Network Reliability Analysis through Survival Signature and Machine Learning Techniques

Yan Shi [a*], Jasper Behrensdorf [a], Jiayan Zhou [b], Yue Hu [a], Matteo Broggi [a], Michael Beer [a, c, d]

[a] *Institute for Risk and Reliability, Leibniz Universität Hannover, Hannover 30167, Germany*
[b] *College of Water Conservancy and Hydropower Engineering, Hohai University, Nanjing 210098, China*
[c] *Institute for Risk and Uncertainty, University of Liverpool, Liverpool L69 7ZF, United Kingdom*
[d] *International Joint Research Center for Resilient Infrastructure & International Joint Research Center for Engineering Reliability and Stochastic Mechanics, Tongji University, Shanghai 200092, China*

Abstract: As complex networks become ubiquitous in modern society, ensuring their reliability is crucial due to the potential consequences of network failures. However, the analysis and assessment of network reliability become computationally challenging as networks grow in size and complexity. This research proposes a novel graph-based neural network framework for accurately and efficiently estimating the survival signature and network reliability. The method incorporates a novel strategy to aggregate feature information from neighboring nodes, effectively capturing the response flow characteristics of networks. Additionally, the framework utilizes the higher-order graph neural networks to further aggregate feature information from neighboring nodes and the node itself, enhancing the understanding of network topology structure. An adaptive framework along with several efficient algorithms is further proposed to improve prediction accuracy. Compared to traditional machine learning-based approaches, the proposed graph-based neural network framework integrates response flow characteristics and network topology structure information, resulting in highly accurate network reliability estimates. Moreover, once the graph-based neural network is properly constructed based on the original network, it can be directly used to estimate network reliability of different network variants, i.e., sub-networks, which is not feasible with traditional non-machine learning methods. Several applications demonstrate the effectiveness of the proposed method in addressing network reliability analysis problems.

Keywords: Network reliability; Survival signature; Graph-based neural network; Adaptive framework; Machine learning

1. Introduction

Complex technological networks, such as power plant networks, transportation networks,

[*] Corresponding author
E-mail: yan.shi@irz.uni-hannover.de

communication networks, and others, are pervasive in modern society. These networks are deeply integrated into the infrastructure of modern society, and their failure can have serious consequences on society's well-being. Consequently, there is a growing demand for modern technological networks to exhibit high reliability in their operations [1]. It is therefore essential to analyze the reliability of networks, which measures their ability to provide the required service while considering component or link uncertainties, during their design and operation [2]. However, as these networks increase in size and complexity, the analysis and assessment of their reliability require significant computational effort. This necessitates the efficient estimation of network reliability to be of utmost importance.

Currently, the traditional approaches for calculating network reliability can be broadly classified into four categories: enumeration methods, direct methods, decomposition methods, and simulation methods. Enumeration methods typically involve complete state enumeration or more advanced techniques like minimal path or minimal cut enumeration [3]. Direct methods aim to calculate network reliability directly from the underlying graph structure, without the need for a preliminary search for minimal paths or cuts [4]. Decomposition methods involve dividing the network into subnetworks, and the overall reliability is then computed based on the reliabilities of these subnetworks [5]. Specifically, Lee and Park [6] introduced a network reliability analysis method that relies on the principles of additivity and eligibility probabilities. This approach involves the identification of a composite path as a subnetwork, typically comprising a greater number of simpler paths than those encompassed in the composition. Yeh [7, 8] developed algorithms based on binary-addition trees to efficiently analyze network reliability, addressing scenarios where network components can exist in either a fully operational or completely failed state. Zuo et al. [9] proposed an efficient recursive algorithm known as the sum of disjoint products algorithm for multi-state network reliability analysis, which assumes that all minimal paths to specific network state are precomputed using the implicit enumeration algorithm [10]. All of the aforementioned methods, in one way or another, rely on combinatorial exhaustive search through networks. However, since the calculation of network reliability is an NP-hard problem [11], it is generally impossible to estimate the reliability of large-scale networks using these methods. As a result, simulation methods that focus on approximating the statistical properties of networks have been studied for calculating the reliability of large-scale networks. For instance, Yeh et al. [12] developed a particle swarm optimization approach based on Monte Carlo simulation (MCS) to solve complex network reliability and optimization problems. Zuev et al. [1] proposed a stochastic framework that utilizes Subset simulation and a Markov chain Monte Carlo technique to estimate the reliability of large-scale networks. Ramirez-Marquez and Rocco [13] formulated a stochastic network interdiction optimization model with network reliability as the constraint and utilized MCS to estimate the network reliability for each interdiction strategy. Chang [14] introduced an algorithm based on MCS with demand confirmation to compute the two-terminal multi-state network reliability problem and explored the degradation of reliability over time.

Another effective approach for addressing network reliability analysis is based on the survival signature [15], which extends the concept of the system signature [16] to handle systems/networks with multiple component types. In the context of binary-state systems/networks, the survival signature can be defined as the probability of the system/network functionality given a specific number of operational components for each component type [15]. In the case of multi-state systems/networks, the survival signature can be defined as the probability of the system/network being in a specific state given a particular number of functional components for each component type [17]. The survival signature has a distinct advantage as it completely decouples the system/network structure from the probabilistic model used to describe component failures. Compared to traditional methods, survival signature-based approaches only require calculating the survival signature once, allowing them to handle network reliability analysis problems under various scenarios, such as common causes of failure and imprecise uncertainty. For binary-state systems/networks, Feng et al. [18] introduced a survival signature-based method explicitly incorporating imprecise uncertainties to calculate upper and lower bounds of system reliability. Aslett et al. [19] employed the survival signature from a Bayesian perspective for reliability analysis of systems and networks. Patelli et al. [20] proposed two simulation algorithms for computing system reliability with non-repairable components and one simulation algorithm for systems with repairable components. Huang et al. [21] developed a methodology based on the survival signature for reliability analysis of phased mission systems with similar component types in each phase. Salomon et al. [22] provided an efficient method that combines the concepts of survival signature, fuzzy probability theory, and two versions of non-intrusive stochastic simulation methods to quantify the reliability of complex systems. In the domain of multi-state systems with multi-state components, both in discrete and continuous scenarios, Liu et al. [23] derived expressions for stress-strength reliability in both contexts. However, estimating the survival signature is greatly affected by the curse of dimensionality. To enhance the computational efficiency of survival signature estimation, Reed [24] proposed a method that transforms the fault tree representation of systems into a binary decision diagram. This method performs exceptionally well when the fault tree or binary decision diagram is already known. Yi et al. [25] introduced a matrix-based representation of the survival signature for multi-state coherent systems with multi-state components, and developed a finite Markov chain embedding method for estimating the survival signature in the context of multi-state consecutive type systems. Additionally, Qin and Coolen [17] devised several effective methods for calculating the survival signature for various types of multi-state systems. Behrensdorf et al. [26] introduced an efficient approach for calculating the survival signature of binary-state networks, utilizing percolation theory and MCS. In this approach, percolation theory is employed to identify areas of the survival signature that can be safely excluded, while MCS is used to approximate the remaining entries. However, estimating the survival signature and reliability of large-scale networks remains a significant challenge, particularly in cases where the network's topology structure is subject to change.

Considering the advantages of machine learning methods in tackling complex engineering problems, various techniques have been employed in practical applications. These methods encompass artificial neural networks (ANN) [27, 28], convolutional neural networks (CNN) [29, 30], and recurrent neural networks (RNN) [31, 32], among others. It is worth noting that existing machine learning-based network reliability analysis methods usually establish a surrogate model that associates different component states with network performances, treating distinct component states as separate input vectors. This procedure disregards the valuable topology structure information inherent in networks. Actually, integrating the topology structure information becomes crucial for enhancing the prediction accuracy of reliability analysis. Fortunately, graph-based neural networks, such as graph convolutional neural networks (GCNN) [33] and higher-order graph neural networks (HGNN) [34], excel in handling problems with graph features. These techniques effectively capture the topology structure information of networks/graphs, enabling more accurate prediction solutions than traditional artificial neural networks.

This work presents a graph-based neural network framework for estimating the survival signature and the reliability of complex networks. The main contributions encompass three key aspects. Firstly, a novel method is proposed to aggregate feature information from neighboring nodes, effectively integrating network response flow characteristics, forming the foundational framework of the developed graph-based neural network. Secondly, an adaptive framework is established to enhance the prediction accuracy of the graph-based neural network. Thirdly, an efficient learning function along with several algorithms is developed to facilitate the adaptive update process of the graph-based neural network. In comparison to traditional machine learning-based frameworks, the proposed graph-based neural network framework not only integrates response flow characteristics but also incorporates the topology structure information of networks, resulting in highly accurate estimates of network reliability. Moreover, once the graph-based neural network is properly constructed, it can directly estimate network reliability of several sub-networks, which is an ability not feasible with traditional non-machine learning methods.

The structure of this work is organized as follows: The basic definition of network reliability analysis using the survival signature is provided in Section 2. The proposed graph-based neural network framework for network reliability analysis is illustrated in Section 3. The estimation procedure is outlined in Section 4. Several applications are introduced in Section 5. Conclusions are drawn in Section 6.

2. Network Reliability Analysis using Survival Signature

Consider a network consisting of m components, and the state vector of components is expressed as $\mathbf{X}=\left(X_1, X_2,\ldots, X_m\right)$, in which $X_i \in \{0,1\}\left(i=1,2,\ldots,m\right)$ with $X_i=1$ and $X_i=0$ represent the i-th component is in working state and failure state, respectively. The state of the network can be described by the structure function $\phi(\mathbf{X})\in\{0,1\}$ with $\phi(\mathbf{X})=1$ if the network is operational and $\phi(\mathbf{X})=0$ if not. Considering the network contains K types of components with m_k representing the number of components of k-th type and $\sum_{k=1}^{K} m_k = m$. Then, the state vector of components can be written as $\mathbf{X}=\left(\mathbf{X}^1,\mathbf{X}^2,\ldots,\mathbf{X}^K\right)$, in which $\mathbf{X}^k=\left(X_1^k, X_2^k,\ldots,X_{m_k}^k\right)$ represents the state vector of components of k-th type.

The survival signature denoted by $\Phi\left(l_1,l_2,\ldots,l_K\right)\left(l_k=0,1,\ldots,m_k;k=1,2,\ldots,K\right)$ is defined to be the probability that the network functions given that l_k of its m_k components of type k for each $k\in\{1,2,\ldots,K\}$ [15]. There are $C_{m_k}^{l_k}$ state vectors $\mathbf{X}^k$ with precisely l_k components X_i^k equal to 1, thus $\sum_{i=1}^{m_k} X_i^k = l_k \left(k=1,2,\ldots,K\right)$. Denote $S_{l_1,l_2,\ldots,l_K}$ as the set of all state vectors for the whole network, and the magnitude of this set is $\prod_{k=1}^{K} C_{m_k}^{l_k}$. It is assumed that the failure times of the same type component are independently and identically distributed or exchangeable, then the survival signature can be expressed as follows:

$$\Phi(l_1,l_2,...,l_K)=\left(\prod_{k=1}^{K}C_{m_k}^{l_k}\right)^{-1}\times\sum_{\mathbf{X}\in S_{l_1,l_2,...,l_K}}\phi(\mathbf{X}) \tag{1}$$

The expression above indicates that the survival signature solely relies on the network topology structure, irrespective of the time-dependent failure behavior of components.

Let $L_k(t)\in\{0,1,...,m_k\}$ denote the number of components of type k in working state at time t and suppose the cumulative distribution function of failure times of type k to be known with $F_k(t)$, then the probability when there are l_k components in working state at time t for each $k\in\{1,2,...,K\}$ is calculated by:

$$P\left(\bigcap_{k=1}^{K}\{L_k(t)=l_k\}\right)=\prod_{k=1}^{K}P(L_k(t)=l_k)=\prod_{k=1}^{K}C_{m_k}^{l_k}\left[F_k(t)\right]^{m_k-l_k}\left[1-F_k(t)\right]^{l_k} \tag{2}$$

where $P(\bullet)$ represents the probability operator. The probability depicted above is computed considering the time-dependent failure behavior of components, without taking into account the network topology structure.

The reliability or survival function of the network, which describes the probability of the network being in an operational state at time t, can be expressed as follows:

$$R(t)=P\{T_S>t\}=\sum_{l_1=0}^{m_1}\cdots\sum_{l_K=0}^{m_K}\Phi(l_1,l_2,...,l_K)P\left(\bigcap_{k=1}^{K}\{L_k(t)=l_k\}\right) \tag{3}$$

in which T_S represents the network failure time. It is evident from Equation (3) that the survival signature effectively separates the network's structure from the failure time distribution of components, which represents its primary advantage. Moreover, calculating the survival signature just once for any given network is sufficient to determine its reliability.

In this study, we focus on the reliability of a two-terminal network, which quantifies the probability of establishing a connection between the terminal node and the source node. It is assumed that both the terminal node, the source node, and all the links in the network are sufficiently robust and will not experience failures during network operation. For each realization of the state vector of components, i.e., $\boldsymbol{x}=(x_1,x_2,...,x_m)$, the value of structure function $\phi(\boldsymbol{x})$ can be calculated by the Dijkstra algorithm [35]. The Dijkstra algorithm is a highly efficient technique for solving the shortest path problem in networks. It aims to determine whether there exists a shortest path given the realization of the state vector of components. If a shortest path exists, it is identified that the value of the structure function to be

$\phi(\boldsymbol{x})=1$; otherwise, the value of the structure function is $\phi(\boldsymbol{x})=0$. However, computing the survival signature for all combinations of state vectors of components exactly is computationally demanding, particularly for large-scale networks. Considering this challenge, the MCS method [26] can be employed to approximate the survival signature with a high accuracy. When estimating the survival signature $\Phi(l_1,l_2,...,l_K)$ under the case $\{l_1,l_2,...,l_K\}$ with the MCS method, N_{MCS} samples $\boldsymbol{x}^{(j)}=\left(\boldsymbol{x}^{1(j)},\boldsymbol{x}^{2(j)},...,\boldsymbol{x}^{K(j)}\right)\left(j=1,2,...,N_{\text{MCS}}\right)$ are randomly generated, in which $\boldsymbol{x}^{k(j)}=\left(x_1^{k(j)},x_2^{k(j)},...,x_{m_k}^{k(j)}\right)$ and $\sum_{i=1}^{m_k} x_i^{k(j)}=l_k$ for each $k\in\{1,2,...,K\}$. The survival signature $\Phi(l_1,l_2,...,l_K)$ can be calculated as follows:

$$\Phi(l_1,l_2,...,l_K)=\frac{1}{N_{\text{MCS}}}\sum_{j=1}^{N_{\text{MCS}}}\phi\left(\boldsymbol{x}^{(j)}\right) \tag{4}$$

It should be noted that for the case $\{l_1,l_2,...,l_K\}$ where the number of possible network states $\prod_{k=1}^{K} C_{m_k}^{l_k}$ are smaller than N_{MCS}, the survival signature is calculated analytically through Eq. (1). Moreover, if the number of working components, i.e., $\sum_{k=1}^{K} l_k$, is less than the shortest path (corresponding to components) of the network when all those components are functioning, the network will certainly fail. Therefore, in this case, the survival signature $\Phi(l_1,l_2,...,l_K)$ is directly set to zero without the need for simulation or analytical calculation. After the survival signature is estimated, the network reliability can be obtained through Eq. (3).

Although the MCS method mentioned above is effective in solving the survival signature, it remains computationally demanding for large-scale networks, particularly when the network topology structure changes, such as the deletion of nodes. In such cases, the MCS method needs to be re-executed to obtain the new survival signature for the new network with the updated topology structure. In the following section, a novel graph-based neural network and an adaptive update framework are developed to analyze network reliability. Once this graph-based neural network is appropriately constructed, it can directly estimate network reliability, even in the presence of changes to the network topology structure.

3. Graph-based Neural Network Framework for Network Reliability Analysis

3.1 The Graph-based Neural Network with Novel Aggregation Layers

Graph-based neural networks represent a specialized class of machine learning models engineered for processing and analyzing graph-structured data. Unlike conventional artificial neural networks, which excel at tasks like image recognition and natural language processing, graph-based neural networks are uniquely designed to harness the inherent structure and interconnections within graphs. Graphs consist of nodes and edges that link these nodes. Graph-based neural networks leverage this topology to propagate information across the graph, allowing them to make informed predictions, classifications, or generate useful representations. These models have shown remarkable success in various applications, ranging from social network analysis and recommendation systems to bioinformatics and drug discovery. At the core of graph-based neural networks is the ability to iteratively update node representations by aggregating information from neighboring nodes, effectively capturing the context and relationships within the graph. This process allows graph-based neural networks to encode valuable information about each node's surroundings, making them particularly effective for tasks like node classification, link prediction, and graph classification. Currently, a range of graph-based neural networks have been developed by employing distinct aggregation layers, including graph convolutional neural networks (GCNN) [33] and higher-order graph neural networks (HGNN) [34], etc.

In this work, the network is represented by a graph $G(N, L)$, where the components are considered as the node set N, the connections between components are represented as the link set L. A novel graph-based neural network is proposed to construct a surrogate model between different network topology structures (i.e., component states) and network responses (i.e., structure functions). The graph-based neural network starts by employing a proposed expression to aggregate feature information from neighboring nodes, effectively integrating the response flow characteristics of networks. Subsequently, the HGNN is utilized to further aggregate feature information from both neighboring nodes and the node itself. This step enables the extraction of network topology structure information, thereby improving the algorithm's understanding of networks. The basic framework of the developed graph-based neural network is shown in Fig. 1. When the graph-based neural network has been effectively trained and constructed, it can be directly utilized to efficiently predict network responses under various network topology structures.

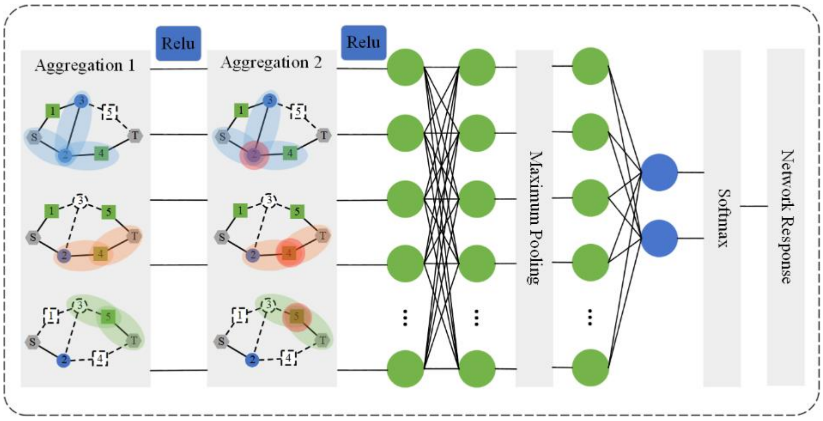


Fig. 1 Basic framework of the developed graph-based neural network

Fig. 2 provides the basic procedure of the proposed graph-based neural network for predicting values of structure function at different network topology structures. In this network, there are two types of components. Type 1 includes components 1, 4, and 5, while type 2 contains components 2 and 3. The failed components are represented by white dashed boxes among all the network topology structures. Initially, several different network topology structures are generated, and the corresponding network responses are calculated using the Dijkstra algorithm. Subsequently, the proposed graph-based neural network is utilized to construct a surrogate model that relates different network topology structures to their corresponding network responses. This surrogate model is then employed to predict the network responses for other network topology structures. By obtaining the network responses for all these network topology structures, the survival signature and network reliability can be easily determined.

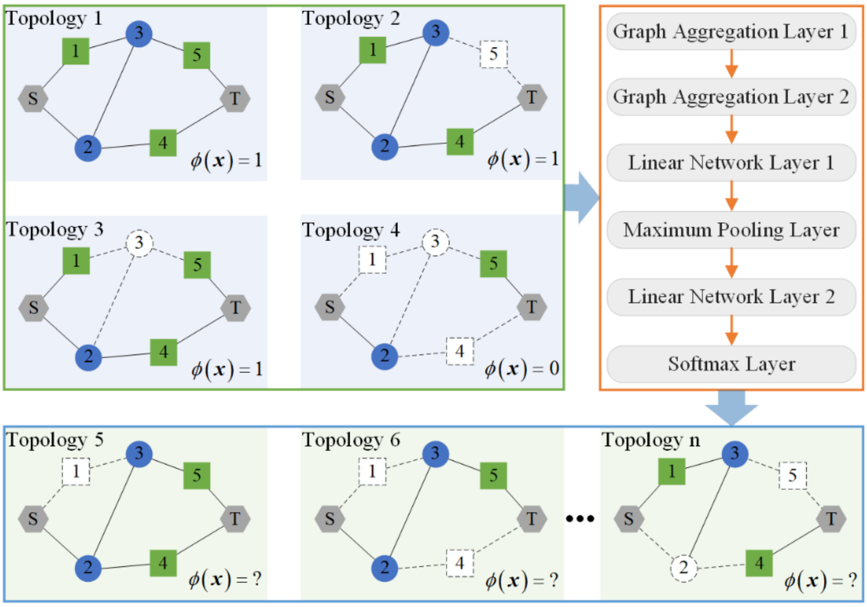


Fig. 2 Basic procedure of the developed graph-based neural network for predicting values of structure function at different network topology structures

To construct the graph-based neural network, the link weight vector is introduced and denoted as $\mathbf{Z}=\left(Z_{1,2},Z_{1,3},\ldots,Z_{i,j},\ldots,Z_{m-1,m}\right)$, where $Z_{i,j}$ means the link weight between the i -th and j -th components. Specifically, the link weight $Z_{i,j}$ is assigned a value of 1 if there is a connection between the i -th and j -th components, and it is set to 0 otherwise. The feature vector of nodes utilized is set to the following expression:

$$\mathbf{X}^{*}=\begin{bmatrix}\mathbf{X}_1^{*}\\ \mathbf{X}_2^{*}\\ \vdots\\ \mathbf{X}_m^{*}\end{bmatrix}=\begin{bmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 1\end{bmatrix}_{m\times m} \tag{5}$$

Then, the following graph aggregation layer 1 is established to aggregate local neighborhood information for node i by integrating link weights:

$$\mathbf{X}_i^{**}=\sum_{j\in N(i)} Z_{j,i}\left[I\left(\mathbf{X}_j^{*}\right)\mathbf{X}_j^{*}\mathbf{W}^{(1)\mathrm{T}}+\mathbf{b}^{(1)}\right] \tag{6}$$

in which $N(i)$ denotes the nodes in neighborhood of node i. $\mathbf{X}_i^{**}$ is the updated feature of node i after graph aggregation layer 1, and the updated feature vector of nodes can be expressed as $\mathbf{X}^{**}=\left[\mathbf{X}_1^{**},\mathbf{X}_2^{**},\ldots,\mathbf{X}_m^{**}\right]^{\mathrm{T}}$. $I\left(\mathbf{X}_j^{*}\right)$ represents the indicator function of node j, and $I\left(\mathbf{X}_j^{*}\right)=1$ if component j is functional and $I\left(\mathbf{X}_j^{*}\right)=0$ otherwise. $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ represents the weight parameter vector and bias parameter vector of the linear neural network shown below:

$$\mathbf{W}^{(1)}=\begin{bmatrix}W_{1,1}^{(1)} & W_{1,2}^{(1)} & \cdots & W_{1,m}^{(1)}\\ W_{2,1}^{(1)} & W_{2,2}^{(1)} & \cdots & W_{2,m}^{(1)}\\ \vdots & \vdots & \ddots & \vdots\\ W_{n_H,1}^{(1)} & W_{n_H,2}^{(1)} & \cdots & W_{n_H,m}^{(1)}\end{bmatrix} \tag{7}$$

$$\mathbf{b}^{(1)}=\left[b_1^{(1)},b_2^{(1)},\ldots,b_{n_H}^{(1)}\right] \tag{8}$$

where n_H means the number of neurons in the hidden layer. In this work, n_H is identified as follows:

$$n_H=\begin{cases}128 & m+2n_L\leq 128\\ 256 & 128<m+2n_L\leq 256\\ 512 & m+2n_L>256\end{cases} \tag{9}$$

in which n_L represents the number of links in networks. The proposed graph aggregation layer 1 shown in Eq. (6) aggregates feature information from neighboring nodes so to effectively integrating the response flow characteristics of networks.

To illustrate the operational process of the proposed aggregation layer 1, we employ a simple network comprising three components as an illustrative example. The operational process of the aggregation layer 1 is depicted in Fig. 3. For component 1, its corresponding neighboring node is component 2. The indicator function for component 2 is represented as $I\left(\mathbf{X}_2^{*}\right)=1$ and the link weight between component 2 and component 1 is $Z_{2,1}=1$. Consequently, utilizing the proposed aggregation layer 1 as outlined in Eq. (6), the updated feature for component 1 can be computed as $\mathbf{X}_1^{**}=Z_{2,1}\left[I\left(\mathbf{X}_2^{*}\right)\mathbf{X}_2^{*}\mathbf{W}^{(1)\mathrm{T}}+\mathbf{b}^{(1)}\right]=\left[W_{1,2}^{(1)}+b_1^{(1)},W_{2,2}^{(1)}+b_2^{(1)},\dots,W_{n_H,2}^{(1)}+b_{n_H}^{(1)}\right]$. Additionally, with the similar operational process, the updated features for component 2 and component 3 can also be determined as $\mathbf{X}_2^{**}=\left[W_{1,1}^{(1)}+b_1^{(1)},W_{2,1}^{(1)}+b_2^{(1)},\dots,W_{n_H,1}^{(1)}+b_{n_H}^{(1)}\right]$ and $\mathbf{X}_3^{**}=\left[0,0,\dots,0\right]_{1\times n_H}$, respectively.

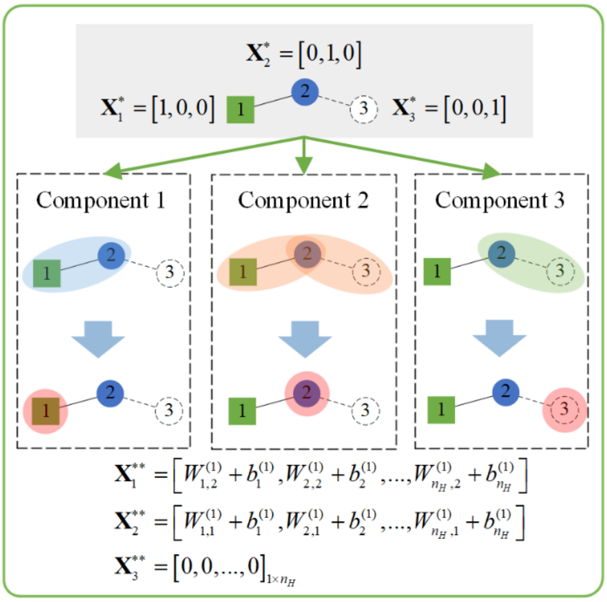

$\mathbf{X}_2^{*}=[0,1,0]$
$\mathbf{X}_1^{*}=[1,0,0]$
$\mathbf{X}_3^{*}=[0,0,1]$
1
2
3
Component 1
Component 2
Component 3
$\mathbf{X}_1^{**}=\left[W_{1,2}^{(1)}+b_1^{(1)},W_{2,2}^{(1)}+b_2^{(1)},\dots,W_{n_H,2}^{(1)}+b_{n_H}^{(1)}\right]$
$\mathbf{X}_2^{**}=\left[W_{1,1}^{(1)}+b_1^{(1)},W_{2,1}^{(1)}+b_2^{(1)},\dots,W_{n_H,1}^{(1)}+b_{n_H}^{(1)}\right]$
$\mathbf{X}_3^{**}=\left[0,0,\dots,0\right]_{1\times n_H}$

Fig. 3 The operational process of the developed aggregation layer 1

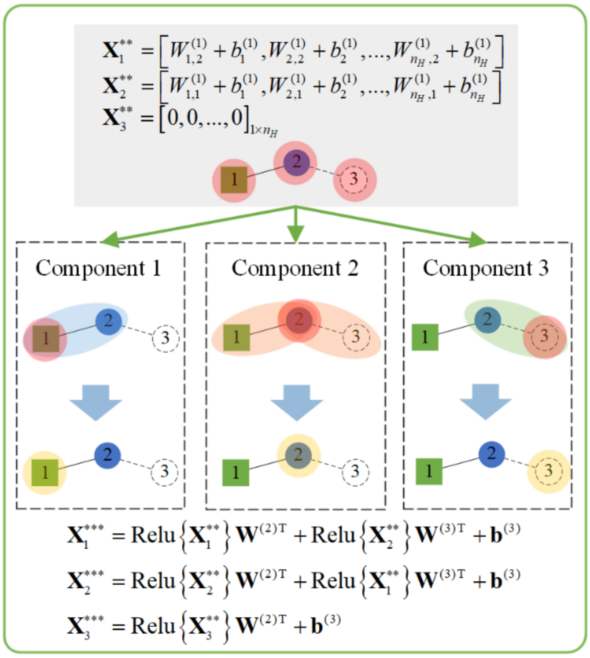
After the graph aggregation layer 1 is constructed, the HGNN [34] is used to be the graph aggregation layer 2 as follows:

$$\mathbf{X}_i^{***}=\mathrm{Relu}\left\{\mathbf{X}_i^{**}\right\}\mathbf{W}^{(2)\mathrm{T}}+\left(\sum_{j\in N(i)}Z_{j,i}\mathrm{Relu}\left\{\mathbf{X}_j^{**}\right\}\right)\mathbf{W}^{(3)\mathrm{T}}+\mathbf{b}^{(3)} \tag{10}$$

in which $\mathrm{Relu}\{\bullet\}$ represents the Relu activation function. $\mathbf{X}_i^{***}$ is the updated feature of node i after graph aggregation layer 2, and the updated feature vector of nodes can be expressed as $\mathbf{X}^{***}=\left[\mathbf{X}_1^{***},\mathbf{X}_2^{***},\dots,\mathbf{X}_m^{***}\right]^{\mathrm{T}}$. $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$ and $\mathbf{b}^{(3)}$ represent the corresponding weight parameter vector and bias parameter vector of linear neural networks, with the size being $n_H\times n_H$, $n_H\times n_H$ and

$1 \times n_H$, respectively. The structures of all these vectors resemble those presented in Eqs. (7) and (8), and therefore, will not be reiterated here. The established graph aggregation layer 2 is utilized to further aggregate feature information from both neighboring nodes and the node itself. This facilitates the extraction of network topology structure information, thereby enhancing the algorithm's comprehension of networks.

The updated feature vector $\mathbf{X}^{**}$ after the aggregation layer 1 is further used as the input of aggregation layer 2 to obtain the update feature vector $\mathbf{X}^{***}$, and the operational process of the aggregation layer 2 is shown in Fig. 4. According to the aggregation layer 2 shown in Eq. (10) and the topology structure of the simple network in Fig. 4, the updated feature for component 1 can be calculated as $\mathbf{X}_1^{***} = \mathrm{Relu}\left\{\mathbf{X}_1^{**}\right\}\mathbf{W}^{(2)\mathrm{T}} + Z_{2,1}\mathrm{Relu}\left\{\mathbf{X}_2^{**}\right\}\mathbf{W}^{(3)\mathrm{T}} + \mathbf{b}^{(3)} = \mathrm{Relu}\left\{\mathbf{X}_1^{**}\right\}\mathbf{W}^{(2)\mathrm{T}} + \mathrm{Relu}\left\{\mathbf{X}_2^{**}\right\}\mathbf{W}^{(3)\mathrm{T}} + \mathbf{b}^{(3)}$. Furthermore, the updated features for component 2 and component 3 can also be determined as $\mathbf{X}_2^{***} = \mathrm{Relu}\left\{\mathbf{X}_2^{**}\right\}\mathbf{W}^{(2)\mathrm{T}} + \mathrm{Relu}\left\{\mathbf{X}_1^{**}\right\}\mathbf{W}^{(3)\mathrm{T}} + \mathbf{b}^{(3)}$ and $\mathbf{X}_3^{***} = \mathrm{Relu}\left\{\mathbf{X}_3^{**}\right\}\mathbf{W}^{(2)\mathrm{T}} + \mathbf{b}^{(3)}$, respectively.



$\mathbf{X}_1^{**} = \left[W_{1,2}^{(1)} + b_1^{(1)}, W_{2,2}^{(1)} + b_2^{(1)}, \ldots, W_{n_H,2}^{(1)} + b_{n_H}^{(1)}\right]$
$\mathbf{X}_2^{**} = \left[W_{1,1}^{(1)} + b_1^{(1)}, W_{2,1}^{(1)} + b_2^{(1)}, \ldots, W_{n_H,1}^{(1)} + b_{n_H}^{(1)}\right]$
$\mathbf{X}_3^{**} = \left[0, 0, \ldots, 0\right]_{1 \times n_H}$
1
2
3
Component 1
Component 2
Component 3
$\mathbf{X}_1^{***} = \mathrm{Relu}\left\{\mathbf{X}_1^{**}\right\}\mathbf{W}^{(2)\mathrm{T}} + \mathrm{Relu}\left\{\mathbf{X}_2^{**}\right\}\mathbf{W}^{(3)\mathrm{T}} + \mathbf{b}^{(3)}$
$\mathbf{X}_2^{***} = \mathrm{Relu}\left\{\mathbf{X}_2^{**}\right\}\mathbf{W}^{(2)\mathrm{T}} + \mathrm{Relu}\left\{\mathbf{X}_1^{**}\right\}\mathbf{W}^{(3)\mathrm{T}} + \mathbf{b}^{(3)}$
$\mathbf{X}_3^{***} = \mathrm{Relu}\left\{\mathbf{X}_3^{**}\right\}\mathbf{W}^{(2)\mathrm{T}} + \mathbf{b}^{(3)}$

Fig. 4 The operational process of the aggregation layer 2

In addition, two additional linear neural networks are employed, along with a maximum pooling layer, to enhance the neural network's ability to address complex network topology problems.

$$\hat{\phi}(\mathbf{X}) = \mathrm{Softmax}\left\{\underset{\mathbf{x}_i^{***} \subset \mathbf{X}^{***}}{\mathrm{Maxpool}}\left\{\mathrm{Relu}\left(\mathbf{X}_i^{***}\right)\mathbf{W}^{(4)\mathrm{T}} + \mathbf{b}^{(4)}\right\} \times \mathbf{W}^{(5)\mathrm{T}} + \mathbf{b}^{(5)}\right\} \tag{11}$$

where $\underset{\mathbf{x}_i^{***} \subset \mathbf{X}^{***}}{\mathrm{Maxpool}}\{\bullet\}$ and $\mathrm{Softmax}\{\bullet\}$ represent the maximum pooling operation and softmax operation, respectively. $\mathbf{W}^{(4)}$, $\mathbf{W}^{(5)}$, $\mathbf{b}^{(4)}$ and $\mathbf{b}^{(5)}$ represent the corresponding vectors of weight parameters and bias parameters of linear neural networks, with the size being $n_H \times n_H$, $2\times n_H$, $1\times n_H$ and 2×1, respectively. Moreover, the structures of all these vectors resemble those presented in Eqs. (7) and (8), and therefore, will not be reiterated here. In this study, we determine all hyperparameters using the adaptive moment estimation (Adam) algorithm [36], with an initial learning rate of 0.01 and weight decay of 0.0001. The Adam algorithm dynamically adapts the learning rate for each parameter by estimating their first-order and second-order moments of gradients. One of its key advantages is that, following bias correction, the learning rate during each iteration falls within a specific range, which helps maintain parameter stability. For more comprehensive information about this algorithm, please refer to Ref. [36].

It is important to emphasize that the suggested graph-based neural network structure can be customized to meet specific mission requirements. For example, the graph aggregation layer can be iterated multiple times, and additional linear neural network layers can be incorporated into the proposed framework to handle large-scale network issues. To enhance the estimation accuracy of the established graph-based neural network for estimating network reliability, an efficient adaptive framework is established in the following subsection.

3.2 Adaptive Network Reliability Analysis

In this adaptive framework for network reliability analysis, the graph-based neural network is continuously updated and strengthened through iterative processes that involve incorporating the most influential samples into the training sample set. This process continues until the prediction accuracy of the graph-based neural network reaches a satisfactory level. Subsequently, the well-constructed graph-based neural network, characterized by high accuracy, enables the estimation of the final network reliability.

Initially, for each case of $\{l_1, l_2, \ldots, l_K\}\left(l_k = 0,1,\ldots,m_k; k=1,2,\ldots,K\right)$, N_{MCS} samples $\mathbf{x}^{(j)}_{l_1,l_2,\ldots,l_K} = \left(\mathbf{x}^{1(j)}_{l_1}, \mathbf{x}^{2(j)}_{l_2}, \ldots, \mathbf{x}^{K(j)}_{l_K}\right)\left(j = 1,2,\ldots,N_{\text{MCS}}\right)$ are randomly generated based on MCS, in which $\mathbf{x}^{k(j)}_{l_k} = \left(x^{k(j)}_{l_k,1}, x^{k(j)}_{l_k,2}, \ldots, x^{k(j)}_{l_k,m_k}\right)$ and $\sum_{i=1}^{m_k} x^{k(j)}_{l_k,i} = l_k$ for each $k \in \{1,2,\ldots,K\}$. It is worth mentioning that if the number of potential network states $\prod_{k=1}^{K} C^{l_k}_{m_k}$ is smaller than N_{MCS}, the samples are obtained from all these potential network states $\boldsymbol{x}^{(j)}_{l_1,l_2,\dots,l_K} = \left(\boldsymbol{x}^{1(j)}_{l_1}, \boldsymbol{x}^{2(j)}_{l_2}, \dots, \boldsymbol{x}^{K(j)}_{l_K}\right)\left(j = 1, 2, \dots, N_{\mathrm{STATE}}\right)$, where N_{STATE} means the number of samples in this case. Additionally, since the network will certainly fail when the number of working components is less than the shortest path (corresponding to components) of the network when all those components are functioning, this is no need to generate the corresponding samples for this case. When all these samples are generated, they are combined to form the total sample set, i.e., $\boldsymbol{x}^{(j)} = \left(\boldsymbol{x}^{1(j)}, \boldsymbol{x}^{2(j)}, \dots, \boldsymbol{x}^{K(j)}\right)\left(j = 1, 2, \dots, N_{\mathrm{TOTAL}}\right)$, in which N_{TOTAL} represents the number of samples in the total sample set.

Then, N_{TRAIN} samples are randomly selected from the total sample set so to form the training sample set, i.e., $\boldsymbol{x}^{\mathrm{Train}(j)} = \left(\boldsymbol{x}^{\mathrm{Train}1(j)}, \boldsymbol{x}^{\mathrm{Train}2(j)}, \dots, \boldsymbol{x}^{\mathrm{Train}K(j)}\right)\left(j = 1, 2, \dots, N_{\mathrm{TRAIN}}\right)$. The Dijkstra algorithm [35] is used to compute the values of structure function, i.e., $\phi\left(\boldsymbol{x}^{\mathrm{Train}(j)}\right)\left(j = 1, 2, \dots, N_{\mathrm{TRAIN}}\right)$, for all these training samples, and the solutions are considered as the response set. The training sample set and the corresponding response set can be used to construct a graph-based neural network as the description shown in subsection 3.1. Additionally, we reshuffle these training samples to get additional N_{SET} sample sets so to construct additional N_{SET} graph-based neural networks for facilitating the convergence criterion and adaptive learning function. The process for reshuffling the training samples is as follows: Firstly, these training samples are divided into approximately equal N_{SET} sample subsets. Secondly, for each $i \in \{1, 2, \dots, N_{\mathrm{SET}}\}$, the subsets without i -th subset are sequentially combined into new subsets. Thirdly, the i -th sample subset is then appended at the end of the corresponding new subsets, resulting in the generation of additional sample sets $\boldsymbol{x}^{\mathrm{SET}(i)(j)} = \left(\boldsymbol{x}^{\mathrm{SET}1(i)(j)}, \boldsymbol{x}^{\mathrm{SET}2(i)(j)}, \dots, \boldsymbol{x}^{\mathrm{SET}K(i)(j)}\right)\left(i = 1, 2, \dots, N_{\mathrm{SET}}; j = 1, 2, \dots, N_{\mathrm{TRAIN}}\right)$. The additional N_{SET} sample sets are employed to construct additional N_{SET} graph-based neural networks. In contrast to the original training sample set, the samples within each additional sample set are divided into training samples and test samples. In each additional sample set, the initial $N_{\mathrm{SET\text{-}TRAIN}}$ samples are chosen as the training samples, i.e., $\boldsymbol{x}^{\mathrm{SET}(i)(j)}\left(i = 1, 2, \dots, N_{\mathrm{SET}}; j = 1, 2, \dots, N_{\mathrm{SET\text{-}TRAIN}}\right)$, while the remaining samples are designated as the test samples, i.e., $\boldsymbol{x}^{\mathrm{SET}(i)(j)}\left(i = 1, 2, \dots, N_{\mathrm{SET}}; j = N_{\mathrm{SET\text{-}TRAIN}} + 1, N_{\mathrm{SET\text{-}TRAIN}} + 2, \dots, N_{\mathrm{TRAIN}}\right)$. The following expression is established to identify the value of $N_{\mathrm{SET\text{-}TRAIN}}$:

$$N_{\text{SET-TRAIN}}=\begin{cases}\text{Round}\left\{\delta N_{\text{TRAIN}}\right\} & \left(1-\delta\right)N_{\text{TRAIN}}\leq N_{\text{ADD}}\\ \text{Round}\left\{\dfrac{N_{\text{TRAIN}}-N_{\text{ADD}}}{N_{\text{TRAIN}}}N_{\text{TRAIN}}\right\} & \text{otherwise}\end{cases}\tag{12}$$

in which $\text{Round}\{\bullet\}$ represents the round operation, and δ is a given scale factor such as $\delta=0.9$. N_{ADD} is the number of newly added training samples in the current iteration, and specific details regarding this parameter are provided later in this work. The expression of $N_{\text{SET-TRAIN}}$ means that at most N_{ADD} samples are set as the test samples in each additional sample set. This processing can guarantee as many training samples as possible, thereby ensuring that the constructed additional N_{SET} graph-based neural networks have sufficient accuracy.

Based on the original training sample set $\boldsymbol{x}^{\text{Train}(j)}\left(j=1,2,...,N_{\text{TRAIN}}\right)$ and the additional N_{SET} sample sets, $N_{\text{SET}}+1$ graph-based neural networks can be trained and constructed. Denote the corresponding prediction solutions for the total sample set as $\hat{\phi}\left(\boldsymbol{x}^{(j)}\right)\left(j=1,2,...,N_{\text{TOTAL}}\right)$ and $\hat{\phi}_i\left(\boldsymbol{x}^{(j)}\right)\left(i=1,2,...,N_{\text{SET}};j=1,2,...,N_{\text{TOTAL}}\right)$, respectively. The survival signature can be further estimated based on Eq. (1) as $\hat{\Phi}\left(l_1,l_2,...,l_K\right)$ and $\hat{\Phi}_i\left(l_1,l_2,...,l_K\right)\left(i=1,2,...,N_{\text{SET}}\right)$ for each case of $\left\{l_1,l_2,...,l_K\right\}\left(l_k=0,1,...,m_k;k=1,2,...,K\right)$, respectively. It is important to highlight that for network reliability calculation, only the predicted survival signature solution $\hat{\Phi}\left(l_1,l_2,...,l_K\right)$ is utilized. The remaining predicted survival signature solutions $\hat{\Phi}_i\left(l_1,l_2,...,l_K\right)\left(i=1,2,...,N_{\text{SET}}\right)$ are employed solely for constructing the convergence criterion of the adaptive framework. In this work, the prediction accuracies of the additional N_{SET} graph-based neural networks are measured as follows:

$$\rho_i=\frac{1}{N_{\text{TRAIN}}}\sum_{j=1}^{N_{\text{TRAIN}}}\left|\phi\left(\boldsymbol{x}^{\text{SET}(i)(j)}\right)+\hat{\phi}_i\left(\boldsymbol{x}^{\text{SET}(i)(j)}\right)-1\right|\tag{13}$$

in which $i=1,2,...,N_{\text{SET}}$. The above formula measures the proportion of correctly predicted samples to the total samples in each additional sample set. The larger the value of ρ_i, the higher the accuracy of the i-th additional graph-based neural network. To enhance the robustness of our analysis, we address the inherent uncertainty that arises during the training of graph-based neural networks by excluding solutions associated with both the highest accuracy $\rho_{\max}=\max\left\{\rho_i\left|i=1,2,...,N_{\text{SET}}\right.\right\}$ and lowest accuracy $\rho_{\min} = \min\left\{\rho_i \,|i = 1, 2, ..., N_{\mathrm{SET}}\right\}$. The remaining prediction solutions for the total sample set and the survival signature are rewritten as $\hat{\phi}_i^*\left(\boldsymbol{x}^{(j)}\right)\left(i = 1, 2, ..., N_{\mathrm{SET}} - 2; j = 1, 2, ..., N_{\mathrm{TOTAL}}\right)$ and $\hat{\Phi}_i^*\left(l_1, l_2, ..., l_K\right)\left(i = 1, 2, ..., N_{\mathrm{SET}} - 2\right)$, respectively. The convergence criterion for the established adaptive framework is determined by considering the standard deviation of the predicted survival signature solutions, using $\hat{\Phi}\left(l_1, l_2, ..., l_K\right)$ as the mean. The corresponding standard deviation is shown below:

$$\sigma_{l_1, l_2, ..., l_K} = \sqrt{\frac{1}{N_{\mathrm{SET}} - 2} \sum_{i=1}^{N_{\mathrm{SET}}-2} \left[\hat{\Phi}\left(l_1, l_2, ..., l_K\right) - \hat{\Phi}_i^*\left(l_1, l_2, ..., l_K\right)\right]^2} \tag{14}$$

Then, the convergence criterion can be established as follows:

$$\sigma_{\max} = \max\left\{\sigma_{l_1, l_2, ..., l_K} \,|l_k = 0, 1, ..., m_k; k = 1, 2, ..., K\right\} \le \sigma_0 \tag{15}$$

where σ_0 is the convergence criterion threshold value. The predicted survival signature $\hat{\Phi}\left(l_1, l_2, ..., l_K\right)$ is considered accurate and this is no need to update these graph-based neural networks unless the convergence criterion specified in Eq. (15) is met.

If the convergence criterion specified in Eq. (15) is not satisfied, the following adaptive learning function is established to select new training samples from the total sample set $\boldsymbol{x}^{(j)} = \left(\boldsymbol{x}^{1(j)}, \boldsymbol{x}^{2(j)}, ..., \boldsymbol{x}^{K(j)}\right)\left(j = 1, 2, ..., N_{\mathrm{TOTAL}}\right)$ in current iteration.

$$F\left(\boldsymbol{x}^{(j)}\right) = 1 - \frac{1}{N_{\mathrm{SET}} - 2} \sum_{i=1}^{N_{\mathrm{SET}}-2} \left|\hat{\phi}\left(\boldsymbol{x}^{(j)}\right) - \hat{\phi}_i^*\left(\boldsymbol{x}^{(j)}\right)\right| \tag{16}$$

in which $i = 1, 2, ..., N_{\mathrm{SET}} - 2$. The learning function quantifies the disparity between the predicted solutions $\hat{\phi}\left(\boldsymbol{x}^{(j)}\right)\left(j = 1, 2, ..., N_{\mathrm{TOTAL}}\right)$ and $\hat{\phi}_i^*\left(\boldsymbol{x}^{(j)}\right)\left(i = 1, 2, ..., N_{\mathrm{SET}} - 2; j = 1, 2, ..., N_{\mathrm{TOTAL}}\right)$. A smaller value of the learning function corresponds to a larger difference among these predicted solutions. When $F\left(\boldsymbol{x}^{(j)}\right) = 1$, the predicted solutions of the sample $\boldsymbol{x}^{(j)}$ exhibit consistency across all these graph-based neural networks. Conversely, as $F\left(\boldsymbol{x}^{(j)}\right)$ approaches zero, it indicates a notable level of epistemic uncertainty in the prediction of the sample $\boldsymbol{x}^{(j)}$. In such scenarios, it is recommended to incorporate this sample into the training sample set in order to update these graph-based neural networks. In this work, N_{ADD} new training samples that corresponding to top N_{ADD} minimum values of learning function are selected to update these graph-based neural networks in each iteration. Furthermore, the following expression is established to identify the number of new training samples added to the training sample set.

$$N_{\text{ADD}} = \begin{cases} N_{\min} & N_{F\le 0.5} < N_{\min} \\ N_{\max} & N_{F\le 0.5} > N_{\max} \\ N_{F\le 0.5} & \text{otherwise} \end{cases} \tag{17}$$

where $N_{F\le 0.5}$ represents the number of learning function values $F\left(\boldsymbol{x}^{(j)}\right)\left(j=1,2,...,N_{\text{TOTAL}}\right)$ that less than or equal to 0.5. $N_{\min}$ and $N_{\max}$ represent minimum and maximum number of new training samples to be added in each iteration, respectively. The above expression restricts the number of new training samples to a value between $N_{\min}$ and $N_{\max}$, which balances the accuracy and efficiency of constructing graph-based neural networks to a certain extent. After the N_{ADD} new training samples are identified, the actual values of structure function corresponding to the N_{ADD} new training samples are computed by the Dijkstra algorithm [35].

The new training samples are incorporated into the training sample set, and the corresponding values of the structure function are added to the response set. This iterative process continues until the convergence criterion specified in Eq. (15) is satisfied. By selecting and including the most influential samples in the training sample set to update these graph-based neural networks, the prediction accuracy of these networks gradually improves. After the iterations, the graph-based neural network, using the original training sample set, can provide an accurate estimation of the survival signature $\hat{\Phi}\left(l_1,l_2,...,l_K\right)$. Finally, substituting the predicted survival signature into Eq. (3) yields the network reliability.

4. Estimation Procedure of Proposed Reliability Analysis Framework

The calculation flow-chart of the proposed reliability analysis framework is shown in Fig. 5. The specific estimation procedure is summarized as follows:

Step 1: Generate the total sample set, i.e., $\boldsymbol{x}^{(j)}=\left(\boldsymbol{x}^{1(j)},\boldsymbol{x}^{2(j)},...,\boldsymbol{x}^{K(j)}\right)\left(j=1,2,...,N_{\text{TOTAL}}\right)$, and the training sample set, i.e., $\boldsymbol{x}^{\text{Train}(j)}=\left(\boldsymbol{x}^{\text{Train}1(j)},\boldsymbol{x}^{\text{Train}2(j)},...,\boldsymbol{x}^{\text{Train}K(j)}\right)\left(j=1,2,...,N_{\text{TRAIN}}\right)$. Use the Dijkstra algorithm [35] to compute the corresponding values of structure function, i.e., $\phi\left(\boldsymbol{x}^{\text{Train}(j)}\right)\left(j=1,2,...,N_{\text{TRAIN}}\right)$, so to form the response set. Initialize N_{ADD} to N_{TRAIN}.

Step 2: Obtain additional N_{SET} sample sets $\boldsymbol{x}^{\text{SET}(i)(j)}=\left(\boldsymbol{x}^{\text{SET}1(i)(j)},\boldsymbol{x}^{\text{SET}2(i)(j)},...,\boldsymbol{x}^{\text{SET}K(i)(j)}\right)$ $\left(i=1,2,...,N_{\text{SET}};j=1,2,...,N_{\text{TRAIN}}\right)$ based on the current training sample set. Calculate the value of $N_{\text{SET-TRAIN}}$ based on Eq. (12) and identify the training and test samples for each additional sample set.

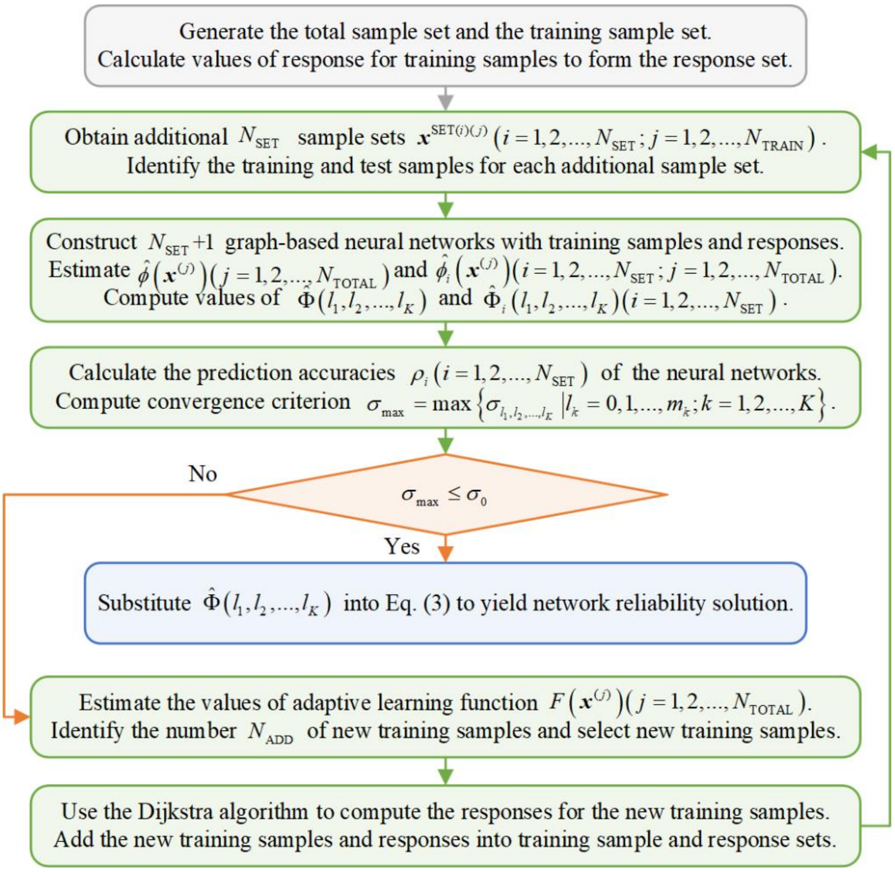

Generate the total sample set and the training sample set.
Calculate values of response for training samples to form the response set.
Obtain additional N_{SET} sample sets $\boldsymbol{x}^{\text{SET}(i)(j)}\left(i=1,2,\ldots,N_{\text{SET}};j=1,2,\ldots,N_{\text{TRAIN}}\right)$.
Identify the training and test samples for each additional sample set.
Construct $N_{\text{SET}}+1$ graph-based neural networks with training samples and responses.
Estimate $\hat{\phi}\left(\boldsymbol{x}^{(j)}\right)\left(j=1,2,\ldots,N_{\text{TOTAL}}\right)$ and $\hat{\phi}_i\left(\boldsymbol{x}^{(j)}\right)\left(i=1,2,\ldots,N_{\text{SET}};j=1,2,\ldots,N_{\text{TOTAL}}\right)$.
Compute values of $\hat{\Phi}\left(l_1,l_2,\ldots,l_K\right)$ and $\hat{\Phi}_i\left(l_1,l_2,\ldots,l_K\right)\left(i=1,2,\ldots,N_{\text{SET}}\right)$.
Calculate the prediction accuracies $\rho_i\left(i=1,2,\ldots,N_{\text{SET}}\right)$ of the neural networks.
Compute convergence criterion $\sigma_{\max}=\max\left\{\sigma_{l_1,l_2,\ldots,l_K}\middle|l_k=0,1,\ldots,m_k;k=1,2,\ldots,K\right\}$.
No
$\sigma_{\max}\le\sigma_0$
Yes
Substitute $\hat{\Phi}\left(l_1,l_2,\ldots,l_K\right)$ into Eq. (3) to yield network reliability solution.
Estimate the values of adaptive learning function $F\left(\boldsymbol{x}^{(j)}\right)\left(j=1,2,\ldots,N_{\text{TOTAL}}\right)$.
Identify the number N_{ADD} of new training samples and select new training samples.
Use the Dijkstra algorithm to compute the responses for the new training samples.
Add the new training samples and responses into training sample and response sets.

Fig. 5 The calculation flow-chart of the proposed reliability analysis framework

Step 3: Construct $N_{\text{SET}}+1$ graph-based neural networks based on the current training sample set, the additional N_{SET} sample sets, and the corresponding response set. Estimate the values of structure function $\hat{\phi}\left(\boldsymbol{x}^{(j)}\right)\left(j=1,2,\ldots,N_{\text{TOTAL}}\right)$ and $\hat{\phi}_i\left(\boldsymbol{x}^{(j)}\right)\left(i=1,2,\ldots,N_{\text{SET}};\,j=1,2,\ldots,N_{\text{TOTAL}}\right)$ based on the graph-based neural networks. Compute the corresponding values of survival signature $\hat{\Phi}\left(l_1,l_2,\ldots,l_K\right)$ and $\hat{\Phi}_i\left(l_1,l_2,\ldots,l_K\right)\left(i=1,2,\ldots,N_{\text{SET}}\right)$ for each case of $\left\{l_1,l_2,\ldots,l_K\right\}\left(l_k=0,1,\ldots,m_k;k=1,2,\ldots,K\right)$ according to the values of structure function and Eq. (1).

Step 4: Calculate the prediction accuracies, i.e., $\rho_i\left(i=1,2,\ldots,N_{\text{SET}}\right)$, of the additional N_{SET} graph-based neural networks based on Eq. (13), and exclude solutions associated with both the highest accuracy and lowest accuracy. Rewrite the remaining prediction solutions of the structure function and survival signature as $\hat{\phi}_i^*\left(\boldsymbol{x}^{(j)}\right)\left(i=1,2,\ldots,N_{\text{SET}}-2;\,j=1,2,\ldots,N_{\text{TOTAL}}\right)$ and $\hat{\Phi}_i^*\left(l_1,l_2,\ldots,l_K\right)$ $\left(i=1,2,\ldots,N_{\text{SET}}-2\right)$, respectively. Compute the convergence criterion based on Eqs. (21) and (22). If the convergence criterion specified in Eq. (15) is satisfied, go to Step 7; else, go to Step 5.

Step 5: Estimate the values of the adaptive learning function, i.e., $F\left(\boldsymbol{x}^{(j)}\right)\left(j=1,2,...,N_{\mathrm{TOTAL}}\right)$, for the total sample set $\boldsymbol{x}^{(j)}=\left(\boldsymbol{x}^{1(j)},\boldsymbol{x}^{2(j)},...,\boldsymbol{x}^{K(j)}\right)\left(j=1,2,...,N_{\mathrm{TOTAL}}\right)$, according to Eq. (16). Identify the number of new training samples, i.e., N_{ADD}, based on Eq. (17), and select N_{ADD} new training samples.

Step 6: Use the Dijkstra algorithm [35] to compute the corresponding values of structure function for the N_{ADD} new training samples. Add the N_{ADD} new training samples and the corresponding values of structure function into the training sample set and the response set, respectively. Let $N_{\mathrm{TRAIN}}=N_{\mathrm{TRAIN}}+N_{\mathrm{ADD}}$ and go to Step 2.

Step 7: Substitute the predicted survival signature $\hat{\Phi}\left(l_1,l_2,...,l_K\right)$ into Eq. (3) to yield the network reliability solution.

5. Applications

Several network reliability analysis problems are introduced to illustrate the effectiveness of the proposed graph-based neural network framework. For all these applications, the initial number of training samples is set to be $N_{\mathrm{TRAIN}}=2m+4n_L$, in which m and n_L represent the number of nodes and links in networks, respectively. The other parameters are set as follows: $N_{\mathrm{MCS}}=5000$, $N_{\mathrm{SET}}=10$, $N_{\min}=50$, $N_{\max}=2m+4n_L$, $\delta=0.9$, $\sigma_0=0.01$. These parameters are determined through preliminary testing, as there is no universally optimal choice for the parameters introduced in our proposed method. The optimal parameter values depend on the particular problem being addressed. The efficacy of the selected parameters is showcased through the applications presented in this section. The solutions of the survival signature and network reliability based on the complete enumeration technique (i.e., calculating the survival signature for all combinations of state vectors of components exactly) and the MCS method [26] are regarded as the reference actual and simulated solutions, respectively. The relative errors between the predicted solutions and the actual/simulated solutions are employed to illustrate the accuracy of the proposed graph-based neural network framework.

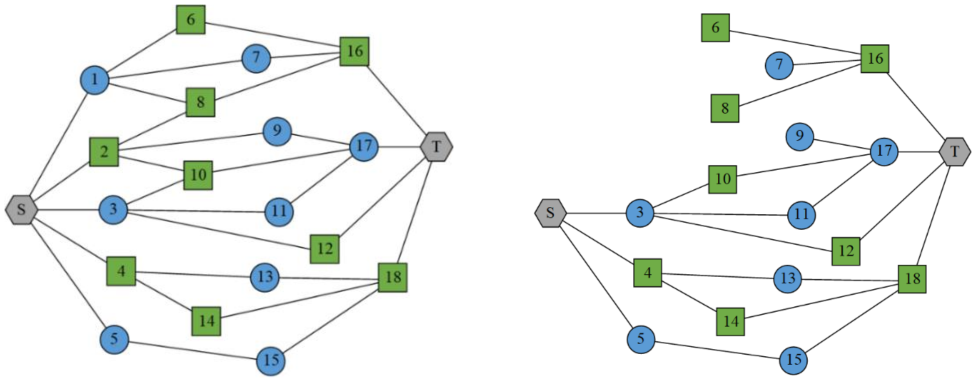
$$RE_{\Phi}\left(l_1,l_2,...,l_K\right)=\frac{\left|\Phi\left(l_1,l_2,...,l_K\right)-\hat{\Phi}\left(l_1,l_2,...,l_K\right)\right|}{\Phi\left(l_1,l_2,...,l_K\right)+\mathrm{e}^{-20}} \tag{18}$$

$$RE_R\left(t\right)=\frac{\left|R\left(t\right)-\hat{R}\left(t\right)\right|}{R\left(t\right)+\mathrm{e}^{-20}} \tag{19}$$

in which $\Phi\left(l_1,l_2,...,l_K\right)$ and $R\left(t\right)$ represent the solutions based on the complete enumeration technique or the MCS method, while $\hat{\Phi}\left(l_1,l_2,...,l_K\right)$ and $\hat{R}\left(t\right)$ are the solutions based on the established method. The small constant e^{-20} is used to prevent numerical singularities when $\Phi\left(l_1,l_2,...,l_K\right)$ or $R\left(t\right)$ equals to zero.

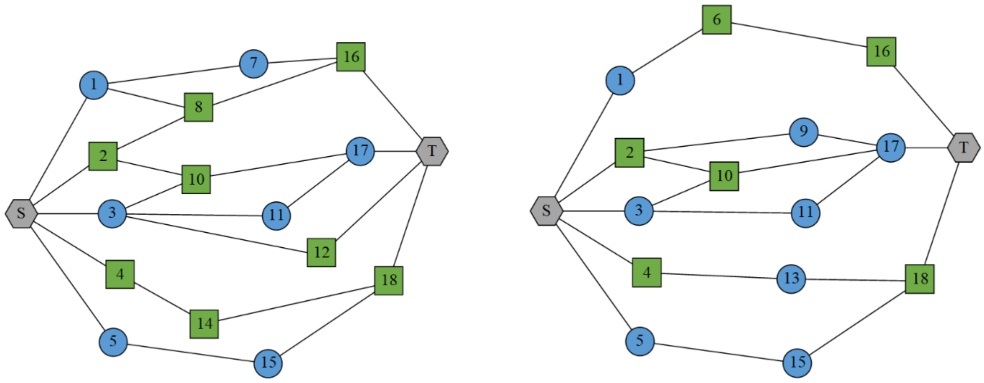
5.1 A Network with Multiple Variants

A network with 20 nodes and 30 links extracted from Ref. [37] is modified to test the effectiveness of the proposed graph-based neural network framework. The original topology structure of this network is shown in Fig. 6(a), and multiple variants of this network are depicted in Fig. 6(b)–(d). In the original network, there are two types of components. Components with circular icons represent type 1 characterized by the exponential distribution with a parameter of $\lambda=0.8$. On the other hand, components with square icons represent type 2 characterized by the exponential distribution with a parameter of $\lambda=1.5$. The multiple variants are generated through deleting several components and the related links of the original network. To be more specific, variant 1 is created by removing components 1 and 2 from the original network. Variant 2 is formed by eliminating components 6, 9, and 13 from the original network. Variant 3 is obtained by deleting components 7, 8, 12, and 14 from the original network. It is important to emphasize that these variants are artificially generated at random to assess the efficacy of the proposed method for analyzing network reliability in sub-networks with varying numbers of remaining components. In real-world scenarios, the identification of sub-networks is contingent upon actual conditions, for instance, in the context of power transmission networks, sub-networks may be identified based on the operational status of individual substations. Once the proposed graph-based neural network is trained and constructed based on the original network, it is directly employed to estimate the survival signature and network reliability of the multiple variants.



(a) Original network

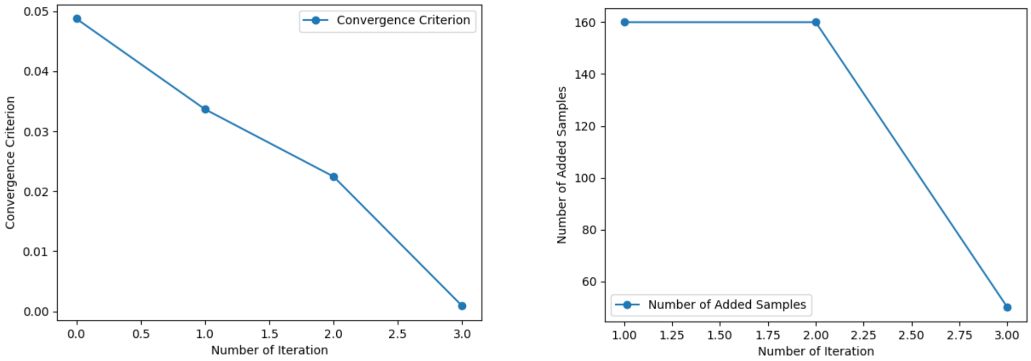
(b) Variant 1



(c) Variant 2 (d) Variant 3

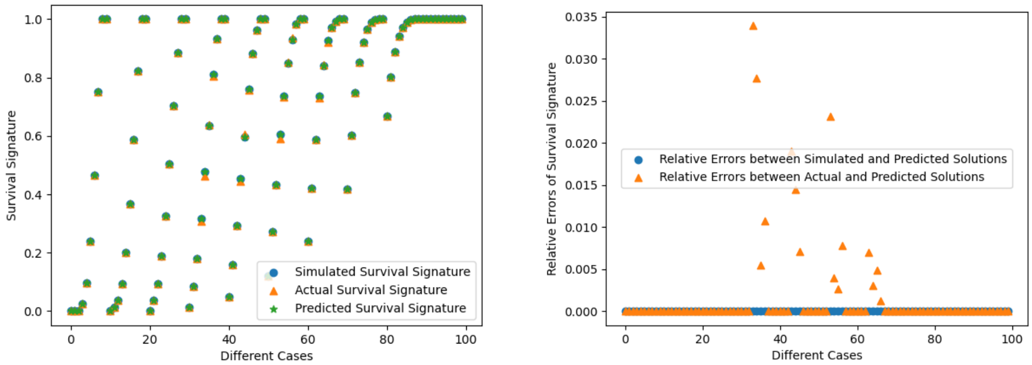
Fig. 6 The network with multiple variants

To establish the adaptive graph-based neural network for reliability analysis of the original network depicted in Fig. 6(a), 160 initial training samples are generated to construct multiple graph-based neural networks, as described in Section 3. In each iteration, the most influential samples are selected to update the graph-based neural networks. After three iterations, the convergence criterion was met, and the final predicted survival signature was utilized to estimate the network reliability. The convergence criterion and the number of added samples were plotted against the number of iterations in Fig. 7. From Fig. 7, it can be observed that in the first, second and third iterations, 160, 160 and 50 new training samples are identified, respectively, using the established algorithm. Hence, the total computational cost (i.e., representing the number of network structure function estimation by use the Dijkstra algorithm) of the proposed graph-based neural network method amounted to 530. Fig. 8 illustrates the survival signature and relative errors based on the proposed method for the original network. It's important to emphasize that different cases in Fig. 8 and following figures refer to various combinations of network states with varying numbers of surviving components. In the case of $\{l_1, l_2, ..., l_K\}$, its corresponding case number is $(m_2+1)(m_3+1)\cdots(m_K+1)l_1+(m_3+1)\cdots(m_K+1)l_2+\cdots+(m_{K-1}+1)(m_K+1)l_{K-2}+(m_K+1)l_{K-1}+l_K$ in this work. For instance, for the original network shown in Fig. 6 with $m_1 = m_2 = 9$, the case numbers are 23 and 56 in Fig. 8 for cases $\{l_1, l_2\} = \{2, 3\}$ and $\{l_1, l_2\} = \{5, 6\}$, respectively. Furthermore, Fig. 9 displays the network reliability and relative errors based on the proposed method for the original network. From Figs. 8 and 9, it is evident that the predicted survival signature and network reliability solutions can well match the actual solutions. The maximum relative errors between actual and predicted solutions for the survival signature and the network reliability are 0.03392 and 0.00514, respectively. Furthermore, the predicted survival signature and network reliability solutions accurately match the simulated solutions, yielding relative errors of zero. Given that epistemic uncertainty may be introduced when employing MCS methods to address the network reliability estimation, we ensure a fair comparison of accuracy by maintaining an identical total sample set for the proposed method and the MCS method. For the original network illustrated in Fig. 6(a), the MCS generates a total of 165725 combinations of network states. However, the proposed adaptive graph-based neural network reliability analysis framework leverages only 530 network state combinations along with their corresponding network responses to construct the graph-based neural network model. Remarkably, this model can accurately predict the network responses for all 165725 network state combinations. With the precise network response predictions, we can subsequently derive accurate survival signature and network reliability solutions. This remarkable achievement of zero relative errors in survival signature and network reliability solutions underscores the exceptional accuracy of the proposed adaptive graph-based neural network method. The solutions highlight the superior performance of the proposed adaptive graph-based neural network method.



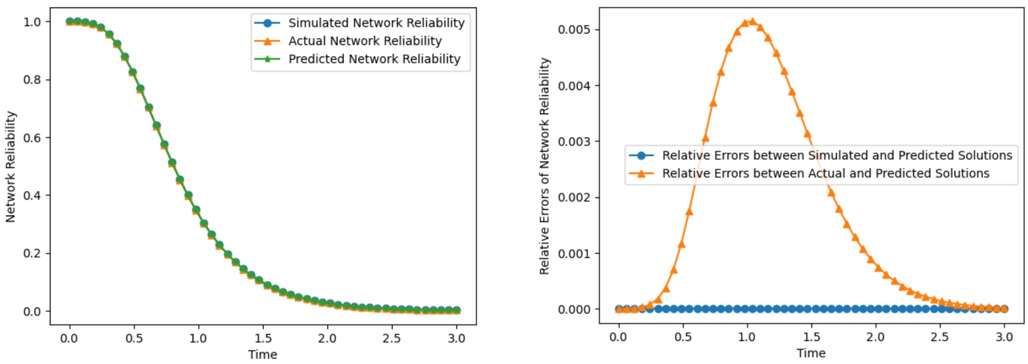
(a) Convergence criterion solutions (b) Number of added samples

Fig. 7 Convergence criterion and number of added samples for original network



(a) Survival signature solutions (b) Relative errors

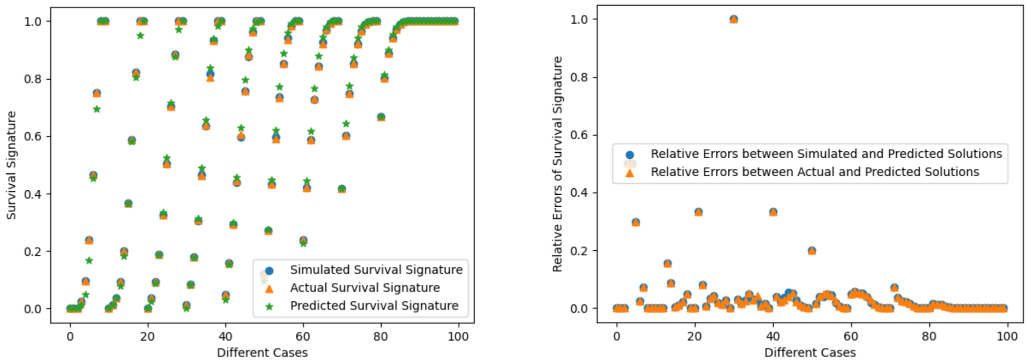
Fig. 8 Survival signature and relative errors based on the proposed method for original network



(a) Network reliability solutions (b) Relative errors

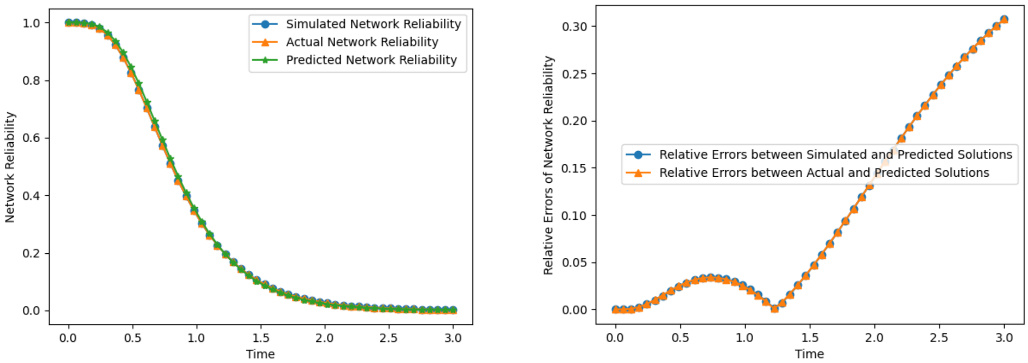
Fig. 9 Network reliability and relative errors based on the proposed method for original network

Additionally, to demonstrate the effectiveness of the established graph-based neural network and adaptive framework, we present a comparison of the survival signature and network reliability solutions using different machine learning techniques, namely ANN and GCNN. The ANN comprises three linear neural networks connected in series, with the Relu activation function employed between two consecutive linear neural networks. The GCNN shares the same network structure as the proposed graph-based neural network but differs in the aggregation methods used in the first two layers. In both ANN and GCNN, the same number (i.e., 530) of training samples as used in the proposed method are utilized. Figs. 10, 11, 12, and 13 display the survival signature solutions and relative errors, network reliability solutions and relative errors based on the ANN and the GCNN for the original network, respectively. From Figs. 10 and 11, it is evident that the ANN yields significant estimation errors for both the survival signature and network reliability solutions. Although the GCNN improves upon the performance of the ANN, it still exhibits considerable estimation errors compared to the proposed adaptive graph-based neural network method. These results underscore the high accuracy of the proposed method.



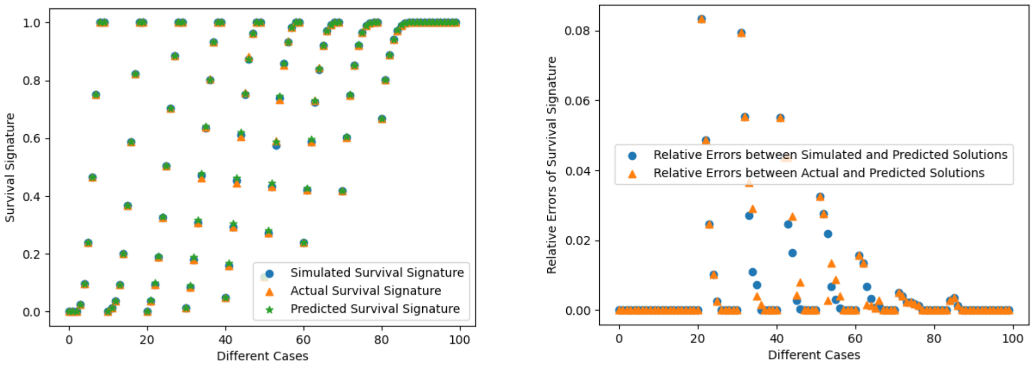
(a) Survival signature solutions (b) Relative errors

Fig. 10 Survival signature and relative errors based on the ANN for original network



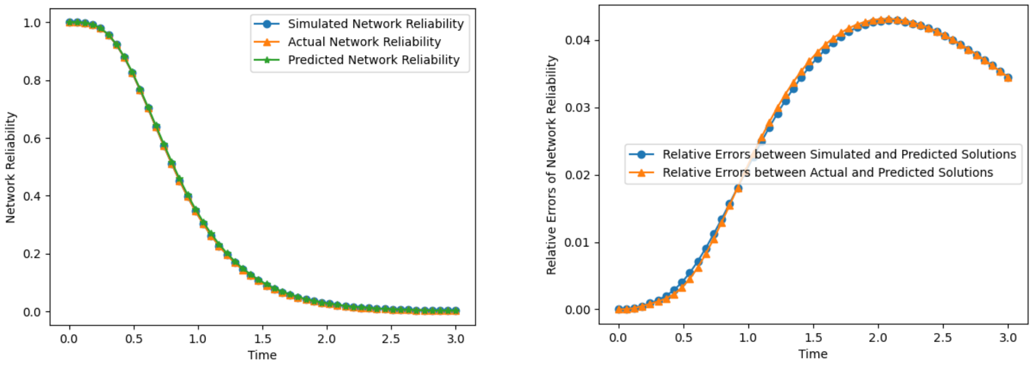
(a) Network reliability solutions (b) Relative errors

Fig. 11 Network reliability and relative errors based on the ANN for original network



(a) Survival signature solutions (b) Relative errors

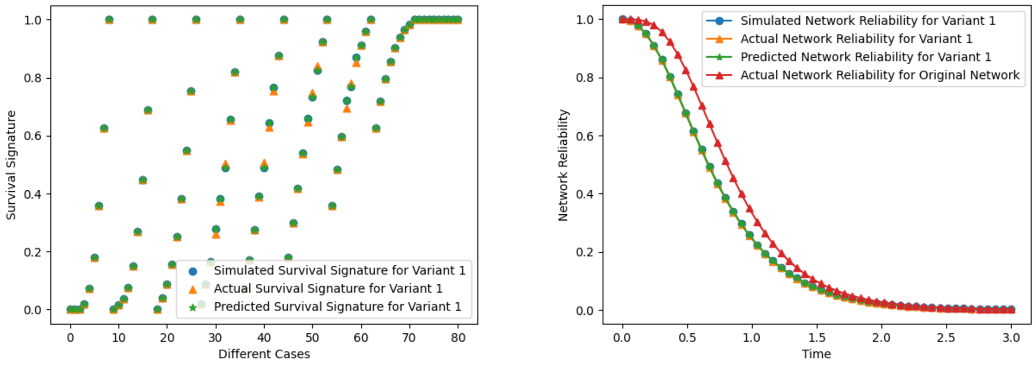
Fig. 12 Survival signature and relative errors based on the GCNN for original network



(a) Network reliability solutions (b) Relative errors

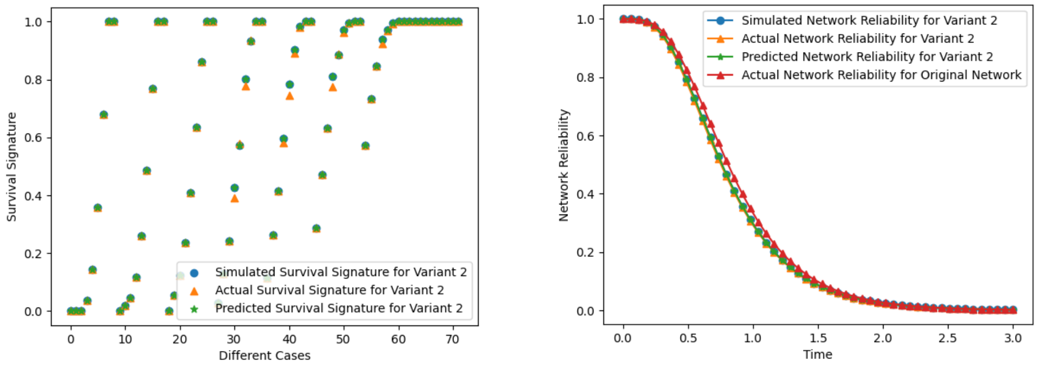
Fig. 13 Network reliability and relative errors based on the GCNN for original network

Once the proposed graph-based neural network is trained and constructed for the original network, it can be directly utilized to estimate the survival signature and network reliability of variant 1, variant 2, and variant 3. The survival signature and network reliability solutions for variant 1, variant 2, and variant 3 are illustrated in Figs. 14, 15, and 16, respectively. These figures demonstrate an accurate match (i.e., except for a few survival signature solutions) between the predicted survival signature and network reliability solutions with their respective reference solutions. Notably, the relative errors between simulated and predicted solutions for all cases are zero. For Variant 1, the maximum relative errors between the actual and predicted solutions for the survival signature and network reliability are 0.07419 and 0.00594, respectively. In the case of Variant 2, these errors increase slightly to 0.08607 for the survival signature and 0.01582 for network reliability. Variant 3 exhibits the lowest errors among all, with maximum relative errors of 0.07143 for the survival signature and an impressive 0.00350 for network reliability. This highlights the superiority of the proposed graph-based neural network, which enables the handling of different variants of the original network, and this is an ability typically unfeasible with non-machine learning methods. Moreover, an analysis of the network reliability solutions for these variants reveals that the overall reliability of variant 1 and variant 3 is lower than that of variant 2. This discrepancy arises from the relative importance of the components involved. Specifically, components 1 and 2, as well as components 7, 8, 12, and 14, have greater significance in maintaining network reliability compared to components 6, 9, and 13. The topology structure of the original network supports this observation. Since components 1 and 2 are directly connected to the source node, they can be assumed to be more critical than components not directly linked to the source or target nodes. Although components 7, 8, 12, and 14 are not directly connected to the source or target nodes, their combined effects generally outweigh those of the three components, i.e., 6, 9, and 13, at the same level.



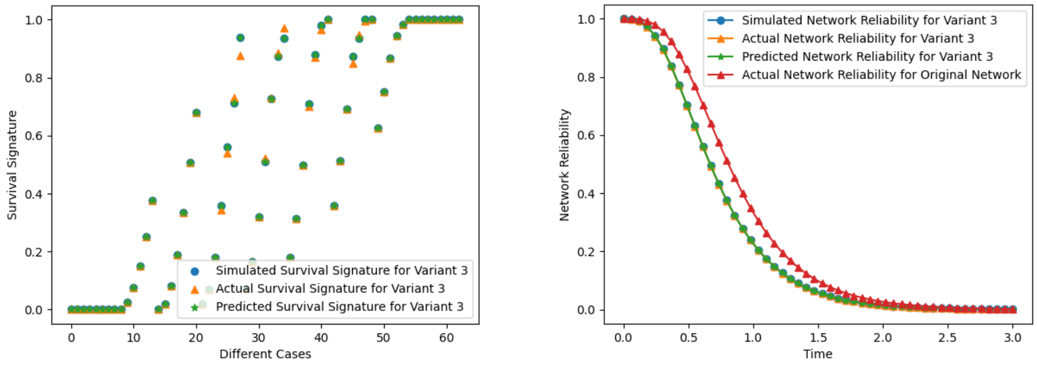
(a) Survival signature solutions (b) Network reliability solutions

Fig. 14 Survival signature and network reliability based on the proposed method for variant 1



(a) Survival signature solutions (b) Network reliability solutions

Fig. 15 Survival signature and network reliability based on the proposed method for variant 2

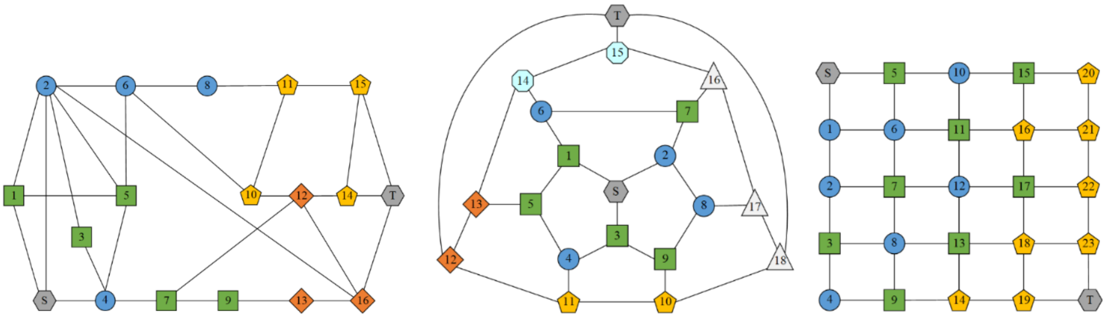


(a) Survival signature solutions (b) Network reliability solutions

Fig. 16 Survival signature and network reliability based on the proposed method for variant 3

5.2 Several Test Networks

Three different networks [38] are modified and shown in Fig. 17 to illustrate the effectiveness of the proposed graph-based neural network method. In all these networks, components are represented by circular icons (type 1), square icons (type 2), pentagon icons (type 3), diamond icons (type 4), octagon icons (type 5), and triangle icons (type 6). The distribution type and distribution parameters for the components in different networks are presented in Table 1.



(a) Network A (b) Network B (c) Network C

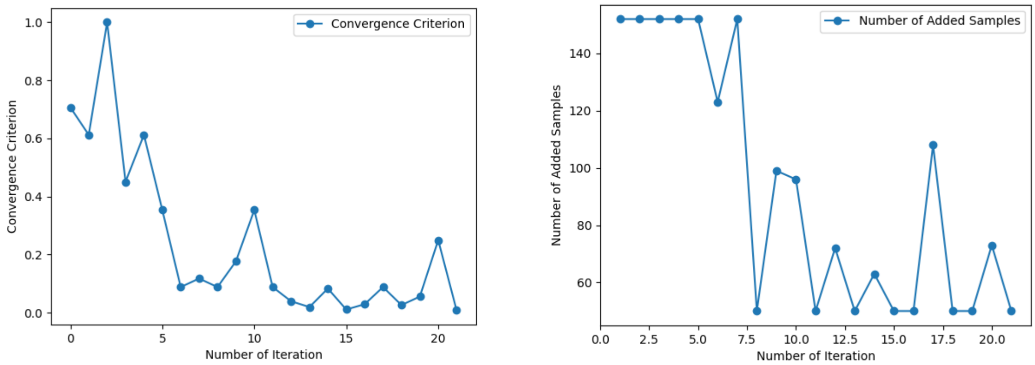
Fig. 17 Four networks with different topology structures

Table 1 Distribution type and distribution parameters for the components

Network name	Component type	Distribution type	Distribution parameters
Network A	1	Exponential	0.8
	2	Weibull	(1.7, 3.6)
	3	Lognormal	(1.5, 2.6)
	4	Gamma	(3.1, 1.5)
Network B	1	Exponential	0.8
	2	Exponential	1.5
	3	Weibull	(1.5, 3.2)
	4	Weibull	(1.8, 3.5)
	5	Lognormal	(1.5, 2.5)
	6	Gamma	(3.0, 1.2)
Network C	1	Weibull	(1.5, 3.0)
	2	Lognormal	(1.8, 2.7)
	3	Gamma	(3.2, 1.5)

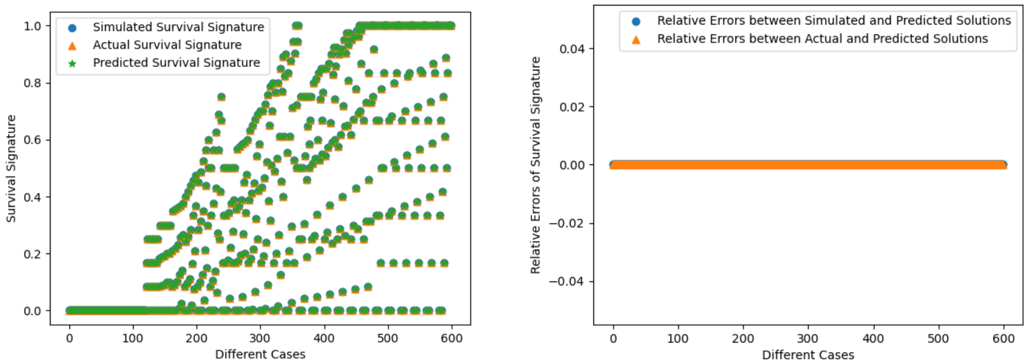
Note: The Lognormal distribution parameters consist of mean and standard deviation parameters, while the Weibull and Gamma distributions are characterized by scale and shape parameters.

The proposed graph-based neural network framework, the ANN and the GCNN are utilized to calculate the survival signature and network reliability of the three different networks. In both ANN and GCNN, the same number of training samples as used in the proposed method are utilized. The convergence criterion and number of added samples, the survival signature and relative errors, as well as the network reliability and relative errors for the three different networks, are presented in Figs. 18-32, respectively. These figures indicate that the proposed adaptive framework exhibits favorable convergence characteristics, with the maximum number of iterations shown in Fig. 23 for network B being less than 30. Despite having a smaller number of nodes and links compared to network C, network B comprises more component types, which generally necessitates a greater number of iterations to achieve an accurate estimation of the survival signature. Regarding the estimation of the survival signature, it is observed that network A and network B, depicted in Figs. 19 and 24, respectively, have zero-relative errors between simulated and predicted solutions for all cases, and zero-relative errors or close to zero-relative errors between actual and predicted solutions for those cases. For the survival signature solutions shown in Fig. 29 for network C, the maximum relative errors between simulated and predicted solutions as well as actual and predicted solutions are 0.02913 and 1.35199, respectively. While the maximum relative error between the actual and predicted solutions is substantial, with actual and predicted survival signature solutions being 0.00051 and 0.00120, respectively, its impact on network reliability estimation is relatively limited. This can be substantiated by the high accuracy of reliability estimation depicted in Fig. 30. The reason for this lies in the fact that in scenarios where the survival signature is small, its influence on the ultimate network reliability estimation is diminished, as clearly illustrated by Eq. (3). Furthermore, at the case of maximum relative error, the simulated survival signature is 0.00120, which is accurately equal to the predicted one. This illustrates that the constructed graph-based neural network can accurately predict network responses for the network states corresponding to this survival signature generated based on MCS. The large maximum relative error between the actual and predicted solutions is due to insufficient sample size of MCS for this case, and it can be decreased by increasing the corresponding sample size. For network C, the current solutions are available since the predicted network reliability solutions are accurate enough. Furthermore, the relative errors of estimated solutions including the survival signature and network reliability based on ANN and GCNN, as depicted in Figures 21, 22, 26, 27, 31, and 32, indicate that both ANN and GCNN exhibit significant inaccuracies.



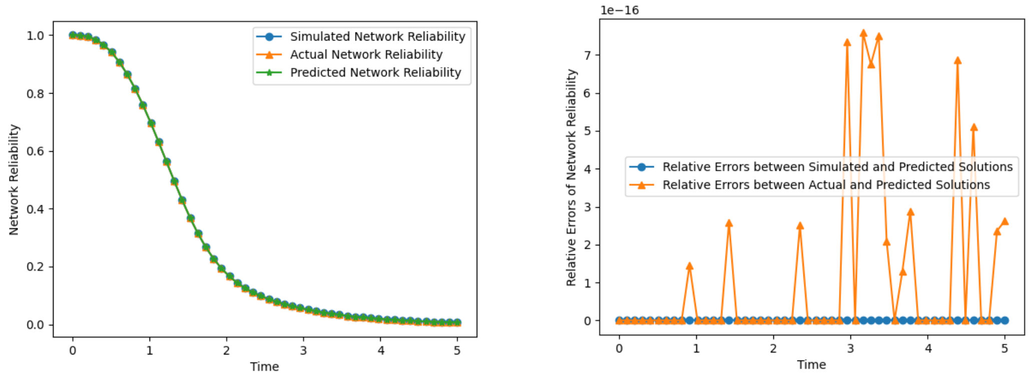
(a) Convergence criterion solutions (b) Number of added samples

Fig. 18 Convergence criterion and number of added samples for network A



(a) Survival signature solutions (b) Relative errors

Fig. 19 Survival signature and relative errors based on the proposed method for network A



(a) Network reliability solutions (b) Relative errors

Fig. 20 Network reliability and relative errors based on the proposed method for network A

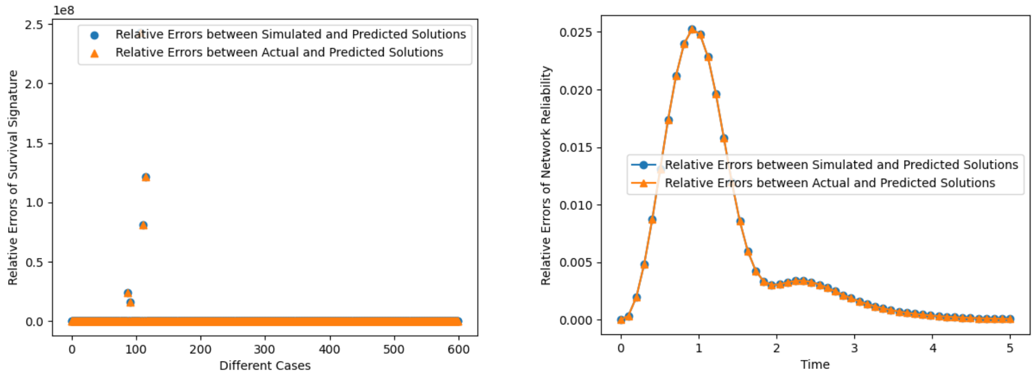


Fig. 21 Relative errors of estimated solutions based on the ANN for network A

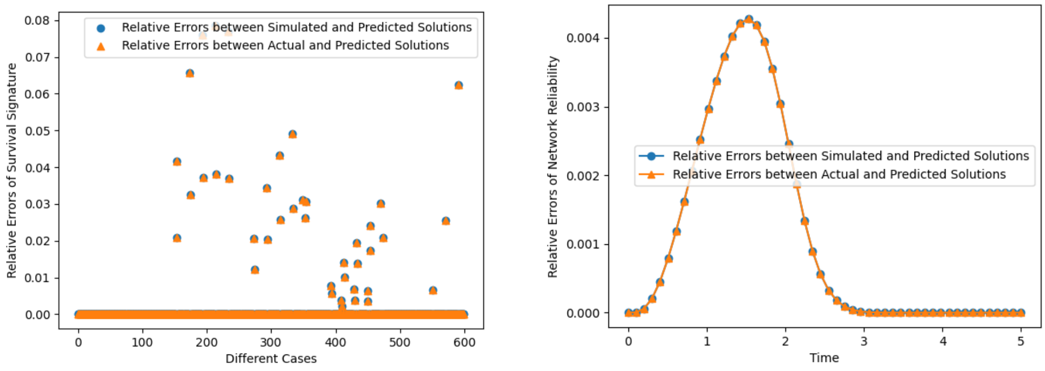
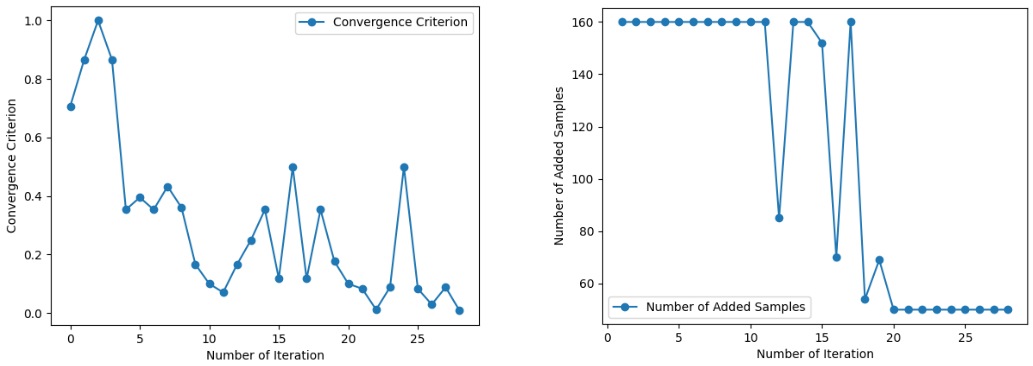


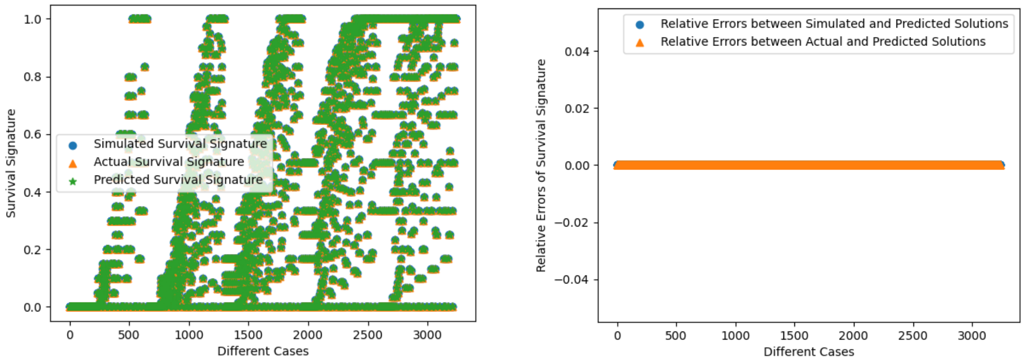
Fig. 22 Relative errors of estimated solutions based on the GCNN for network A

The network reliability solutions presented in Figs. 20, 25, and 30 demonstrate a close match between the predicted, simulated and actual network reliability solutions. The maximum relative error, as depicted in Fig. 30, is 0.00340 between the actual and predicted solutions, highlighting the high accuracy of the proposed method in estimating network reliability. It is worth noting that the computational accuracy of the graph-based neural network can be further enhanced by setting a more stringent convergence threshold value. In our experiments, the total computational costs of the proposed graph-based neural network for network A, network B, and network C are 2098, 3280, and 3780, respectively. In contrast, the total computational costs of the MCS are 65519, 261156, and 1436637, respectively, excluding the scenario where the number of working components is fewer than the shortest path in the network, resulting in certain network failure. The total computational costs of the complete enumeration technique are 65536, 262144, and 8388608, respectively. The proportion of the calculation amount for the proposed method to the total calculation amount of the complete enumeration technique is approximately 3.2013%, 1.2512%, 0.2499%, and 0.0451%, respectively. These findings underscore the high computational efficiency of the proposed graph-based neural network framework.



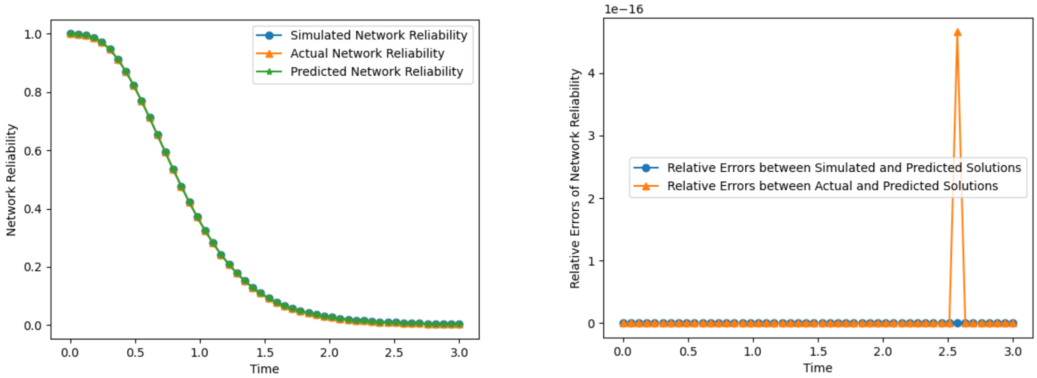
(a) Convergence criterion solutions (b) Number of added samples

Fig. 23 Convergence criterion and number of added samples for network B



(a) Survival signature solutions (b) Relative errors

Fig. 24 Survival signature and relative errors based on the proposed method for network B



(a) Network reliability solutions (b) Relative errors

Fig. 25 Network reliability and relative errors based on the proposed method for network B

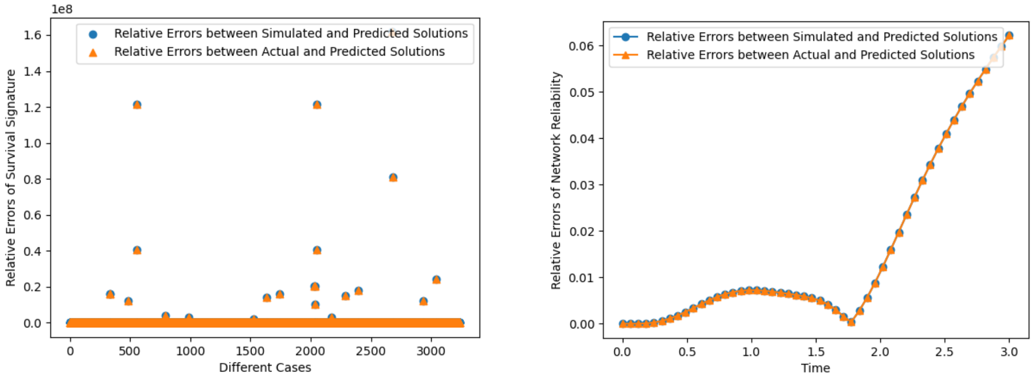


Fig. 26 Relative errors of estimated solutions based on the ANN for network B

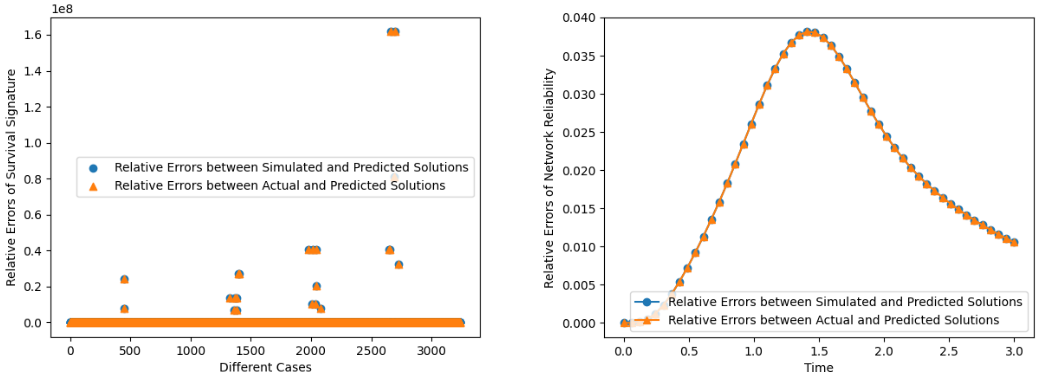
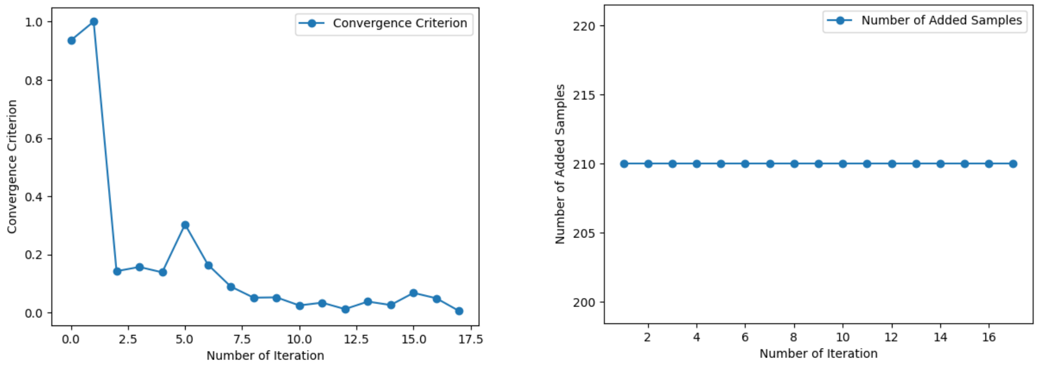
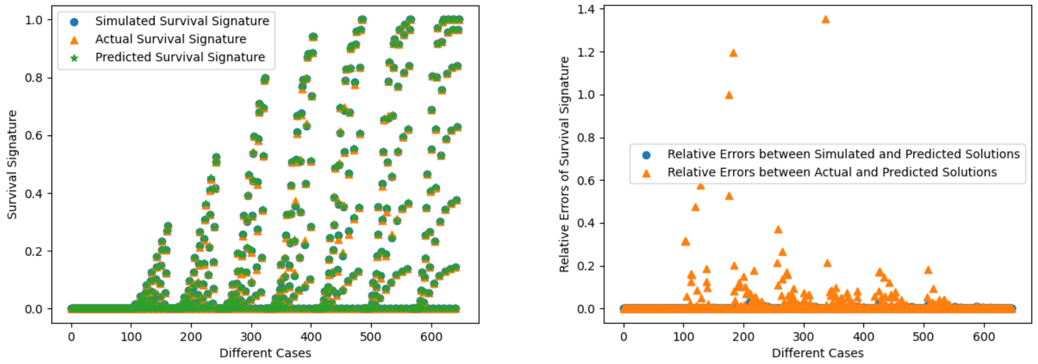


Fig. 27 Relative errors of estimated solutions based on the GCNN for network B



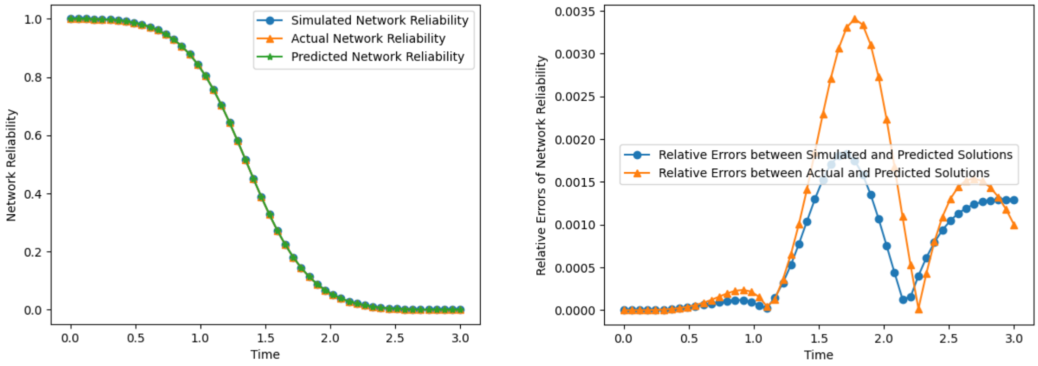
(a) Convergence criterion solutions (b) Number of added samples

Fig. 28 Convergence criterion and number of added samples for network C



(a) Survival signature solutions (b) Relative errors

Fig. 29 Survival signature and relative errors based on the proposed method for network C



(a) Network reliability solutions (b) Relative errors

Fig. 30 Network reliability and relative errors based on the proposed method for network C

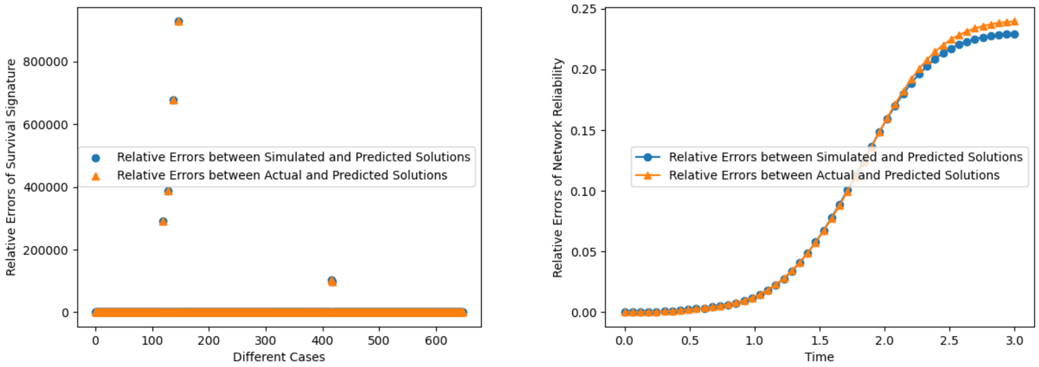


Fig. 31 Relative errors of estimated solutions based on the ANN for network C

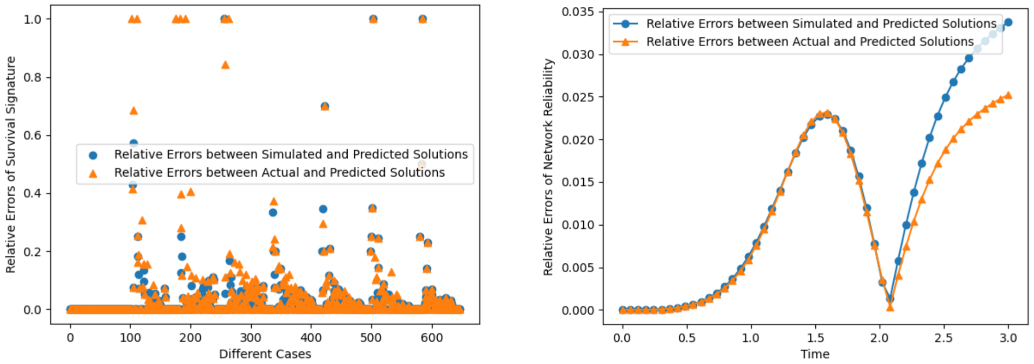


Fig. 32 Relative errors of estimated solutions based on the GCNN for network C

6. Conclusions

In this study, a novel graph-based neural network framework is proposed to effectively estimate survival signatures and network reliability. The graph-based neural network begins by utilizing a developed strategy to aggregate feature information from neighboring nodes, thereby effectively integrating the response flow characteristics of the networks. Following this, the HGNN is employed to further aggregate feature information from both neighboring nodes and the node itself. This technique facilitates the extraction of network topology structural information, thereby enhancing the algorithm's comprehension of networks. Additionally, an adaptive framework is established to improve prediction accuracy. Compared to traditional machine learning-based frameworks, the proposed graph-based neural network framework integrates response flow characteristics and incorporates network topology structure information, resulting in highly accurate estimates of network reliability.

The effectiveness of the proposed method is demonstrated by applying it to various networks. The obtained solutions have shown that the proposed method not only provides accurate network reliability estimates but also exhibits favorable convergence characteristics. Moreover, our findings indicate that once the proposed graph-based neural network is trained and constructed on the original network, it performs well in estimating network reliability for different sub-networks, a capability not achievable with traditional non-machine learning methods. It's important to note that the graph-based neural network developed in this study is tailored for two-terminal networks. For more complex configurations, such as k-terminal networks or all-terminal networks, it may experience reduced efficiency. Addressing the challenges posed by k-terminal networks or all-terminal networks will be a key focus of our future research.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant 52205252), and the National Natural Science Foundation of Sichuan Province (Grant 2023NSFSC0876). The first author would also thanks for the support of the Alexander von Humboldt Foundation of Germany.

Conflict of interest statement

The authors declare that they have no conflict of interest.

References

[1] Zuev K M, Wu S, Beck J L. General network reliability problem and its efficient solution by subset simulation. Probabilistic Engineering Mechanics, 2015, 40: 25-35.

[2] Gaur V, Yadav O P, Soni G, et al. A literature review on network reliability analysis and its engineering applications. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 2021, 235(2): 167-181.

[3] Abraham J A. An improved algorithm for network reliability. IEEE Transactions on Reliability, 1979, 28(1): 58-61.

[4] Goyal N K. Network reliability evaluation: a new modeling approach. International Conference on Reliability and Safety Engineering. 2005: 473-488.

[5] Liu T, Bai G, Tao J, et al. An improved bounding algorithm for approximating multistate network reliability based on state-space decomposition method. Reliability Engineering & System Safety, 2021, 210: 107500.

[6] Lee S M, Park D H. An efficient method for evaluating network-reliability with variable link-capacities. IEEE Transactions on Reliability, 2001, 50(4): 374-379.

[7] Yeh W C. A quick BAT for evaluating the reliability of binary-state networks. Reliability Engineering & System Safety, 2021, 216: 107917.

[8] Yeh W C. Novel binary-addition tree algorithm (BAT) for binary-state network reliability problem. Reliability Engineering & System Safety, 2021, 208: 107448.

[9] Zuo M J, Tian Z, Huang H Z. An efficient method for reliability evaluation of multistate networks given all minimal path vectors. IIE transactions, 2007, 39(8): 811-817.

[10]Lin J S, Jane C C, Yuan J. On reliability evaluation of a capacitated-flow network in terms of minimal pathsets. Networks, 1995, 25(3): 131-138.

[11]Ball M O. Computational complexity of network reliability analysis: an overview. IEEE Transactions on Reliability, 1986, 35(3): 230-239.

[12]Yeh W C, Lin Y C, Chung Y Y, et al. A particle swarm optimization approach based on Monte Carlo simulation for solving the complex network reliability problem. IEEE Transactions on Reliability, 2010, 59(1): 212-221.

[13]Ramirez-Marquez J E. Stochastic network interdiction optimization via capacitated network reliability modeling and probabilistic solution discovery. Reliability Engineering & System Safety, 2009, 94(5): 913-921.

[14]Chang P C. MC-based simulation approach for two-terminal multi-state network reliability evaluation without knowing d-MCs. Reliability Engineering & System Safety, 2022, 220: 108289.

[15]Coolen F P A, Coolen-Maturi T. Generalizing the signature to systems with multiple types of components. Complex systems and dependability. Springer Berlin Heidelberg, 2012: 115-130.

[16]Boland P J, Samaniego F J. The signature of a coherent system and its applications in reliability. Mathematical reliability: An expository perspective, 2004: 3-30.

[17]Qin J, Coolen F P A. Survival signature for reliability evaluation of a multi-state system with multi-state components. Reliability Engineering & System Safety, 2022, 218: 108129.

[18]Feng G, Patelli E, Beer M, et al. Imprecise system reliability and component importance based on survival signature. Reliability Engineering & System Safety, 2016, 150: 116-125.

[19]Aslett L J M, Coolen F P A, Wilson S P. Bayesian inference for reliability of systems and networks using the survival signature. Risk Analysis, 2015, 35(9): 1640-1651.

[20]Patelli E, Feng G, Coolen F P A, et al. Simulation methods for system reliability using the survival signature. Reliability Engineering & System Safety, 2017, 167: 327-337.

[21]Huang X, Aslett L J M, Coolen F P A. Reliability analysis of general phased mission systems with a new survival signature. Reliability Engineering & System Safety, 2019, 189: 416-422.

[22]Salomon J, Winnewisser N, Wei P, et al. Efficient reliability analysis of complex systems in consideration of imprecision. Reliability Engineering & System Safety, 2021, 216: 107972.

[23]Liu Y, Shi Y, Bai X, et al. Stress–strength reliability analysis of multi-state system based on generalized survival signature. Journal of Computational and Applied Mathematics, 2018, 342: 274-291.

[24]Reed S. An efficient algorithm for exact computation of system and survival signatures using binary decision diagrams. Reliability Engineering & System Safety, 2017, 165: 257-267.

[25]Yi H, Cui L, Balakrishnan N. Computation of survival signatures for multi-state consecutive-k systems. Reliability Engineering & System Safety, 2021, 208: 107429.

[26]Behrensdorf J, Regenhardt T E, Broggi M, et al. Numerically efficient computation of the survival signature for the reliability analysis of large networks. Reliability Engineering & System Safety, 2021, 216: 107935.

[27]Nagulapati V M, Lee H, Jung D W, et al. Capacity estimation of batteries: Influence of training dataset size and diversity on data driven prognostic models. Reliability Engineering & System Safety, 2021, 216: 108048.

[28]Xu K, Xie M, Tang L C, et al. Application of neural networks in forecasting engine systems reliability. Applied Soft Computing, 2003, 2(4): 255-268.

[29]Jia W, Tian Y, Luo R, et al. Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot. Computers and Electronics in Agriculture, 2020, 172: 105380.

[30]Zhang P, Wang X, Zhang W, et al. Learning spatial–spectral–temporal EEG features with recurrent 3D convolutional neural networks for cross-task mental workload assessment. IEEE Transactions on neural systems and rehabilitation engineering, 2018, 27(1): 31-42.

[31]Jin L, Yan J, Du X, et al. RNN for solving time-variant generalized Sylvester equation with applications to robots and acoustic source localization. IEEE Transactions on Industrial Informatics, 2020, 16(10): 6359-6369.

[32]Chen J, Jing H, Chang Y, et al. Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process. Reliability Engineering & System Safety, 2019, 185: 372-382.

[33]Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[34]Morris C, Ritzert M, Fey M, et al. Weisfeiler and leman go neural: higher-order graph neural networks. Proceedings of the AAAI conference on artificial intelligence. 2019, 33(01): 4602-4609.

[35]Dijkstra E W. A note on two problems in connexion with graphs. Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022: 287-290.

[36]Kingma D P, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[37]Dai Y, Poh K. Solving the network interdiction problem with genetic algorithms. Proceedings of the fourth Asia-Pacific conference on industrial engineering and management system, Taipei. 2002: 18-20.

[38]Ramirez-Marquez J E. Deterministic network interdiction optimization via an evolutionary approach. Reliability Engineering & System Safety, 2009, 94(2): 568-576.