighbors in each turn, and the voting ability of neighbors of elected spreader will be decreased permanently. Li et al. proposed a classified neighbors algorithm [22] by assigning different weights for each class of neighbors and summing up the contributions of neighbors to quantify the nodal spreading capability and further to differentiate the influence of various nodes. Fei et al. [23] thought the mutual attraction between different nodes has been defined in complex network, which is inversely proportional to the square of the distance between two nodes. And proposed a method based on the inverse-square law to identify influential nodes. In order to adjust the decreasing factor adaptively according to the degree of global network, Guo et al. [24] proposed a new method named EnRenew based on node's entropy to select a set of influential nodes. Lv et al. [25]. proposed an average shortest path centrality to identify critical nodes, in which the relative change of the average shortest path of the whole network is taken into account.

Although there are many methods to measure the influence of nodes in complex networks, most of them can be regarded as a mission to find what kind of structure will make the node more influential. With the development of machine learning, some researchers have proposed the concept of GCNs [26,27],

* Corresponding author.
E-mail address: dbchen@uestc.edu.cn (D.-B. Chen).

https://doi.org/10.1016/j.knosys.2020.105893
0950-7051/

which aims at learning low-dimensional latent representation of graphs. These representations can be used as features for a wide range of tasks on graphs such as classification, clustering, link prediction and visualization. Inspired by this concept of GCNs, we convert the critical node identification problem in complex networks into a regression problem. Many traditional influential nodes mining algorithms usually directly use the adjacency matrix, which may contain both the intrinsic structure and the redundant information. In order to reduce the noise or redundant information and preserve the intrinsic structure information. By using adjacency matrices of networks and GCNs, we propose a simply yet effectively iterative method named *RCNN* to rank nodes. The performance of the proposed method is compared with degree centrality, betweenness centrality, k-shell and H-index in SIR model [28,29] on nine synthetic and fifteen real networks. The results show that the proposed method in this paper can effectively identify critical nodes which have greater impacts on information spreading in complex networks.

The rest of paper is organized as follows: The related works are presented in Section 2 and the background is presented in Section 3. The proposed method to identify critical nodes based on graph convolutional networks is defined in Section 4. Experiment results and the time complexity are analyzed and discussed in Section 5. Conclusions and future interested research topics are given in Section 6.

2. Related works

In recent years, with the wide application of machine learning, GCNs have been used in community detection [30,31], link prediction [32,33] and etiology of diseases [34]. GCNs are semi-supervised methods for graphs and usually can be trained with task-specific loss through back-propagation like standard CNNs. However, due to the lack of grid structures, standard convolution of image data or text data cannot be directly applied to graphics. So the difficulty of GCNs lies in how to extract grid structures from the graph.

Belkin et al. [35] first introduced convolution for graph data from spectral domain using the graph Laplacian matrix $\mathbf{L}$, which plays a similar role as the Fourier basis for signal processing [36]. Patchy–San [37] used Weisfeiler–Lehman kernel [38] to assort a unique order of nodes and arranged nodes in a line using this pre-defined order. Patchy–San defined a receptive field for each node v_i by selecting a fixed number of nodes from their k-step neighborhoods to mimic CNNs, and then adopted standard CNNs with proper normalization. Kipf et al. presented a scalable approach for semi-supervised learning on complex networks which is based on an efficient variant of convolutional neural networks [39]. In order to explore the impact of temporal information on the importance of the nodes, Qu et al. [40] proposed a temporal information gathering (TIG) process for evaluating the significance of the nodes in temporal networks. The key to the TIG process is that the importance of a node depends on the importance of its neighborhood. By utilizing a variant of recurrent neural network, Qi et al. [41] presented a deep autoencoding gaussian mixture model (DAGMM) to adapt context evolution and extract context features for nodes. In DAGMM, low-dimensional representations of nodes are generated by a deep autoencoder. All of these methods learn graph representations that can be used to infer implied graph properties. And in this paper, these properties are applied to identifying critical nodes under spreading dynamic.

3. Background

In this section, some background information about networks, convolutional neural networks and graph convolutional networks is given.

3.1. Network

A network G can be defined as a pair (V, E) with $V = \{v_1, \ldots, v_n\}$ the set of nodes and $E \subseteq V \times V$ the set of edges. Let n and m be the number of nodes and edges respectively. Each network can be represented by an adjacency matrix $\mathbf{A}$ of size $n \times n$, where $\mathbf{A}_{i,j} = 1$ if there an edge from node v_i to node v_j, otherwise $\mathbf{A}_{i,j} = 0$. A walk is a series of nodes in the network, where consecutive nodes are connected by an edge. A path is a walk with distinct nodes. We use $d(u, v)$ to denote the distance between node u and v, that is, the length of the shortest path between u and v. $\Gamma_l(u)$ is the l-hop neighbors of node u, that is, the length of the shortest path between u and a node in $\Gamma_l(u)$ is l.

3.2. Convolutional neural networks

CNNs are kind of Feedforward neural networks with convolutional computation and deep structure. It is one of the representative algorithms of deep learning. CNNs are an incredibly successful technology that has been applied to computer vision and natural language processing [42–45]. CNNs in essence are neural networks which use the convolution operation as one of its layers. A typical CNN consists of several convolution and pooling layers. The purpose of the first convolutional layer is to extract a common pattern found within localized regions of the input data. CNNs convolve learned filters over the input data, computing the inner product and outputting the result as tensors whose depth is the number of filters.

3.3. Graph convolutional networks

GCNs is a useful tool for processing non-Euclidean data such as: social networks, knowledge graphs and protein interaction networks. GCNs generalize the convolutional operation from traditional data (images or grids) to graph data. The key is to learn a function f to generate the representation of a node v_i by aggregating its own features $\mathbf{X}_i$ and neighbors' features $\mathbf{X}_j$ [39]. And a non-linear transformation is applied to the resultant outputs after feature aggregation. The final hidden representation of each node receives information from a further neighborhood by stacking multiple layers. GCNs play an important role in building up many other complex GCN models, including auto-encoder-based models [46], generative models [47], and spatial–temporal networks [48], etc.

4. Method

We aim to bring convolutional neural networks to bear on the problem of identifying critical nodes in complex networks. We propose a framework for learning representations for arbitrary networks. Similar to convolutional neural network for images, we construct locally connected neighborhoods from the input networks. These neighborhoods serve as the receptive fields of a convolutional architecture, allowing the framework to learn effective network representations. The details of RCNN are described as following steps:

(1) Feature Matrices and Labels: Firstly, find $L-1$ neighbors for each node, prefer the nearest neighbor (1-hop neighbors, 2-hop neighbors, etc.), if two nodes have equal hops, the node with larger degree has been chosen preferential. Secondly, generate subnetworks from the original network, the subnetwork G_u contains node u and its $L-1$ neighbors, and $\mathbf{A}^u$ is the adjacency matrix of G_u. Thirdly, let $u_0, u_1, \ldots, u_{L-1}$ be node u and its $L-1$

neighbors, $\mathbf{B}^u$ is the feature matrix of node u converted from $\mathbf{A}^u$ and its elements are defined as

$$B^u_{i,j} = \begin{cases} \mathbf{A}^u_{0,j} k_{u_j}, & i = 0, j = 1, 2, \ldots, L-1 \\ \mathbf{A}^u_{i,0} k_{u_i}, & i = 1, 2, \ldots, L-1, j = 0 \\ k_{u_i}, & i = j = 0, 1, 2, \ldots, L-1 \\ \mathbf{A}^u_{i,j}, & other \quad cases \end{cases} \tag{1}$$

where k_{u_i} is degree of node u_i in the original network. In addition, if the network is disconnected, we need expand $\mathbf{B}^u$ with zero padding. It is also expressed as $\mathbf{B}^u$ without causing confusion. For example, as shown in Fig. 1, let $L = 8$, we extract an enclosing subgraph for a target node $a(k_a = 4)$, and its neighbors $b(1-hop, k_b = 3)$, $c(1-hop, k_c = 1)$, $d(1-hop, k_d = 3)$, $e(2-hop, k_e = 3)$, $f(1-hop, k_f = 4)$, $g(2-hop, k_g = 1)$, $h(2-hop, k_h = 2)$. Then mark all nodes and $\mathbf{A}^a$ and $\mathbf{B}^a$ are:

$$\mathbf{A}^a = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B}^a = \begin{bmatrix} 4 & 4 & 3 & 2 & 1 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 & 1 & 0 & 1 \\ 3 & 0 & 3 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

If $L = 10$, we need expand $\mathbf{B}^a$ with zero padding as follows:

$$\mathbf{B}^a = \begin{bmatrix} 4 & 4 & 3 & 2 & 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} \\ 4 & 4 & 0 & 0 & 0 & 1 & 0 & 1 & \mathbf{0} & \mathbf{0} \\ 3 & 0 & 3 & 0 & 0 & 1 & 0 & 0 & \mathbf{0} & \mathbf{0} \\ 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 1 & 0 & 0 & 3 & 0 & 0 & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3}$$

Finally, obtain all nodes' infected scales by SIR spreading model as the labels. In this paper, training infection rate is μ_t/μ_c and testing (real) infection rate is μ/μ_c.

(2) Convolutional Architecture: The convolutional neural network in this paper have 2 convolutional layers, 2 pooling layers and 1 fully-connected layer. In the first convolutional layer, kernel size is 5×5, input channels is 1 and output channels is 16, stride is 1 and padding is 2. In the second convolutional layer, kernel size is 5×5, input channels is 16 and output channels is 32, stride is 1 and padding is 2. 2 pooling layers are 2×2 max pooling and the fully-connected layer is $32 * (L/4) * (L/4) \times 1$. The activation function is ReLU and the loss function is squared loss function.

(3) Ranking Nodes: Firstly, use training sets (generate feature matrices and labels for all nodes) to train the CNN model. Secondly, select arbitrary networks, generate feature matrices for all nodes without labels and get the score of these nodes by the trained CNN models. Finally, ranking the nodes by their scores.

5. Experiments and discussions

The performance of the *RCNN* is evaluated by SIR spreading model, and compared with well-known existing metrics such as degree centrality, betweenness centrality and k-shell in nine synthetic and fifteen real networks.

5.1. Evaluation metrics

SIR Spreading Model. In SIR model, each node is in one of three statuses: Susceptible(S), Infected(I) and Recovered(R). Susceptible nodes can be infected at each time step. Infected nodes have been infected and try to infect susceptible nodes in their neighbors with probability μ at each time step. Recovered nodes have been recovered from infected status and will never be infected by infected nodes again. At each time step, infected nodes will be recovered with probability β(for simplicity, $\beta = 1$ in this paper). The process terminates if there is not any infected node in network. We can set a node to be infected and the others to be susceptible to estimate the influence of a single node in the network. Let node v is infected and other nodes are susceptible firstly. F_v is defined as the number of infected nodes when spread process achieving steady state. We can use F_v to represent the finally infected scale of node v in this paper.

Kendall's tau. [49] To evaluate the performance of different ranking methods, we use Kendall's tau as a rank correlation coefficient, which is defined as

$$\tau(X, Y) = \frac{C - D}{n(n-1)/2}, \tag{4}$$

where X and Y are two different lists with length n. C and D is the numbers of concordant and discordant pairs between X and Y, respectively. For example, let $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are a set of joint ranks from X and Y. If $x_i > x_j$ and $y_i > y_j$ or $x_i < x_j$ and $y_i < y_j$, (x_i, y_i) and (x_j, y_j) is concordant pair. The pair is said to be discordant if $x_i > x_j$ and $y_i < y_j$ or $x_i < x_j$ and $y_i > y_j$. And the pair is neither concordant nor discordant if $x_i = x_j$ or $y_i = y_j$.

5.2. Benchmark methods

The degree centrality [10] is defined as

$$DC(u) = \frac{k_u}{n-1}, \tag{5}$$

where n is the number of nodes in G and k_u is the degree of node u.

The betweenness centrality [16] is defined as

$$BC(u) = \sum_{u \neq s, u \neq t, s \neq t} \frac{g^u_{st}}{g_{st}}, \tag{6}$$

where g_{st} is the number of the shortest paths between node s and t and g^u_{st} is the number of the paths which pass through u among all the shortest paths between s and t.

The k-shell [11] of node u is defined as the largest value k of a k-core containing node u. And a k-core is a maximal subgraph that contains nodes of degree k or more

The H-index [12] is defined as

$$h(u) = H(k_{j_1}, k_{j_2}, \ldots, k_{j_{k_u}}), \tag{7}$$

where $k_{j_1}, k_{j_2}, \ldots, k_{j_{k_u}}$ are degrees of node u's neighbors. H is an operator which acts on a finite number of reals $(x_1, x_2, \ldots, x_n)$ and returns an integer $y = H(x_1, x_2, \ldots, x_n) > 0$, where y is the maximum integer such that there exist at least y elements in $(x_1, x_2, \ldots, x_n)$, each of which is no less than y.

5.3. Datasets

In Datasets, we use six synthetic networks to train *RCNN* model, nine synthetic and fifteen real networks to test RCNN model. All synthetic networks are generated by BA model [50] (a graph of n nodes is grown by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree) with different number of nodes and edges.

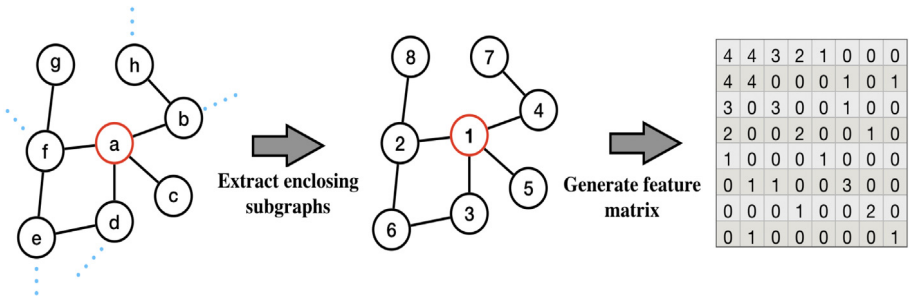

Fig. 1. An illustration of generating a feature matrix: (1) enclosing subgraph extraction; (2) subgraph pattern encoding.

Training Data Sets.

Train_n_k: A scale-free network generated by BA model with n nodes and average degree is k.

Testing Data Sets.

(1) Test_n_k: A scale-free network generated by BA model with n nodes and average degree is k. (2) NS [51]: A co-authorship network of scientists working on network theory and experiment. (3) USAir [52]: A network of flights between US airports in 2010. (4) Jazz [53]: A collaboration network between Jazz musicians. (5) Email [54]: An email communication network at the University Rovira i Virgili in Tarragona in the south of Catalonia in Spain. (6) Oz [55]: A network contains friendship ratings between 217 residents living at a residence hall located on the Australian National University campus. (7) Router [56]: A network contains autonomous systems of the Internet connected with each other. (8) Faa [57]: This network is constructed from the USA's FAA (Federal Aviation Administration) National Flight Data Center (NFDC), Preferred Routes Database. (9) Figeys [58]: A network of interactions between proteins in Humans, from the first large-scale study of protein–protein interactions in Human cells using a mass spectrometry-based approach.(10) Sex [59]: Network is of a bipartite sexual activity web community in which nodes are females (sex sellers) and males (sex buyers) and links between them are established when males write posts indicating sexual encounters with females. (11) Powergrid [60]: A network contains information about the power grid of the Western States of the United States of America. (12) Arenas [61]: A metabolic network of the roundworm Caenorhabditis elegans.(13) Chicago [62]: A road transportation network of the Chicago region (USA). (14) Hamster [63]: A network contains friendships and familylinks between users of the website hamsterster.com. (15) Vidal [64]: A network represents an initial version of a proteome-scale map of Human binary protein-protein interactions. (16) Stelzl [65]: A network represents interacting pairs of protein in Humans (Homo sapiens).

In Table 1, some statistical properties of real networks are listed. In this paper, these networks are both undirected and unweighted. NS, USAir, Jazz, Email, Oz, Router, Faa, Figeys, Sex, Powergrid, Arenas, Chicago, Hamster, Vidal and Stelzl are real-world networks which represent human interactions in diverse social systems and have different topological and characteristics. Table 2 shows the training time of six training data sets with different L. *RCNN* is trained for 500 epochs with a learning rate of 0.001 on each training data sets. All experiments are run on a single 1600 MHz GPU with 8 GB memory.

Table 1
The basic topological features of fifteen real-world networks. n is the number of nodes. m is the number of edges. $\langle k\rangle$ is the average degree. k_{max} is the maximum degree. $\langle c\rangle$ is the average clustering coefficient. H is the degree heterogeneity and $H = \frac{\langle k^2\rangle}{\langle k\rangle^2}$.

Networks	n	m	$\langle k\rangle$	k_{max}	c	H
NS	1461	2742	3.8	34	0.6936	1.8485
USAir	1574	17215	21.9	314	0.5042	5.1303
Jazz	198	2742	27.7	100	0.6174	1.3951
Email	1133	5451	9.6	71	0.2201	1.9421
Oz	217	1839	16.9	56	0.3627	1.2094
Router	5022	6258	2.5	106	0.0115	5.5031
Faa	1226	2408	3.9	34	0.0675	1.8727
Figeys	2239	6432	5.7	314	0.0399	9.7474
Sex	16730	39044	4.7	305	0	6.0119
Powergrid	4941	6594	2.7	19	0.0801	1.4504
Arenas	453	2025	8.9	237	0.6465	4.4850
Chicago	1467	1298	1.8	12	0	3.0592
Hamster	2426	16631	13.7	273	0.5376	3.1010
Vidal	3023	6149	4.1	129	0.0658	3.7960
Stelzl	1702	3155	3.7	95	0.0060	4.5375

Table 2
The training time (seconds) of six training data sets with different L.

Networks	$L = 8$	16	24	28	32	40
Train_1000_4	18.12	19.88	23.65	25.56	27.14	29.80
Train_1000_10	18.32	19.87	23.54	25.45	27.00	29.59
Train_1000_20	18.17	19.83	23.19	25.91	27.02	29.19
Train_3000_4	46.39	52.11	61.59	63.95	66.82	80.47
Train_3000_10	46.36	52.02	61.87	63.78	66.90	83.03
Train_3000_20	46.84	52.99	61.97	63.79	67.60	79.98

5.4. Parameters analysis

In order to evaluate the performance of *RCNN* on different training sets with different L, μ_t and μ. We perform parameter analysis on *RCNN*. In Fig. 2, we show the kendall's tau coefficients between the ranking scores and the real infected scales with $\mu_t/\mu_c = \mu/\mu_c = 1.5$ where $\mu_c = \frac{\langle k\rangle}{\langle k^2\rangle - \langle k\rangle}$. From Fig. 2, *RCNN* get excellent performances when the average degree of the training set is close to the average degree of the testing set, it means that *RCNN* performs best when the structure of the training and testing sets is most similar. And it can be seen that when the number of nodes more than 1000, the size of training sets has slightly impact on results. What is more, when using a training set with a lower average degree, it also performs good on a higher average degree testing set. But when the training set has higher average degree, it performs poorly on testing sets with lower average degree and varies heavy with L. In general, *RCNN* will performs well in most case and varies slightly with L when the average degree of training sets is not too large, such as 4.

Because the performance of *RCNN* is slightly related to L, in Fig. 3, we show the kendall's tau coefficients between the ranking scores and the real infected scales on different μ_t and μ with $L = 28$ and the training set being Train_1000_4. From Fig. 3, it can be seen that there is a high correlation coefficient in any position of these heat maps. When infection rate μ/μ_c and train infection rate μ_t/μ_c varies from 1.0 to 2.0, the performance of *RCNN* will vary slightly with the change of μ_t. In general, the change of μ_t has slightly influence on *RCNN*, and it performs well in most cases.

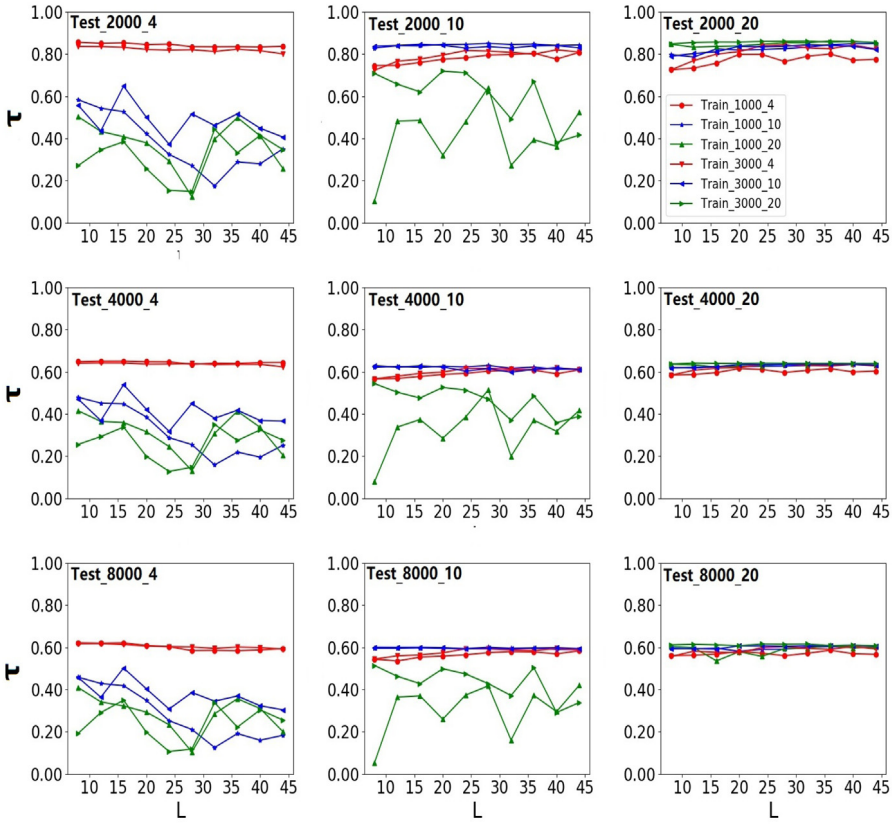

Fig. 2. The kendall's tau correlation coefficients between the ranking scores and real infected scales with varying L and training sets.

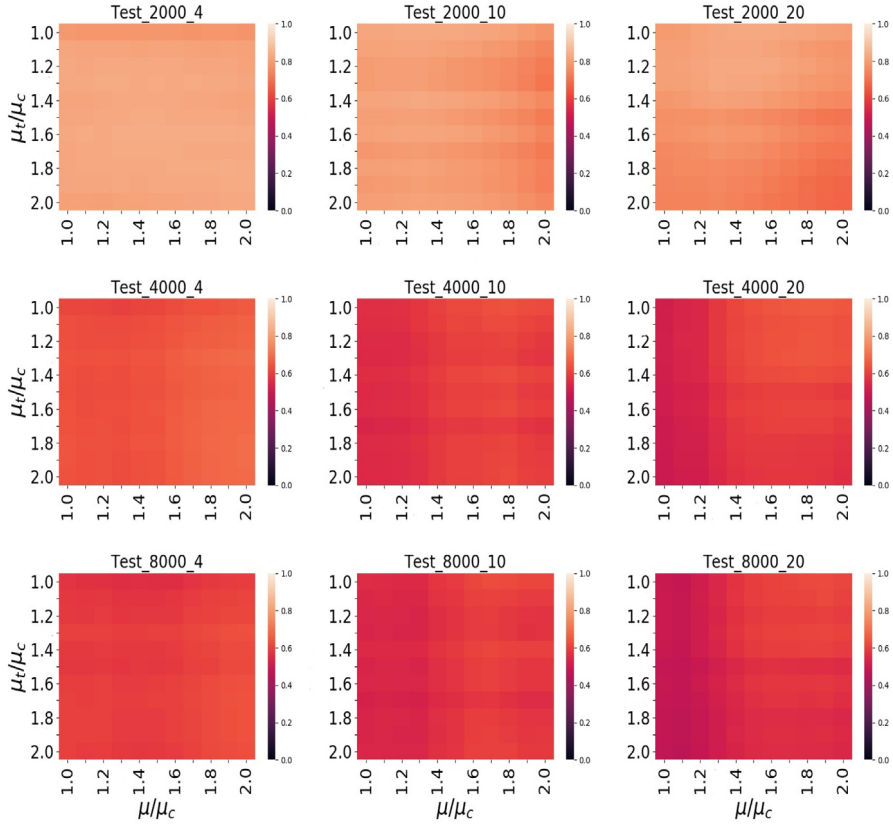

Fig. 3. The heat map of kendall's tau correlation coefficients between the ranking scores and real infected scales with varying infection rate μ/μ_c and training infection rate μ_t/μ_c by *RCNN*.

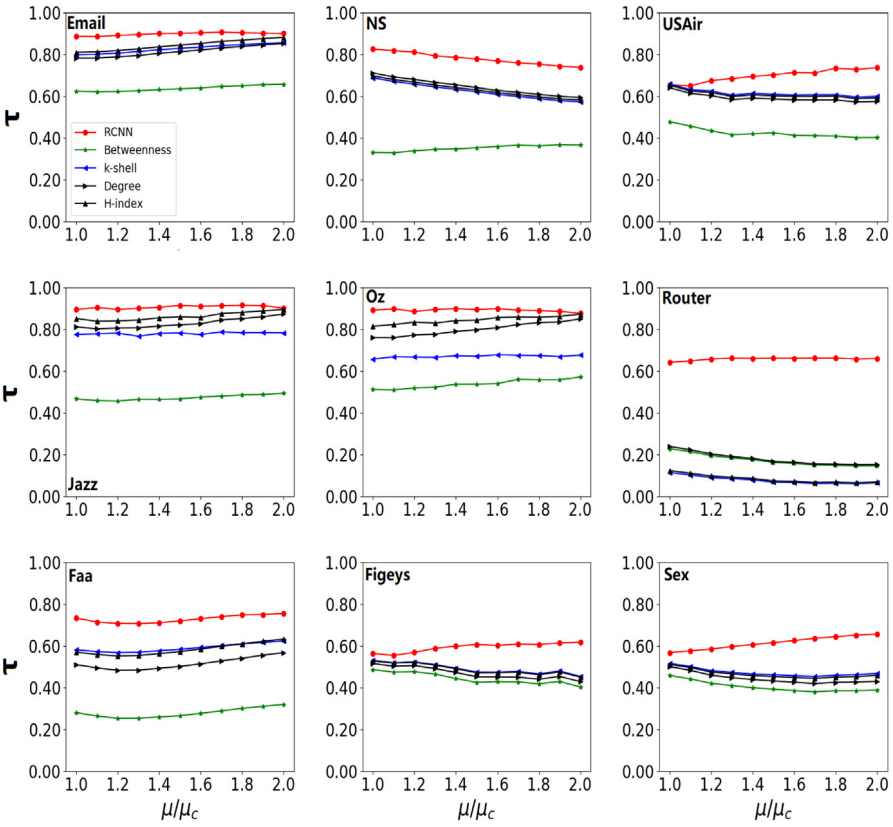

Fig. 4. kendall's tau correlation coefficients between the ranking scores and the relative differences of real infected scales with varying infection rate μ/μ_c where $\mu_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$.

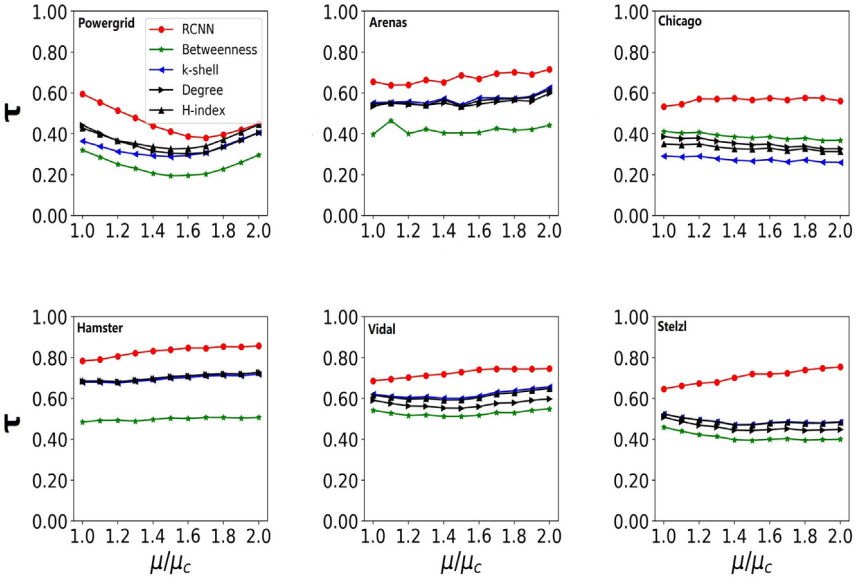

Fig. 5. kendall's tau correlation coefficients between the ranking scores and the relative differences of real infected scales with varying infection rate μ/μ_c where $\mu_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$.

5.5. Compared with benchmark methods

As mentioned in parameters analysis, when we select Train_1000_4 as the training set, *RCNN* performs well in most cases with slight influence of train infection rate μ_t/μ_c and L. So we can fix training infection rate $\mu_t/\mu_c = 1.5$ and $L = 28$ in the following analysis. In order to evaluate the performance of these algorithms under different infection rates μ/μ_c, *RCNN* are compared with other four methods. From Figs. 4 and 5, it can be seen that *RCNN* has the maximum correlation coefficient in most cases under the infection rate varying from 1.0 to 2.0. It means that the ranking results of *RCNN* are the closest to those of real infected scales in all methods. These results demonstrate that *RCNN* can find the nodes which have a greater impact on information spreading under different infection rates.

Table 3
The running time (seconds) of five methods on different real-word networks.

Networks	*RCNN*	Degree	Betweenness	*k*-shell	H-index
NS	0.2881	0.0012	1.0568	0.0047	0.0050
USAir	0.2762	0.0010	16.3315	0.0329	0.0222
Jazz	0.0449	0.0002	0.2708	0.0025	0.0017
Email	0.2160	0.0007	5.7771	0.0077	0.0054
Oz	0.0442	0.0002	0.2659	0.0020	0.0015
Router	0.9775	0.0026	88.1508	0.0195	0.0192
Faa	0.2386	0.0007	5.0775	0.0071	0.0130
Figeys	0.4351	0.0012	21.1437	0.0147	0.0088
Sex	2.4314	0.0092	679.4550	0.0868	0.0512
Powergrid	0.9724	0.0028	84.2357	0.0180	0.0257
Arenas	0.0889	0.0004	0.8130	0.0040	0.0023
Chicago	0.3830	0.0007	2.2881	0.0042	0.0044
Hamster	0.4713	0.0014	25.6983	0.0232	0.0146
Vidal	0.5637	0.0018	30.5360	0.0143	0.0347
Stelzl	0.3248	0.0016	9.5288	0.0077	0.0196

5.6. Complexity analysis

We compare the time complexity of the above five methods. Although *RCNN* need time to generate training sets and training parameters, we can use the parameters for all networks by training model once time. Let the size of feature matrices be $L \times L (L \ll n)$, the time complexity of *RCNN* is [66] $O(I \cdot \sum_{l=1}^{D} M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l)$, where I is the number of iterations. D is the number of convolutional layers. M_l is the side length of the output feature maps of convolutional kernels at the lth convolutional layer. K_l and C_l are the side length of convolutional kernels and the number of output channels at the lth convolutional layer, respectively. $C_0 = 1$ in this paper and the time complexity of *RCNN* is $O(3600n \cdot I \cdot L^2)$. Actually, *RCNN* is an algorithm with linear complexity with the number of nodes.

In addition, the time complexity of degree centrality is $O(m)$. We need take $O(m)$ to calculate the k-shell score of all nodes. So the time complexity of k-shell is also $O(m)$. And the time complexity of betweenness centrality is $O(mn)$ [67]. Table 3 shows the running time of all methods in fifteen real-word networks. From above analysis, *RCNN* achieves the best ranking results in a reasonable time. The running time of *RCNN* is less than one second on all networks except Sex.

6. Conclusion

How to identify critical nodes in complex networks is an interesting and important topic in many applications. Most of previous researches concentrate on finding what kind of structure will make the node more influential. Inspired by the concept of GCNs, by using adjacency matrices of networks and CNNs, *RCNN* is proposed in this paper. According to the experimental results on nine synthetic and fifteen real networks, *RCNN* performs much better than other four benchmark methods in identifying the nodes which have great impact on information spreading. We can use them to detect the potential super-spreaders for accelerating information propagation. The results of parameters analysis show that *RCNN* outperforms other four methods significantly in most cases with fixing the train set, the size of feature matrices and the train infection rate μ_t/μ_c. What is more, *RCNN* have a low computational complexity and can be used in large-scale networks. The method presented in this paper have provided a new idea for identifying critical nodes in complex networks, and the methods can be extended to many other dynamical analyses such as the impact of edges on spreading dynamics and critical nodes in directed networks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

En-Yu Yu: Investigation, Data curation, Methodology, Validation, Visualization, Writing - original draft. **Yue-Ping Wang:** Writing - review & editing. **Yan Fu:** Writing - review & editing. **Duan-Bing Chen:** Supervision, Funding acquisition, Writing - review & editing. **Mei Xie:** Writing - review & editing.

Acknowledgments

This work is jointly supported by the National Natural Science Foundation of China with Grant Nos. 61673085 and 61433014, Science Strength Promotion Programme of UESTC, China (Grant No. Y03111023901014006) and National Key R&D Program of China (Grant No. 2017YFC1601005).

References

[1] G. Zamora-Lpez, C. Zhou, J. Kurths, Cortical hubs form a module for multisensory integration on top of the hierarchy of cortical networks, Front. Neuroinformatics 4 (2010) 1.
[2] X. Zhang, J. Zhu, Q. Wang, H. Zhao, Identifying influential nodes in complex networks with community structure, Knowl.-Based Syst. 42 (2013) 74–84.
[3] B. Hou, Y. Yao, D. Liao, Identifying all-around nodes for spreading dynamics in complex networks, Physica A 391 (2012) 4012–4017.
[4] A. Zeng, C.J. Zhang, Ranking spreaders by decomposing complex networks, Phys. Lett. A 377 (2013) 1031–1035.
[5] J.G. Liu, Z.M. Ren, Q. Guo, Ranking the spreading influence in complex networks, Physica A 392 (2013) 4154–4159.
[6] Y.B. Zhou, M. Li, L. Lü, Quantifying the influence of scientists and their publications: distinguishing between prestige and popularity, New J. Phys. 14 (2012) 033033.
[7] L. Lü, D.B. Chen, T. Zhou, The small world yields the most effective information spreading, New J. Phys. 13 (2011) 123005.
[8] T. Zhang, P. Li, L.X. Yang, X. Yang, Y.Y. Tang, Y. Wu, A discount strategy in word-of-mouth marketing, Commun. Nonlinear Sci. Numer. Simul. 74 (2019) 167–179.
[9] F. Zhou, L. Lü, M.S. Mariani, Fast influencers in complex networks, Commun. Nonlinear Sci. Numer. Simul. 74 (2019) 69–83.
[10] P. Bonacich, Factoring and weighting approaches to status scores and clique identification, J. Math. Sociol. 2 (1972) 113–120.
[11] M. Kitsak, L.K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H.E. Stanley, H.A. Makse, Identification of influential spreaders in complex networks, Nat. Phys. 6 (2010) 888–893.
[12] L. Lü, T. Zhou, Q.M. Zhang, H.E. Stanley, The H-index of a network node and its relation to degree and coreness, Nature Commun. 7 (2016) 10168.
[13] R. Pastor-Satorras, C. Castellano, Topological structure and the h index in complex networks, Phys. Rev. E 95 (2017) 22301.
[14] D. Chen, L. Lü, M.S. Shang, Y.C. Zhang, T. Zhou, Identifying influential nodes in complex networks, Physica A 391 (2012) 1777–1787.
[15] F. Harary, P. Hage, Eccentricity and centrality in networks, Soc. Netw. 17 (1995) 57–63.
[16] L.C. Freeman, A set of measures of centrality based on betweenness, Sociometry 40 (1977) 35.
[17] L.C. Freeman, Centrality in social networks conceptual clarification, Soc. Netw. 1 (1978) 215–239.
[18] J.M. Kleinberg, Authoritative sources in a hyperlinked environment, J. ACM 46 (1999) 604–632.
[19] L. Lü, Y.C. Zhang, C.H. Yeung, T. Zhou, Leaders in social networks, the delicious case, PLoS One 6 (2011) e21202.
[20] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN Syst. 30 (1998) 107–117.
[21] J.X. Zhang, D.B. Chen, Q. Dong, Z.D. Zhao, Identifying a set of influential spreaders in complex networks, Sci. Rep. 6 (2016) 27823.
[22] C. Li, L. Wang, S. Sun, C. Xia, Identification of influential spreaders based on classified neighbors in real-world complex networks, Appl. Math. Comput. 320 (2018) 512–523.
[23] L. Fei, Q. Zhang, Y. Deng, Identifying influential nodes in complex networks based on the inverse-square law, Physica A 512 (2018) 1044–1059.

[24] C. Guo, L. Yang, X. Chen, D. Chen, H. Gao, J. Ma, Influential nodes identification in complex networks via information entropy, Entropy 22 (2020) 242.
[25] Z. Lv, N. Zhao, F. Xiong, N. Chen, A novel measure of identifying influential nodes in complex networks, Physica A 523 (2019) 488–497.
[26] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, 2019, ArXiv Prepr. arXiv:1901.00596.
[27] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, 2018, ArXiv Prepr. arXiv:1812.08434.
[28] M. Kimura, K. Saito, H. Motoda, Blocking links to minimize contamination spread in a social network, ACM Trans. Knowl. Discov. Data 3 (2009) 1–23.
[29] M.E.J. Newman, Spread of epidemic disease on networks, Phys. Rev. E 66 (2002) 16128.
[30] X. Ma, D. Dong, Q. Wang, Community detection in multi-layer networks using joint nonnegative matrix factorization, IEEE Trans. Knowl. Data Eng. (2018) 1.
[31] X. Ma, D. Dong, Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks, IEEE Trans. Knowl. Data Eng. 29 (2017) 1045–1058.
[32] X. Ma, P. Sun, Y. Wang, Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks, Physica A 496 (2018) 121–136.
[33] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Adv. Neural Inf. Process. Syst. (2018) 5165–5175.
[34] X. Ma, W. Tang, P. Wang, X. Guo, L. Gao, Extracting stage-specific and dynamic modules through analyzing multiple networks associated with cancer progression, IEEE/ACM Trans. Comput. Biol. Bioinform. (2016) 1.
[35] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Adv. Neural Inf. Process. Syst, 2002, pp. 585–591.
[36] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, 2012, ArXiv Prepr. arXiv:1211.0053.
[37] M. Niepert, M. Ahmad, K. Kutzkov, Learning convolutional neural networks for graphs, in: 33rd Int. Conf. Mach. Learn. ICML 2016, vol. 4, 2016, pp. 2958–2967.
[38] N. Shervashidze, P. Schweitzer, E.J. Van Leeuwen, K. Mehlhorn, K.M. Borgwardt, Weisfeiler–Lehman graph kernels, J. Mach. Learn. Res. 12 (2011) 2539–2561.
[39] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, ArXiv Prepr. arXiv:1609.02907.
[40] C. Qu, X. Zhan, G. Wang, J. Wu, Z.K. Zhang, Temporal information gathering process for node ranking in time-varying networks, Chaos 29 (2019) 033116.
[41] S. Qi, Z. Bo, Y.H. Wu, L.A. Tang, H. Zhang, G. Jiang, TGNet: Learning to rank nodes in temporal graphs, in: Int. Conf. Inf. Knowl. Manag. Proc., 2018, pp. 97–106.
[42] L.E. Atlas, T. Homma, R.J. Marks II, An artificial neural network for spatio-temporal bipolar patterns: application to phoneme classification, in: Neural Inf. Process. Syst, 1988, pp. 31–40.
[43] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (1989) 541–551.
[44] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.
[45] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015).
[46] T.N. Kipf, M. Welling, Variational graph auto-encoders, 2016, ArXiv Prepr. arXiv:1611.07308.
[47] J. You, R. Ying, X. Ren, W.L. Hamilton, J. Leskovec, Graphrnn: A deep generative model for graphs, 2018, ArXiv Prepr. arXiv:1802.08773.
[48] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, 2017, ArXiv Prepr. arXiv:1709.04875.
[49] W.R. Knight, A computer method for calculating Kendall's tau with ungrouped data, J. Amer. Statist. Assoc. 61 (1966) 436–439.
[50] A.L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.
[51] M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (2006) 36104.
[52] J. Kunegis, KONECT – The Koblenz Network Collection, in: Proc. Int. Conf. World Wide Web Companion, 2013 pp. 1343–1350.
[53] P.M. Gleiser, L. Danon, Community structure in jazz, Adv. Complex Syst. 06 (2003) 565–573.
[54] R. Guimerá, L. Danon, A. Díaz-Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (2003) 65103.
[55] L.C. Freeman, C.M. Webster, D.M. Kirke, Exploring social structure using dynamic three-dimensional color images, Soc. Netw. 20 (1998) 109–118.
[56] N. Spring, R. Mahajan, D. Wetherall, Measuring ISP network topologies with rocketfuel, ACM SIGCOMM Comput. Commun. Rev. (2002) 1–14.
[57] Air traffic control - Network analysis of Air traffic control - KONECT. http://konect.uni-koblenz.de/networks/maayan-faa.
[58] R.M. Ewing, P. Chu, F. Elisma, D. Figeys, Large-scale mapping of human protein-protein interactions by mass spectrometry, Mol. Syst. Biol. 3 (2007).
[59] L.E.C. Rocha, F. Liljeros, P. Holme, Simulated epidemics in an empirical spatiotemporal network of 50,185 sexual contacts, PLoS Comput. Biol. 7 (2011) e1001109.
[60] D. Watts, S. Strogatz, Collective dynamics of small-world networks, Nature 393 (1998) 440–442.
[61] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Phys. Rev. E 72 (2005) 27104.
[62] R.W. Eash, K.S. Chon, Y.J. Lee, D.E. Boyce, Equilibrium traffic assignment on an aggregated highway network for sketch planning, Transp. Res. Rec. 994 (1983) 30–37.
[63] KONECT, Hamsterster Full Network Dataset, KONECT, 2016.
[64] J.F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, et al., Towards a proteome-scale map of the human protein-protein interaction network, Nature 437 (2005) 1173–1178.
[65] U. Stelzl, U. Worm, M. Lalowski, C. Haenig, F.H. Brembeck, H. Goehler, et al., A human protein-protein interaction network: A resource for annotating the proteome, Cell 122 (2005) 957–968.
[66] K. He, J. Sun, Convolutional neural networks at constrained time cost, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 5353–5360.
[67] U. Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 25 (2001) 163–177.