Neurocomputing 468 (2022) 211–221

Contents lists available at ScienceDirect

Neurocomputing

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

Scalable graph neural network-based framework for identifying critical nodes and links in complex networks

Sai Munikoti [a,*], Laya Das [b], Balasubramaniam Natarajan [a]

[a] Department of Electrical and Computer Engineering, Kansas State University, Manhattan, KS 66506, USA
[b] Reliability and Risk Engineering Lab, ETH Zurich, 8092 Zurich, Switzerland

ARTICLE INFO

Article history:
Received 25 April 2021
Revised 16 August 2021
Accepted 8 October 2021
Available online 12 October 2021
Communicated by Zidong Wang

Keywords:
Node prediction
Link prediction
Graph neural network
Robustness
Resilience

ABSTRACT

Identifying critical nodes and links in graphs is a crucial task. These nodes/links typically represent critical elements/communication links that play a key role in a system's performance. However, a majority of the methods available in the literature on the identification of critical nodes/links are based on an iterative approach that explores each node/link of a graph at a time, repeating for all nodes/links in the graph. Such methods suffer from high computational complexity and the resulting analysis is also network-specific. To overcome these challenges, this article proposes a scalable and generic graph neural network (GNN) based framework for identifying critical nodes/links in large complex networks. The proposed framework defines a GNN based model that learns the node/link criticality score on a small representative subset of nodes/links. An appropriately trained model can be employed to predict the scores of unseen nodes/links in large graphs and consequently identify the most critical ones. The scalability of the framework is demonstrated through prediction of nodes/links scores in large scale synthetic and real-world networks. The proposed approach is fairly accurate in approximating the criticality scores and offers a significant computational advantage over conventional approaches.

1. Introduction

Systems that are composed of a network of interconnected sub-systems are becoming increasingly commonplace in engineering and natural sciences. The interconnections in such systems represent interdependence between different sub-systems and introduce interesting spatio-temporal dynamics. Systems ranging from smart cities, smart grids and transportation networks to social networks as well as ecological and biological networks are made up of several sub-systems that interact with each other and give rise to a collective functional behavior. Graph theory offers a powerful framework for studying the behavior of such complex interconnected systems by representing them as an interconnected network consisting of nodes and edges. Graph-theoretic techniques along with machine learning algorithms such as graph neural networks have recently gained popularity for studying both engineered and natural systems [1,2]. For instance, smart cities that are envisioned to connect billions of multi-modal sensors for optimized operation of the system are often modeled as graphs for analyzing interdependence and robustness [3]. Similarly, biological systems such as protein–protein interaction systems are typically represented and analyzed as graphs that allow a systematic study of a large number of collective biological behaviors and co-expressed features [2].

These interconnected systems depend on the proper functioning of their constituent sub-systems (nodes) and their interconnections (links) for reliable operation of the overall system, for instance achieving a functional objective or providing a certain output/product. However, typically there exists a set of *critical nodes/links* that play a more crucial role in determining the output of the system than other (non-critical) nodes/links. These nodes/links represent a set of sub-systems and/or their interconnections, whose removal from the graph maximally disconnects the network, and thus severely disrupts the operation of the system. As a result, identification of critical nodes/links in complex networks is an important task for analysis and/or design of the underlying systems, and bears significance in several applications including social networks analysis, feature expression in biological networks, quality assurance and risk management in telecommunication networks, assurance of robustness in urban networks, control of social contagion, among others. For instance, identifying critical nodes/links in protein–protein interaction networks can be particularly useful in drug design. In such scenarios, critical nodes represent a minimum cardinality set of proteins whose removal would destroy the primary interactions and thus help neutralize poten-

* Corresponding author.
E-mail address: saimunikoti@ksu.edu (S. Munikoti).

https://doi.org/10.1016/j.neucom.2021.10.031
0925-2312/

tially harmful organisms (e.g., bacteria or viruses) [4]. Similarly, in covert terrorist networks, critical nodes are individuals whose deletion from the graph will result in a maximum breakdown of communication between individuals in the network [5] and can significantly disrupt the operation of the network. In the case of engineered networks such as the Internet, critical nodes/links are those which if compromised, will allow a hacker to increase the impact of a virus on the network, and which if strongly protected/shut down can efficiently stop the spread of a virus. Similar applications can be found in social networks, transportation engineering and emergency evacuation planning.

The problem of identifying critical nodes/links in a graph is becoming increasingly important across several domains. This is primarily because of the increase in the frequency of natural disasters as well as adversarial attacks in the recent past, causing large disruptions in engineered systems. For instance, the rise in the usage of digital infrastructure around the globe because of the ongoing COVID 19 pandemic has been accompanied by a surge in cyber attacks on different networks [6,7], making it crucial to identify and reinforce the cyber-security of key data stores/nodes. In biological systems, the use of high-throughput technologies to detect protein interactions has led to an exponential growth in the size of the protein interaction graphs in various species [8]. In such applications, the identification of critical nodes/links can assist in easing the study of interactions in biological networks.

As described earlier, critical nodes are those nodes whose removal maximally decreases the network functionality in terms of connectivity. This network functionality is often studied in the literature with the help of robustness metrics of the graph. Robustness quantifies the impact of loss of resources (nodes/links) on the performance of a system, and directly relates to the criticality scores of the nodes/links. There are numerous metrics and methodologies available in the literature that quantify the robustness of graphs such as effective graph resistance, flow robustness, total graph diversity, etc. [9]. Owing to the inherent topological structure of graphs, each node/link contributes differently to the graph robustness, and hence its removal/loss affects the robustness to a different degree. In this regard, the notion of node/link criticality score is introduced, which quantifies the decrease in robustness (such as effective graph resistance) when the corresponding node/link is removed from the graph. Criticality scores are then employed to rank the nodes/links, with the topmost rank assigned to the node/link whose removal maximally decreases the graph robustness and vice versa.

1.1. Related Work

Several methods have been proposed to compute criticality scores based on graph robustness [10–12]. However, such approaches typically measure the score of a node/link and repeat the process for all nodes/links in a graph. As a result of this iterative approach, such techniques exhibit computational complexity that increases drastically with the size of the graph, i.e., the number of nodes/links in the graph. For example, the complexity for identifying the optimal link whose removal maximally reduces robustness is of order $O\left(N^5\right)$ for a graph with N nodes [11]. As a result, various efforts are directed towards approximating such algorithms and developing a computationally efficient solution [13,14,11]. These approaches offer a trade-off between scalable computation and accurate identification of links. However, even with the proposed trade-offs, the lowest achieved complexity for the case of link identification is of order $O\left(N^2\right)$. Similarly, the authors in [12] deploy effective graph resistance as a metric to relate the topology of a power grid to its robustness against cascading failures. Specifically, the authors propose various strategies to identify node pairs where the addition of links will optimizes graph robustness. The minimum achieved complexity is of the order $O\left(N^2 - N + 2L_c\right)$, where $L_C = \frac{N!}{2!(N-2)!-L}$ with L links and N nodes. However, with an increase in graph size, the accuracy of such approximations decreases, accompanied by a considerable increase in execution time. To overcome computational complexity, the authors in [15], propose a method based on a genetic algorithm to enhance network robustness. Particularly, the authors focus on identifying links whose removal would severely decrease the effective graph resistance of the graph. However, the algorithm requires fine-tuning of various parameters and is not scalable. Moreover, several applications involve dynamically changing network topologies requiring node/link robustness to be dynamically estimated and maintained. The existing approaches being analytical in nature, and not exhibiting an inductive nature, one has to repeat the same procedure whenever the graph structure changes.

In contrast to optimization-based, and related approximation-based approaches, machine learning frameworks that exploit their ability of extract patterns from data and topology of the system have also been applied for critical node identification. For instance, En-Yu et al. in [16] have proposed the use of convolutional neural networks applied on a feature matrix derived from the adjacency matrix of a graph to identify critical nodes. The neural network model learns the topological structure of the graph and identifies the critical nodes in the system. Sun et al. in [17] formulate a binary classification problem to identify the influential nodes in a graph by making use of the adjacency matrix and the eigenvector matrix of an influence graph derived from the original topology of the system. However, despite the use of machine learning algorithms for automatically learning the topological information from the features, the above approaches lack significantly in their suitability to real-life problems. Specifically, the use of a convolutional neural network with the adjacency matrix as an input restricts the applicability of the trained model to networks of the same size as the trained graphs. Moreover, such an approach always depends on the information of the entire graph to calculate the criticality of a node anywhere in the graph. Furthermore, in these approaches, the fundamental definition of critical nodes involves the use of either a susceptible-infected-recovered (SIR) model, or an influence graph derived from the original graph of the system. These definitions inherently capture the ability of a node to infect/influence a large portion of the graph, a property referred to in the literature as influence of the node. On the other hand, the criticality of a node refers to the importance of the node in proper functioning of the system, and is distinct from its influence. In addition, existing methods are limited to identification of critical (or rather influential) nodes in the network, and do not address the problem of critical link identification. Finally, these methods are not inductive in nature, and thus require fresh training for graphs of different sizes and types. This article addresses the above gaps in the literature, and proposes an inductive learning approach that offers computational advantage over optimization-based approaches, while being more flexible and scalable compared to existing machine learning-based approaches. The proposed framework leverages the sub-graph around node/link neighborhood to predict criticality scores.

1.2. Contributions

This article proposes an inductive learning approach for identifying critical nodes/links in a graph that affords both computational advantage and scalability to graphs of different size. Specifically, the article makes the following contributions to the literature on identification of critical nodes/links in a graph:

1. The problem of identifying critical nodes/links in a graph is formulated as an inductive machine learning problem for the first time. This formulation allows exploiting the local neighbourhood information of a node/link and does not require knowledge of the graph's complete topology.
2. The proposed framework is used to identify the critical nodes/links in real-life interconnected systems in social networks, biological networks and power grid network. Results show that the mean accuracy of identifying the top 5% of critical nodes/links in such systems is more than 90%.
3. The inductive nature of the formulation allows one to train the model on a portion of the graph or a synthetic graph of similar characteristics and apply the trained model to the rest of the graph. This is demonstrated with numerical examples in the article.
4. The suitability of the framework to real-life applications by fine-tuning pre-trained models with transfer learning has been demonstrated. This is particularly advantageous in applications where the graph under consideration does not have enough nodes/links to train the neural network model from scratch.

The rest of the paper is organized as follows: Section 2 describes various graph robustness metrics considered in this article followed by a brief introduction of graph neural network. Section 3 presents the proposed framework. The experimental setup and results are presented in Section 4. Finally, conclusions are provided in Section 5.

2. Background

This article proposes an inductive learning based approach for the identification of critical nodes/links in large complex networks. This is a two-step process, wherein the impact of a node/link on the robustness of a graph is estimated by removal of the node/link, followed by a relative ranking of the nodes/links based on their impact. This involves the use of an appropriate metric of robustness and a Graph neural network (GNN) to capture the impact of nodes/links on the metric, and to perform the relative ranking. These components of the approach are discussed next.

2.1. Metrics of Graph Robustness

The robustness of a system is its ability to function properly in the presence of disturbances/perturbations, such as failures of components. In a graph theoretic setting, this translates to the ability of the graph to function with loss of nodes/links. Several metrics have been proposed in the literature [9] that quantify the robustness of a graph against random and targeted loss of nodes/links. The authors in [9,18], extensively study various graph robustness metrics for different types of graphs and provide the most appropriate metrics for each type of graph. It has also been pointed out in the literature [9] that there is no generic metric of robustness that can work for all types of graphs (i.e., with different degree distributions, average clustering coefficient, assortativity, etc.) under all scenarios (i.e, node/link removal, targeted and random removal, etc.). As a result, there exist several metrics depending on the objective and properties of the graph.

Effective graph resistance (R_g) is a widely used metric to quantify graph robustness [11,12,19] and is equal to the sum of the effective resistances over all pairs of nodes [20]. The effective resistance between any two nodes of a graph is computed by performing series and parallel operations on an electrical circuit-equivalent of the graph. R_g considers both the number of paths between nodes and their length (link weight), intuitively measuring the presence and quality of backup possibilities in the graph. The spectral form of R_g can be expressed as:

$$R_g = \frac{2}{N-1}\sum_{i=1}^{N-c}\frac{1}{\lambda_i}, \tag{1}$$

where, $\lambda_i, i = 1, 2, 3, \ldots .N$ are the eigen values of the Laplacian matrix of a graph G with N nodes, and c is the number of connected components in the graph. Eigen values of the graph Laplacian matrix is computed first, and then the sum of reciprocal of non zero eigen values is computed to determine the final R_g score. The expression in Eq. (1) is a normalized expression, in that it allows the comparison of the metric, R_g across graphs of different dimensions.

Weighted spectrum (W_s) is also a widely used metric for graph robustness [21,22] and is defined as the normalized sum of n-cycles in a graph [21]. An n-cycle in a graph G is defined to be a sequence of nodes $u_1, u_2, \ldots u_n$ where u_i is adjacent to u_{i+1} for $i \in [1, n-1]$ and u_n is adjacent to u_1. This metric is initially introduced to analyze the resiliency of internet topology. Thereafter, it has been compared to various other resiliency metrics and found to be a versatile metric especially for geographically correlated attacks [18]. The weighted spectrum can be expressed as,

$$W_s = \sum_i (1-\lambda_i)^n, \tag{2}$$

where, different values of n correspond to different graph properties. For instance, $n = 3$ denotes the number of triangles in a graph with W_s related to the weighted clustering coefficient. Similarly, with $n = 4$, W_s is proportional to the number of disjoint paths in a graph. Since this work focuses on quantifying the robustness of the graph based on connectivity, $n = 4$ is used in this study. Thus, we first compute the eigen values of the normalized graph Laplacian matrix. Then, we sum the difference of eigenvalues from 1 raised to the power 4 to get the final W_s metric. It must be noted here that although this work uses R_g and W_g as metrics of robustness, the proposed approach is generic enough to be applied for any robustness metric.

2.2. Graph neural networks

Graph neural networks are a variant of artificial neural networks that are designed to capture patterns in data that can be represented in a graphical structure. The authors in [23] initially introduced the transformation of the convolutional operations from Euclidean domain to problems involving graphs. The working principle of such models resembles that of convolutional neural networks and can work directly on graphs and exploit their topological information. The standard learning tasks on graph data are node classification, link prediction, graph classification, etc. GNN addresses various learning tasks across domains including computer vision [24,25], natural language processing [26,27], bio-chemistry [2], etc. A GNN typically involves learning features called node embedding vectors followed by feedforward layers for regression or classification tasks. However, the algorithm proposed in [23] depends on the size of the graph leading to scalability issues. To address this scalability issue, the authors in [28] propose an inductive learning framework called GraphSAGE, where node embeddings are learned using subgraphs and thus, is independent of graph size. Furthermore, this framework can be leveraged to infer about the unseen/new nodes of graphs belonging to the same family. The standard procedure to learn this embedding vector involves a "message passing mechanism", where the information (node feature) is aggregated from the neighbors of a node and then combined with its own feature to generate a new feature vector. This process is repeated to generate the final embedding for each node of a graph. The GraphSAGE algorithm learns the mapping (aggregator) function instead of learning the embedding vectors. Hence, it can induce the embedding of a new node during training,

given its features and neighborhood. The proposed framework makes use of GraphSAGE as it is a state-of-the-art GNN modeling framework and is applicable to graphs of different sizes.

3. Proposed framework

This paper focuses on identifying critical nodes/links in large graphs. Critical node/link identification is a combinatorial optimization problem introduced by [4]. In the following, we present the problem formulation and algorithmic details of the proposed framework.

3.1. Problem formulation

Consider a graph $G = (V, E)$, represented as a tuple of nodes V and links E. The problem of critical node identification is to identify a set of nodes $V_c \subseteq V$, whose deletion from the graph results in a maximal decrease in network functionality measured in terms of a graph robustness metric. Similarly, the problem of link identification is to identify a set of links $E_c \subseteq E$, whose deletion from the graph results in a maximal decrease in the graph robustness metric. It is important to note that in evaluating the decrease in robustness, we remove a single node/link at a time. Assigning criticality based on a sequence of node/link removal is beyond the scope of this article and is kept as a future work.

The above problem is addressed by incorporating local neighborhood information for nodes/links. This approach assumes that there exists a functional relation/mapping that links the local topology and features with the criticality of a node. The approach then makes use of a graph neural network framework, specifically, GraphSAGE to learn this mapping from data via training. Consider a node u, whose criticality score we wish to learn. The first step involves generating the feature vector of the node u (h_u) as well as of the neighboring nodes i.e., $h_v \ \forall v \in N(u)$ where $N(u)$ represents the neighborhood of node u. According to the assumption of the approach, the underlying mapping between these features and the criticality score r_u can be expressed as:

$$r_u = f(h_u, h_v) v \in N(u) \tag{3}$$

The graph neural network framework makes use of the network topology and the feature vectors to learn the underlying mapping for different graphs, and generates an approximation of this mapping $\hat{f}(h_u, h_v, N(u))$. In this regard, this article proposes a two-step approach for inductive learning-based approximation of the criticality scores of nodes/links in a graph, referred to as Inductive Learner for Graph Robustness (ILGR). The framework is first introduced for the identification of critical nodes, and is then extended to link identification problem. The first step involves the use of computationally manageable graphs to learn appropriate node embeddings and subsequently criticality scores with a GNN that allows faster learning. This is followed by the prediction of criticality scores for new nodes or unseen nodes (nodes not used for training) of the graph. It must be noted, however, that in order for the neural network to reliably predict the scores at the time of deployment, the training and testing data (graph) should exhibit some similar properties. In this regard, one can use standard (synthetic) graphs such as power-law graphs and power-law cluster graphs that are known to exhibit properties similar to most real-world networks. However, in certain applications where such synthetic networks with similar properties cannot be obtained, it is possible to use a subset of the nodes of the testing graph and adopt transfer learning to tune the parameters of an already trained neural network model.

The proposed framework is illustrated in Fig. 1. The GNN model first learns *node embeddings* from node features and neighborhood sub-graph, which are then used to calculate the criticality scores. The following sections discuss in detail, the steps involved in obtaining the node criticality scores from a set of features and neighborhood.

3.2. Node embedding module

The first module of ILGR learns the embedding vector for each node by utilizing the graph structure and node target (criticality) scores. This is achieved in a manner such that nodes that are close in the graph space also lie close in the embedding space. As an initialization, the embedding of each node is composed of only the degree of the node, followed by a predefined number of ones. The node embeddings are learned based on GraphSAGE which is described briefly in the previous section. However, there are some modifications that have been made in the implementation. The node embedding module is further subdivided into two tasks. The first task learns a representation for every node based on a combination of the representation of its neighboring nodes, parametrized by a quantity K, which quantifies the size of the neighborhood of nodes. Specifically, the parameter K controls the number of hops to be considered in the neighborhood. For instance, if $K = 2$, then all the nodes which are 2 hops away from the selected node will be considered as neighbors. This defines the neighborhood of a node v as:

$$N(v) = \{u : \mathrm{D}(u, v) \leqslant K, \quad \forall u \in G\} \tag{4}$$

where $\mathrm{D}(u, v)$ is a function that returns the smallest distance between nodes u and v. After defining the neighborhood, an aggregator function is employed to associate weights to each neighbor's embedding and create a neighborhood embedding for a selected node. Unlike previous works [28,29], where weights are predefined, this work uses the attention mechanism to automatically learn the weights corresponding to each neighbor node as [30]:

$$h^l_{N(v)} = \mathrm{Attention}\left(Q^l h^{l-1}_k\right) \quad \forall k \epsilon N(v) \tag{5}$$

where, $h^l_{N(v)}$ represents the embedding of neighbourhood of a node v in layer l of the GNN, h^{l-1}_k represents the embedding of k^{th} neighboring node of v in the $l-1$ layer of the GNN. Q^l is the attention weight at layer l of the GNN. Thereafter, for each neighborhood depth until $k = K$, a neighborhood embedding is generated with the aggregator function for each node and concatenated with the existing embedding for node v. However, the existing node embedding is not solely the output from the embedding of previous layer as implemented in various previous works. Rather, for existing node embedding, this article proposes to use the output of node embeddings from the previous two layers. This is similar to skip connections used by various researchers in the past for enhancing model performance for images and speech-related applications [31,32], and can be expressed as:

$$h^l_v = \mathrm{Relu}\left(W^l \left[h^{l-1}_v || h^{l-2}_v || h^l_{N(v)}\right]\right) \forall k \epsilon N(v) \tag{6}$$

where, h^l_v represents the embedding of node v in layer l of the GNN, h^{l-1}_v and h^{l-2}_v denote the embedding of node v in layers $l-1$ and $l-2$ respectively and $h^l_{N(v)}$ is the embedding aggregated from neighbors of v as given in Eq. (5). The operations performed by Eqs. (5) and (6) are known as aggregation and combination, respectively. This aggregation and combination process is repeated for all layers of the model to obtain final node embeddings. The steps involved in the embedding module is summarized in Algorithm 1. The symbol || in step 5 of the Algorithm 1 denotes the concatenation operation.

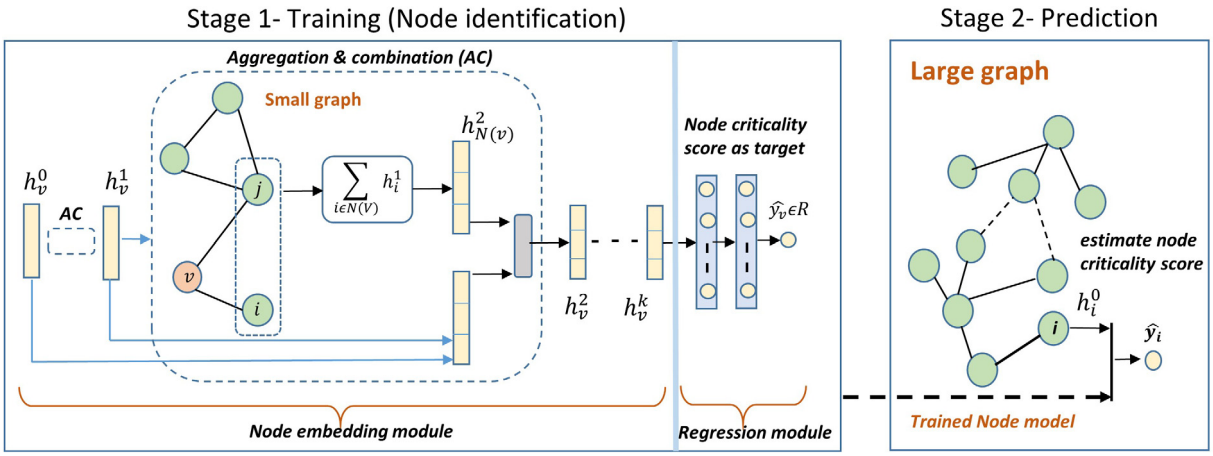

Fig. 1. Proposed ILGR framework for node identification.

Algorithm 1: ILGR embedding module

Input: Graph G, input node features $X_v \forall v \epsilon V$, unknown model weights W(combination weights) and Q(aggregation weights).

Output: Nodes embedding vector $z_v \ \forall v \epsilon V$.

Initialize: $h_v^0 = X_v \ \forall v \epsilon V$
for layer $l = 1$ to $l = L$ **do**
  for node $v = 1$ to $v = V$ **do**
    $h_{N(v)}^l$ = Attention$(Q^l h_k^{l-1}) \ \forall k \epsilon N(v)$;
    $h_v^l = \text{Relu}(W^l \left[h_v^{l-1} || h_v^{l-2} || h_{N(v)}^l\right]) \ \forall v \epsilon V$;
  end for
end for
return Final embedding vector $z_v = h_v^L \ \forall v \epsilon V$;

3.3. Regression module

The output of the embedding module is passed through a regression module which is composed of multiple feedforward layers. The feedforward layers transform the embedding non-linearly and finally generate a scalar that denotes the node criticality score. The output of the m^{th} layer in the regression module can be expressed as:

$$y_v^m = f(W^m y_v^{m-1} + b^m) \tag{7}$$

where W^m and b^m represent the weights and biases in the m^{th} layer, f is the activation function such as ReLU, Softmax, etc. y_v^m is the output of m^{th} layer corresponding to node v and $y_v^0 = z_v$ (output of embedding module). The complete framework for node identification is depicted via Fig. 1.

The framework for link criticality scores is almost similar to that of the node analysis with small differences. The output of the node embedding module is connected to a link embedding layer which generates link embedding from the associated pair of node embeddings. There are various ways to combine node embeddings including, concatenation, inner product, mean, L2 norm, etc. In this work, the Hadamard product is used to generate link embedding. Thereafter, the link embedding is passed through the regression module to predict link criticality score. The entire framework of ILGR for link prediction is shown in Fig. 2. It is worth noting that the benefit of posing the node/link identification task as a regression problem is that once the criticality scores are predicted for nodes/links, then the user can select top-N % of the nodes/links based on the requirement. Hence, the framework is more flexible and does not depend on a specific threshold for determining criticality.

The algorithm learns the weights of an aggregator function and feedforward layers by minimizing an appropriate loss function. In the proposed framework, the output of the model is node/link criticality score, which is further employed to rank nodes/links and identify the critical ones. So, instead of exactly learning the criticality values, it is sufficient to learn any real values provided the relative order of nodes remains intact. A suitable loss function for this scenario is the ranking loss [33]. Unlike other loss functions, such as Cross-Entropy Loss or Mean Squared Error Loss, whose objective is to learn to predict a value or a set of values given an input, the objective of ranking loss is to preserve the relative distances between the inputs. Here, the pair-wise ranking loss has been used that looks at a pair of node ranks at a time. The goal of training the

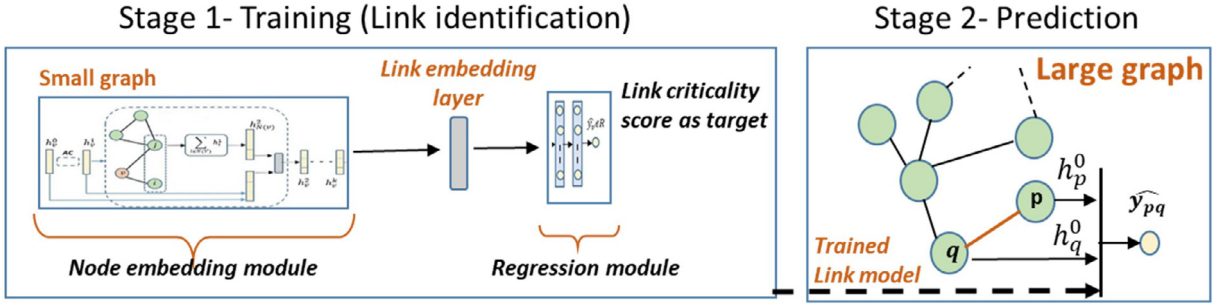

Fig. 2. Proposed ILGR framework for link identification.

model is to minimize the number of inversions in the ranking, i.e., cases where a pair of node ranks is in the wrong order relative to the ground truth. This loss function can be expressed as:

$$L_{ij} = -f(r_{ij}) \log \sigma(\hat{y}_{ij}) - (1 - f(r_{ij})) \log(1 - \sigma(\hat{y}_{ij})) \quad (8)$$

where, r_i is the ground truth value of criticality score for node i. $r_{ij} = r_i - r_j$ is the actual rank order, which the model is learning to infer through $\hat{y}_{ij} = \hat{y}_i - \hat{y}_j$ by minimizing the loss L_{ij}. f is a sigmoid function. The loss is aggregated for all the training node/link pairs, and then optimized to update the model weights. The net training loss can be written as,

$$Loss = \sum_{(i,j) \subset E} L_{ij}$$

where (i,j) denotes an edge pair belonging to graph link set E. Once weights are learned, then an embedding vector and consequently the node/link scores can be predicted for a test node/link given its features and neighboring information.

3.4. Model settings and Training

There are various hyper-parameters in the model that need to be tuned for training the model. The number of node embedding layers, i.e., the depth of the GNN is selected as three. The number of neurons in these layers are $64, 32$, and 16 respectively. The regression module consists of three feedforward layers with $12, 8$ and 1 neurons respectively. The activation function in all the layers is kept as *relu*. The aggregation and combination operation involve trainable network weights. The ranking loss function is optimized via ADAM optimizer in the TensorFlow framework with its default settings. The training of both the embedding and regression modules is conducted end to end with input being a specific node/link along with its neighbor information and output being the corresponding criticality score.

A different model is trained for each family of synthetic graphs. This is because, different families of graphs vary in their overall structure and link connections, i.e., degree distributions, assortativity, average clustering coefficient, etc. For a particular graph family, different random instances of graph are sampled, and then the ground truth of node/link criticality scores are computed for each sampled graph using a conventional approach as described in Algorithm 3. Fundamentally, it involves an iterative method of removing a node/link from the graph and computing the robustness metric of the residual graph. The term "residual graph" refers to the leftover graph after the removal of node/link. This process is repeated for all the nodes/links of the graph. Thereafter, the nodes/links are ranked based on the computed criticality scores. The ground truth criticality scores are then used to train our Graph neural network model on multiple graphs of the same family. The training steps are outlined in the Algorithm 2. At each iteration of the algorithm, we first compute the embedding vector for nodes via the embedding module. Embedding module primarily takes node features and neighborhood information as input and outputs embedding vectors by passing information across different layers of the graph neural network. Next, the regression module uses these embedding vectors to estimate node/link criticality scores. The loss between these estimated and actual ground truth criticality scores is optimized to update the model parameters. This process is repeated multiple times to obtain the final trained model. Thereafter, the trained model is validated to predict criticality score on graphs of higher dimensions. For example, a model can be trained on multiple power-law graphs of dimension $100 - 1000$, and can be evaluated on power-law graphs of dimension $100 - 100000$. The model is trained end to end on the Tensorflow framework with Stellargraph library. For real-world graphs such as US Power grid [34,35], Wiki-vote, already trained models are employed for predicting node ranks. Additionally, transfer learning is also implemented for a real-world network to demonstrate the efficacy of the framework. The experimental setup and results are shown in the next section.

Algorithm 2: Algorithm of ILGR

Input: Model with unknown weights.
Output: Trained model.
1: Generate ground truth criticality scores of nodes/links based on graph robustness score
2: **for** each iteration **do**
3: Get each node embedding from embedding module.
4: Estimate criticality score of nodes/links through regression module.
5: Update weights of both modules by solving Eqn. (5)
6: **end for**
7: Predict node/link score on test graph.
8: **return** Top $N\%$ of most critical nodes/links.

Algorithm 3: Conventional approach of identifying critical nodes/links on the basis of graph robustness

Input: Graph G with V nodes.
Output: Node/link critical scores
1: **for** n in V **do**
2: Remove node/link n from graph G
3: Compute robustness metric of the residual graph $(G - n)$
4: Assign criticality scores to node/link n
5: **end for**
6: Rank nodes/links on the basis of above computed criticality scores. Top ranks correspond to more critical nodes/links.
7: **return** Top $N\%$ of most critical nodes/links.

4. Experimental results

This section discusses results obtained with the proposed framework for node and link criticality score prediction over a wide range of applications. The datasets used in this work to demonstrate the applicability of the proposed framework are first discussed, followed by the evaluation metrics used to report the performance of the framework. A baseline approach is used to compare the performance of the proposed framework with existing approaches. [1]

4.1. Datasets

We examine the performance of ILGR on both synthetic and real world graphs. The two commonly used synthetic graphs are generated using Python NetworkX library are as follows:

1. **Power law**: Graphs whose degree distribution follow power law (i.e., heavy tailed), and many real networks have shown to be of this family [36]. It can be generated through a process of preferential attachment in which the probability that a new node N_y connects with an existing node N_x is proportional to the fraction of links connected to N_x.

[1] Codes related to this work can be found at https://github.com/saimunikoti/GraphNeuralNetwork-Resilience-ComplexNetworks

2. **Power law cluster**: Graphs which exhibit both power law degree distribution and clusters, and many real-world networks manifest these properties [37]. As shown in [38], one can construct a PLC graph by following a process of preferential attachment but in some fraction of cases (p), a new node N_y connects to a random selection of the neighbors of the node to which N_y last connected.

The real-world networks that are analyzed in this work are as follows [39]:

1. **Bio-yeast**: It is a protein–protein interaction network for yeast consisting of 1458 nodes and 1948 links. [37].
2. **US-powergrid**: It represents the western US power grid with 4941 nodes representing buses and 6594 links as transmission lines [35].
3. **Wiki-vote**: It contains all the Wikipedia voting data from the inception of Wikipedia till January 2008. 7115 Nodes in the network represent wikipedia users with 103689 links, where each link from node i to node j indicates that user i voted a user j [34].
4. **cit-DBLP**: It is the citation network of DBLP, a database of scientific publications. There are 12591 nodes and 49743 links. Each node in the network is a publication, and each edge represents a citation of a publication by another publication [35].

4.2. Evaluation metrics

The trained model predicts the criticality scores of the test nodes/links in a graph, which are then used to identify the most critical nodes/links. In a general setting of robustness analysis, it is more relevant to identify top ranked nodes/links that are most critical, rather than knowing the ranks of all the nodes/links. Therefore, we have used Top-N% accuracy to evaluate the proposed framework against the conventional approach. It is defined as the percentage of overlap between the Top-N% nodes/links as predicted by the proposed method and the Top-N% nodes/links as identified by conventional baseline approach, i.e., Algorithm 3 (discussed in next section). Top-N% accuracy can be expressed as,

$$\frac{|\{\text{Predicted Top-N\% nodes/links}\} \cap \{\text{True Top-N\% nodes/links}\}|}{|V| \times (N/100)}, \tag{9}$$

where, $|V|$ is the number of nodes/links and N is the desired band. In this work, the results are reported for Top-5% accuracy. Further, the computational efficiency of the proposed approach is demonstrated in terms of execution time. More specifically, the execution time is same as wall-clock running time, i.e., the actual time the computer takes to process a program.

4.3. Baseline approaches

To evaluate the performance of our proposed approach, we compare ILGR with a conventional method of estimating node/link criticality. The classical methodology involves an iterative method of removing a node/link from the graph and computing the robustness metric of the residual graph. The term "residual graph" referred to a leftover graph after the removal of node/link. This process repeats for all the nodes/links of the graph. Thereafter, the computed criticality scores of all the nodes/links are arranged to generate ranks and identify the most critical ones whose removal maximally decreases the graph robustness. Algorithm 3 summarizes the typical conventional approach to identify critical nodes based on graph robustness metrics. Furthermore, we have also implemented a Graph convolutional algorithm (GCN) which is used in many recent works related to critical node/link identification problem [23,16,17]. Mathematically, GCN can be written as:

$$h^{l+1} = \sigma\left(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{0.5}h^{l}W^{l}\right) \tag{10}$$

where, h^l denotes the l^{th} layer of the neural network, σ is the non-linearity function (e.g., ReLU), and W is the weight matrix for this layer. $\widetilde{D}$ and $\tilde{A}$ are the degree and adjacency matrices for the graph with the superscript referring to the additional link between each node and itself. The model settings (number of layers, node samples, activation functions, etc.) of GCN is kept identical to our model for fair comparison. From Eq. (10), it can be inferred that GCN is dependent on the input dimension (i.e., adjacency matrix of input graph $\widetilde{A}$) for prediction. Thus, a model can only be trained on graphs of identical node dimension and consequently can solely be used for node/link identification in graphs of similar dimension. This reduces the scalability and generalizability of the framework which is collectively addressed in our approach.

4.4. Results and discussion for node identification

This subsection access the performance of the proposed ILGR framework for nodes, in terms of Top-5% accuracy and algorithm execution time. Different types of a graph with varying dimensions are considered for assessment. Here, we study two different types of synthetic graphs, i.e., power-law and power-law cluster. Further, with respect to the robustness index, we analyze two different metrics namely effective graph resistance and weighted spectrum, which are described in the earlier section. Therefore, for each type of graph, the proposed method is evaluated for two different metrics leading to four different scenarios. Table 1 tabulates the Top-5% accuracy of each such scenario. The models are trained on 30 graphs of dimension varying from 100 to 1000 nodes. The trained model is then employed to predict node criticality scores in power-law and power-law cluster graphs of higher dimensions, i.e., 500, 1000, 5000, and 10000 nodes.

It can be inferred from the Table 1 that in the process of identifying Top-5% of the most critical nodes, the mean accuracy of the PL model for robustness metrics R_g and W_s is 94.8% and 95.6%, respectively. The mean is taken across graphs of different node counts. Similarly, the accuracy of PLC model for R_g and W_s is 93.3% and 94.5%, respectively. The scalability of the framework is depicted via the model's high performance in graphs of increasing node dimensions. PL models perform relatively better than PLC models because the inherent topology is comparatively simpler in PL than that of PLC. More specifically, the learning mechanism of the ILGR is based on nodes sub-graphs, which generates embedding vectors by combining aggregation and combination operations. In PL graphs, there are very few nodes with a high degree while most of them bear small degrees. Therefore, node sub-graphs are more or less similar for most of the nodes. On the other hand, PLC graphs have a high clustering coefficient which introduces variability and complexity in sub-graphs, thereby makes the learning more challenging. This can be seen from the sample graphs in Fig. 3. Nevertheless, our framework attains sufficiently high accuracy for both the models, which demonstrates the accuracy of the proposed framework in the task of estimating node criticality scores. Table 1 also shows the comparison of the proposed ILGR with the GCN algorithm trained for networks with 1000 nodes. It can be observed that the proposed approach outperforms GCN model for both PL and PLC networks. Furthermore, it must be noted that while it is required to train different GCN models for different sizes of networks, the same ILGR model can be used to predict the criticality for networks of any dimension.

Table 1
Accuracy of node identification task in synthetic graphs. PL: Power-law; PLC-Power-law cluster; Ntest: # test nodes.

Test graph size	Ntest 500		Ntest 1000		Ntest 5000		Ntest 10000	
graph/scores	Rg	Ws	Rg	Ws	Rg	Ws	Rg	Ws
PL (100–1000): **ILGR**	0.972	0.965	0.960	0.960	0.939	0.958	0.924	0.942
PLC (100–1000): **ILGR**	0.943	0.952	0.937	0.950	0.935	0.948	0.919	0.930
PL (1000): GCN	-	-	0.942	0.938	-	-	-	-
PLC (1000): GCN	-	-	0.927	0.935	-	-	-	-

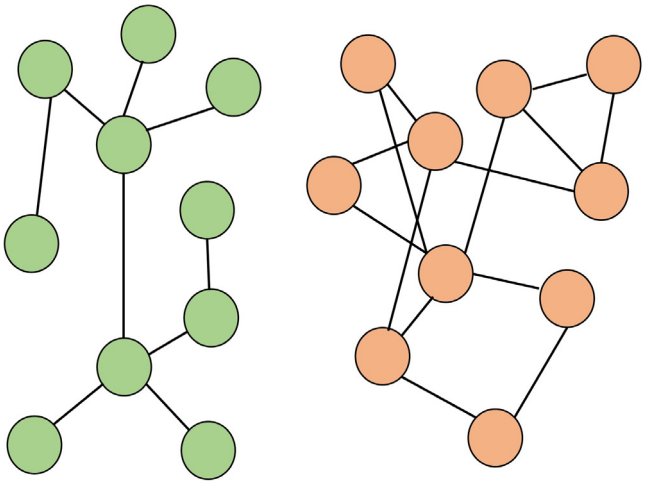

Fig. 3. Left: Power law; Right: Power law cluster graph.

The generalizability and the scalability of the proposed approach are further reinforced through Table 2, which reports the Top-5% accuracy in real-world networks, predicted through models trained on PL and PLC graphs. It can be observed that the PL model has a mean accuracy of 87.4% and 89.5 % for R_g and W_s, respectively, where the mean is taken across four different real graphs. Similarly, the mean accuracy of PLC model is 90.0% and 89.8 % for R_g and W_s, respectively. The proposed framework has sufficiently high accuracy in detecting critical nodes with both the robustness metrics, even though the model has never seen the real-world graph during the training period. This works because the nodes sub-graphs of real-world networks could match with that of synthetic graphs and therefore the model might have counter that type of sub-graphs during the training period. Furthermore, the models trained on PLC graphs perform better than that of PL graphs. The reason is that the PLC graphs manifest both power-law degree distribution and clusters, hence, are more accurate depiction of real-world networks compared to PL graphs. In addition, the model performance for any alternate graph family or robustness metrics can be enhanced via efficient re-tuning. In this regard, we implement transfer learning by tuning the model trained on PLC graph for estimating criticality scores in real-world bio-yeast network. Although we have used 150 nodes for tuning, the model performance has been increased by 2.7% compared to the scenario when the model trained solely on PLC graphs is used for estimation. Thus, ILGR can accurately identify critical nodes for any large network in a very efficient manner which further strengthens its scalability.

Along with the accurate identification of nodes, the proposed framework provides an appreciable advantage in execution time which is indicated by the running times in Table 3. The proposed method is multiple orders faster than the conventional approach, and this gap will increase as the network size grows. All the training and experiments are conducted on a system with an Intel i9 processor running at 3.4 GHz with 6 GB Nvidia RTX 2070 GPU. The time reported for the proposed approach only includes the prediction time as training is done offline. Even the training time of the proposed method is relatively less than that of the conventional approach.

4.5. Results and discussion for link identification

This subsection assesses the performance of the proposed ILGR framework in the task of critical link identification. The validation is done across different graph sizes and graph types. Separate models are trained for all possible combinations of graph type and metric type, which results in four distinct models. For each model, 30 different random graphs of dimension varying from 1000 to 2000 links are generated for training. The ground truth for the selected links is computed with Algorithm 3. The training model is similar to that of node except that a layer is added that generates a link embedding vector from an associated pair of node embeddings. Consequently, the link embedding is stacked with a regression module to predict link criticality score. The trained model is then used to predict link scores in graphs of higher dimensions, i.e., $1000, 2000, 10000$, and 20000 links. Table 4 reports accuracy for R_g and W_s in PL and PLC graphs. It can be observed that the mean accuracy of PL model in detecting Top-5% of the critical links is 91% and 94% for R_g and W_s both, respectively. Similarly, the mean accuracy of PLC model is 97.5% and 96.1% for R_g and W_s, respectively. The model has fairly high identification accuracy even though the mean is taken across graphs of higher dimensions than that of training. There is a very nominal fall of accuracy with increasing size although the graph size scales in the order of two. Furthermore, Table 4 depicts the comparison of the proposed ILGR with the GCN algorithm trained for networks with 1000 links. It can be inferred that the proposed approach outperforms the GCN model for both PL and PLC networks as well as allows identification of critical links in networks of different sizes.

The scalability and generalizability of our approach in the task of link identification are further supported by evaluating model performance on real-world networks. Table 5 tabulates the Top-5% accuracy in four real-world networks. The accuracy is reported for all the four different models that have been trained on synthetic graphs. It can be inferred that the model has sufficiently high accuracy in identifying critical nodes for both robustness metrics, even though the model has never seen the real-world graph during the training period. Compared to PL graphs, models trained on PLC graphs seem to have high accuracy for real-world networks and the reason is the same as discussed earlier. Further, the execution times in Table 6 demonstrate the computational efficiency of the proposed method.

Although the performance of ILGR in the tasks of node identification and link identification from the estimated criticality scores are similar to a large extent (due to common framework, i.e., node embeddings of sub-graphs), there are some differences. First of all, the overall performance of ILGR for link identification is better than

Table 2
Accuracy of node identification task in real-world graphs. Ntest: # test nodes.

Robustness metric	Rg			Ws		
graph/scores	Ntest	Model	Top-5%	Ntest	Model	Top-5%
bio-yeast	1500	Rg-pl	0.895	1500	ws-pl	0.877
bio-yeast	1500	Rg-plc	0.912	1500	ws-plc	0.898
US powergrid	4941	Rg-pl	0.86	4941	ws-pl	0.923
US powergrid	4941	Rg-plc	0.928	4941	ws-plc	0.914
Wiki-vote	7115	Rg-pl	0.865	7115	ws-pl	0.893
Wiki-vote	7115	Rg-plc	0.892	7115	ws-plc	0.887
cit-DBLP	12591	Rg-pl	0.875	7115	ws-pl	0.889
cit-DBLP	12591	Rg-plc	0.878	7115	ws-plc	0.895

Table 3
Running time of node identification task.

graph/specs	Ntest	Model	Time: proposed (s)	Time: conventional(s)
PL	5000	Rg-pl	17	64600
PLC	5000	Rg-plc	17	64600
US Powergrid	4941	Rg-plc	16	64212
Wiki-Vote	7115	Rg-plc	23	86420

Table 4
Accuracy of link identification task in synthetic graphs.

Test graph size	Ntest 1000		Ntest 2000		Ntest 10000		Ntest 20000	
graph/scores	Rg	Ws	Rg	Ws	Rg	Ws	Rg	Ws
PL (200–2000): **ILGR**	0.979	0.991	0.970	0.985	0.963	0.982	0.953	0.97
PLC (200–2000): **ILGR**	0.981	0.968	0.98	0.961	0.972	0.96	0.967	0.957
PLC (1000): GCN	0.975	0.987	-	-	-	-	-	-
PLC (1000): GCN	0.989	0.974	-	-	-	-	-	-

Table 5
Accuracy of link identification task in real-world graphs. Ntest: # test links.

Robustness metric	Rg			Ws		
graph/scores	Ntest	Model	Top-5%	Ntest	Model	Top-5%
bio-yeast	1948	Rg-pl	0.920	1948	ws-pl	0.904
bio-yeast	1948	Rg-plc	0.926	1948	ws-plc	0.952
US powergrid	6594	Rg-pl	0.946	6594	ws-pl	0.91
US powergrid	6594	Rg-plc	0.931	6594	ws-plc	0.946
Wiki-vote	30000	Rg-pl	0.887	30000	ws-pl	0.871
Wiki-vote	30000	Rg-plc	0.896	30000	ws-plc	0.925
cit-DBLP	49743	Rg-pl	0.870	49743	ws-pl	0.831
cit-DBLP	49743	Rg-plc	0.872	49743	ws-plc	0.892

Table 6
Running time of link identification task.

graph/specs	Ntest	Model	Time: proposed (s)	Time: conventional (s)
PL	5000	Rg-pl	19	64830
PLC	5000	Rg-plc	19	64830
US Powergrid	6594	Rg-plc	21	65470
Wiki-Vote	10000	Rg-plc	25	88231

that of node identification as seen from the Tables 2–5. This is because, a link embedding vector is dependent on two nodes' embedding vectors, thereby including more information compared to the node case which solely uses single node embeddings. The second difference can be seen in their execution times. Link identification takes more time than that of a node as it involves an extra operation (i.e., the combination of node pair embeddings to generate link embedding) apart from common executions.

4.6. Computational complexity of algorithms

The proposed ILGR framework consists of an embedding module and a regression module. The embedding module is based on GraphSAGE algorithm which involves aggregation and combination operation, and basically it is a matrix multiplication of node feature vector with weight matrix. Similarly, the regression module consists of feed forward layers which require matrix multiplication of weights with node embedding vectors. However, these operations are not dependent on the graph size (i.e., number of nodes/links) and are rather dependent on the size of the node feature/embedding vector and number of layers as discussed in [28]. Therefore, with respect to graph size (N), the overall time complexity of proposed ILGR framework is constant which in terms of Big O can be written as $O(1)$. On the other hand, the time complexity of conventional optimization or analytical based approaches is dependent on the number of nodes/links of the graph. For instance, an exhaustive search to identify a critical link is of order $O\left(N^5\right)$ for a graph of N nodes [19]. Although, certain analytical approximation algorithms are developed, the lowest achieved complexity for the case of link identification is of order $O\left(N^2\right)$ [2]. Thus, $O(1)$ makes the proposed method far more efficient than any of the conventional approaches.

5. Conclusions and Future work

This paper proposes a graph neural network based ILGR framework for fast identification of critical nodes and links in large complex networks. Criticality score is defined based on two graph robustness metrics, i.e., effective graph resistance and weighted spectrum. ILGR framework consists of two parts, where in the first part, a graph neural network based embedding and regression model are trained end to end on synthetic graphs with a small subset of nodes/links. The second part deals with the prediction of scores for unseen nodes/links of the graph. The Top-5% identification accuracy of the model is more than 90% for both the robustness metrics. Further, the scalability of the model is shown by identifying critical nodes/links on real-world networks. The proposed approach is multiple orders faster compared to the conventional method. As part of future work, we will systematically incorporate graph/model uncertainty, and extend the framework from single node/link analysis to concurrent case where criticality scores would be assigned to a group of nodes/links. Then, the task would be to identify a critical set of nodes/links, and this could find potential applications in attack graphs.

CRediT authorship contribution statement

Sai Munikoti: Conceptualization, Methodology, Writing - original draft, Software. **Laya Das:** Methodology, Writing - review & editing. **Balasubramaniam Natarajan:** Validation, Writing - review & editing.

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.

Acknowledgement

This material is based upon work supported by National Science Foundation under award number 1855216.

References

[1] Z. Lu, W. Lv, Y. Cao, Z. Xie, H. Peng, B. Du, Lstm variants meet graph neural networks for road speed prediction, Neurocomputing 400 (2020) 34–45.
[2] P. Bongini, M. Bianchini, F. Scarselli, Molecular generative graph neural networks for drug discovery, Neurocomputing 450 (2021) 242–252.
[3] S. Munikoti, K. Lai, B. Natarajan, Robustness assessment of hetero-functional graph theory based model of interdependent urban utility networks, Reliability Engineering & System Safety 107627 (2021).
[4] A. Arulselvan, C.W. Commander, L. Elefteriadou, P.M. Pardalos, Detecting critical nodes in sparse graphs, Computers & Operations Research 36 (7) (2009) 2193–2200.
[5] N. Chaurasia, A. Tiwari, On the use of brokerage approach to discover influencing nodes in terrorist networks, Social Networking, Springer (2014) 271–295.
[6] N. Pandey, A. Pal, et al., Impact of digital surge during covid-19 pandemic: A viewpoint on research and practice, International Journal of Information Management 102171 (2020).
[7] M. Kashif, M.K. Javed, D. Pandey, et al., A surge in cyber-crime during covid-19, Indonesian Journal of Social and Environmental Issues (IJSEI) 1 (2) (2020) 48–52.
[8] F. Yang, K. Fan, D. Song, H. Lin, Graph-based prediction of protein-protein interactions with attributed signed graph embedding, BMC bioinformatics 21 (1) (2020) 1–16.
[9] M.J. Alenazi, J.P. Sterbenz, Comprehensive comparison and accuracy of graph metrics in predicting network resilience, in: 2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), IEEE, 2015, pp. 157–164.
[10] V. Boginski, C.W. Commander, Identifying critical nodes in protein-protein interaction networks, in: Clustering challenges in biological networks, World Scientific, 2009, pp. 153–167..
[11] X. Wang, E. Pournaras, R.E. Kooij, P. Van Mieghem, Improving robustness of complex networks via the effective graph resistance, The European Physical Journal B 87 (9) (2014) 221.
[12] X. Wang, Y. Koç, R.E. Kooij, P. Van Mieghem, A network approach for power grid robustness against cascading failures, in: 2015 7th international workshop on reliable networks design and modeling (RNDM), IEEE, 2015, pp. 208–214..
[13] H. Wang, P. Van Mieghem, Algebraic connectivity optimization via link addition, in: Proceedings of the 3rd International Conference on Bio-Inspired Models of Network, Information and Computing Sytems, 2008, pp. 1–8..
[14] P. Van Mieghem, D. Stevanović, F. Kuipers, C. Li, R. Van De Bovenkamp, D. Liu, H. Wang, Decreasing the spectral radius of a graph by link removals, Physical Review E 84 (1) (2011) 016101.
[15] C. Pizzuti, A. Socievole, A genetic algorithm for enhancing the robustness of complex networks through link protection, in: International Conference on Complex Networks and their Applications, Springer, 2018, pp. 807–819.
[16] E.-Y. Yu, Y.-P. Wang, Y. Fu, D.-B. Chen, M. Xie, Identifying critical nodes in complex networks via graph convolutional networks, Knowledge-Based Systems 198 (2020) 105893.
[17] M. Sun, Y. Jiang, Y. Wang, H. Xie, Z. Wang, Identification of critical nodes in dynamic systems based on graph convolutional networks, in: 2020 3rd International Conference on Unmanned Systems (ICUS), IEEE, 2020, pp. 558–563.
[18] M.J. Alenazi, J.P. Sterbenz, Evaluation and comparison of several graph robustness metrics to improve network resilience, in: 2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM), IEEE, 2015, pp. 7–13..
[19] W. Ellens, R.E. Kooij, Graph measures and network robustness, arXiv preprint arXiv:1311.5064 (2013)..
[20] W. Ellens, F. Spieksma, P. Van Mieghem, A. Jamakovic, R. Kooij, Effective graph resistance, Linear Algebra and its Applications 435 (10) (2011) 2491–2506.
[21] D. Fay, H. Haddadi, A. Thomason, A.W. Moore, R. Mortier, A. Jamakovic, S. Uhlig, M. Rio, Weighted spectral distribution for internet topology analysis: theory and applications, IEEE/ACM Transactions on networking 18 (1) (2009) 164–176.
[22] X. Long, D. Tipper, T. Gomes, Measuring the survivability of networks to geographic correlated failures, Optical Switching and Networking 14 (2014) 117–133.
[23] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016)..
[24] X. Liu, Y. Li, R. Xia, Adaptive multi-view graph convolutional networks for skeleton-based action recognition, Neurocomputing 444 (2021) 288–300.
[25] X. Yu, S. Lu, L. Guo, S.-H. Wang, Y.-D. Zhang, Resgnet-c: A graph convolutional neural network for detection of covid-19, Neurocomputing 452 (2021) 592–605.
[26] Z. Liang, J. Du, Y. Shao, H. Ji, Gated graph neural attention networks for abstractive summarization, Neurocomputing 431 (2021) 128–136.
[27] P. Yan, L. Li, M. Jin, D. Zeng, Quantum probability-inspired graph neural network for document representation and classification, Neurocomputing 445 (2021) 276–286.
[28] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in neural information processing systems, 2017, pp. 1024–1034.
[29] C. Fan, L. Zeng, Y. Ding, M. Chen, Y. Sun, Z. Liu, Learning to identify high betweenness centrality nodes from scratch: A novel graph neural network

approach, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 559–568.
[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017)..
[31] J. Yamanaka, S. Kuwashima, T. Kurita, Fast and accurate image super resolution by deep cnn with skip connection and network in network, in: International Conference on Neural Information Processing, Springer, 2017, pp. 217–225.
[32] M. Tu, X. Zhang, Speech enhancement based on deep neural networks with skip connections, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 5565–5569.
[33] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, H. Li, Ranking measures and loss functions in learning to rank, Advances in Neural Information Processing Systems 22 (2009) 315–323.
[34] J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, url:http://snap.stanford.edu/data (Jun. 2014)..
[35] R.A. Rossi, N.K. Ahmed, The network data repository with interactive graph analytics and visualization, in: AAAI, 2015. url:http://networkrepository.com.
[36] A.-L. Barabási, R. Albert, H. Jeong, Scale-free characteristics of random networks: the topology of the world-wide web, Physica A: statistical mechanics and its applications 281 (1–4) (2000) 69–77.
[37] M.E. Newman, Models of the small world, Journal of Statistical Physics 101 (3–4) (2000) 819–841.
[38] P. Holme, B.J. Kim, Growing scale-free networks with tunable clustering, Physical Review E 65 (2) (2002) 026107.
[39] A. Hagberg, P. Swart, D.S Chult, Exploring network structure, dynamics, and function using networkx, Tech. rep., Los Alamos National Lab. (LANL), Los Alamos, NM (United States) (2008)..

Sai Munikoti Sai Munikoti received the B.Tech. degree in electrical engineering from the Birsa Institute of Technology Sindri, India, in 2015, and the M.Tech. degree in electrical engineering from Indian Institute of Technology Gandhinagar, India, in 2018. He is currently pursuing Ph.D. with the department of Electrical and Computer engineering at Kansas State University, Manhattan, KS, USA. His research interests span areas of Graph Machine learning and Resilience analysis.

Laya Das Laya Das received his B. Tech. degree in Electronics and Telecommunications from Biju Patnaik University of Technology, Odisha, India (2011), M. Tech. (2014) and PhD degree (2018) in Electrical Engineering from Indian Institute of Technology Gandhinagar. He is currently a postdoctoral researcher in Reliability and Risk Engineering Lab, ETH Zurich, Switzerland. His research interests lie in application of signal processing, machine learning, optimization, control and resilience analysis of cyber-physical systems.

Balasubramaniam Natarajan Balasubramaniam Natarajan (SM'08) received the B.E. degree (Hons.) in electrical and electronics engineering from the Birla Institute of Technology and Science at Pilani, Pilani, India, in 1997, the Ph.D. degree in electrical engineering from Colorado State University, Fort Collins, CO, USA, in 2002, and the Ph.D. degree in statistics from Kansas State University, Manhattan, KS, USA, in 2018. He is currently a Clair N. Palmer and Sara M. Palmer Endowed Professor and the Director of the Wireless Communication and Information Processing Research Group, Kansas State University. He has worked on and published extensively on modeling, analysis and networked estimation, and the control of smart distribution grids and cyber physical systems in general. He has published more than 200 refereed journal and conference papers. His research interests include statistical signal processing, stochastic modeling, optimization, and control theories. He has served on the editorial board for multiple IEEE journals including the IEEE Transactions on Wireless communications.