# Generative Design for Resilience of Interdependent Network Systems

**Jiaxin Wu**
Department of Industrial and Enterprise Systems Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: jiaxinw3@illinois.edu

**Pingfeng Wang**[1]
Mem. ASME
Associate Professor
Department of Industrial and Enterprise Systems Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: pingfeng@illinois.edu

*Interconnected complex systems usually undergo disruptions due to internal uncertainties and external negative impacts such as those caused by harsh operating environments or regional natural disaster events. To maintain the operation of interconnected network systems under both internal and external challenges, design for resilience research has been conducted from both enhancing the reliability of the system through better designs and improving the failure recovery capabilities. As for enhancing the designs, challenges have arisen for designing a robust system due to the increasing scale of modern systems and the complicated underlying physical constraints. To tackle these challenges and design a resilient system efficiently, this study presents a generative design method that utilizes graph learning algorithms. The generative design framework contains a performance estimator and a candidate design generator. The generator can intelligently mine good properties from existing systems and output new designs that meet predefined performance criteria while the estimator can efficiently predict the performance of the generated design for a fast iterative learning process. Case studies results based on synthetic supply chain networks and power systems from the IEEE dataset have illustrated the applicability of the developed method for designing resilient interdependent network systems.* [DOI: 10.1115/1.4056078]

*Keywords: data-driven design, design optimization, design theory and methodology, generative design, machine learning*

## 1 Introduction

With the increases in both scale and complexity, interdependent critical infrastructures (ICIs), such as power systems or transportation networks, become more vulnerable to disruptive events. And natural disasters impose great negative impacts on the system reliability, e.g., a winter storm and associated cold waves led to a large-scale blackout event that affected more than five million people in Feb. 2021 in Texas. Such vulnerability, therefore, drives the research efforts that could lead to robust and resilient ICIs. For instance, how to efficiently design a large-scale system that can resist potential external disruptions or how can the decision-maker evaluate the uncertain dynamic behavior of the ICI undergoing different disruptive events? To quantify the system's performance during disruption or to comprehend the system's capability toward uncertain disruptive scenarios, researchers have adopted the term "resilience" from the ecology field [1]. Different from the terminology of system reliability, in which the time-dependent degraded system performance and the possibility of failure are studied, the resilience metric is utilized to complement the analysis of real-time system behavior. Based on the U.S. Department of Defense report, a resilient ICI should not only withstand the impacts of disruptive events but also need to acquire the capability of self-healing from damages [2]. Thus, to realize a resilient ICI through design or operational management strategies, the stakeholders need to tackle challenges in three folds: (1) how should the system proactively detect the occurrence of abnormalities with the possible external or internal disruptions; (2) how large is the bandwidth of the ICI for withstanding adversarial impacts; and (3) how quick the ICI can self-recover to its nominal state [3].

Motivated by the challenges from those three aspects, different frameworks have been proposed to help the ICI establish self-healing capability after system disruptions, therefore, achieving failure resilience. Here, we categorize the research efforts about engineering resilience based on the temporal stages of the proposed frameworks, i.e., before and after the disruptions. During the post-disruption stage, several real-time operational frameworks have been proposed to guide how the system should behave after disruptive events. For instance, researchers try to attain a resilient operational framework by scheduling optimal repair tasks under uncertainties [4–6] as well as repair resources [7], forming self-sustainable microgrids [8–10] and guided recovery through control strategies [11,12]. All aforementioned studies focus on solving the optimal decisions of how to utilize the existing resources or back-ups to recover the ICI, on the promise that setup a contingency plan such as network reconfiguration beforehand. In other words, during the post-disruption stage, the self-recovery capability is realized in two steps: appropriate emergency response, e.g., system reconfiguration, followed by performing optimal restorations.

Although comprehensive post-disruption frameworks have been proposed to guide how a system should behave after failure events, methodologies for proactively improving the system resilience or quantifying the resilience level of the ICIs are still unknown. And without an appropriate pre-disruption design/planning framework, the stakeholders need to frequently apply the aforementioned contingency plans to ensure system nominal performance, which leads to a more significant cost for the resilience enhancement. As a result, it is required to study suitable strategies to ensure system resilience even during the planning stage and to ease the necessity of adopting post-disruption control efforts.

To improve the system resilience and thus achieve system operations with better quality, however, current engineering resilience design research has been focusing on proposing ad hoc models. For instance, researchers have proposed different system modeling and analysis methods to quantify and analyze the resilience level of complex engineering systems, e.g., power distribution systems and

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received June 13, 2022; final manuscript received October 8, 2022; published online November 17, 2022. Assoc. Editor: Xu Han.



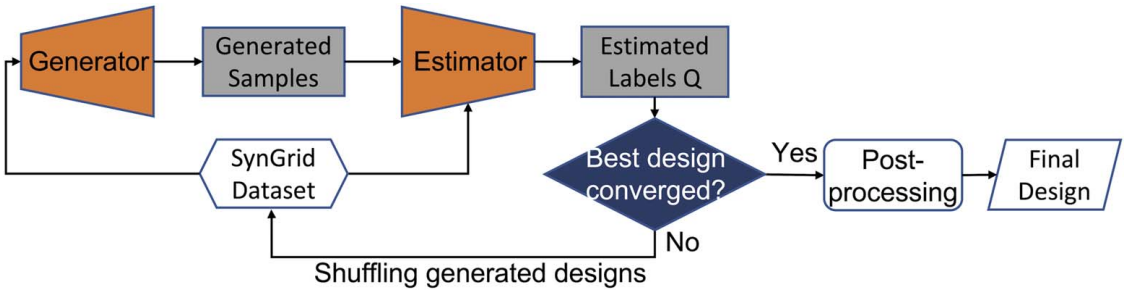



**Fig. 1 Flowchart of the proposed generative design framework for ICIs: there are two major components, the design generator and estimator. And both components are formulated based on graph learning algorithms.**

supply chain networks, undergoing a disruptive event during the pre-disruption stage [13,14]. Besides, various approaches based on mathematical programming models are proposed to solve the best design strategies for ICIs under different operating scenarios [15,16]. Furthermore, probabilistic approaches such as the Bayesian network have been adopted to analyze and quantify the overall system resilience with the presence of disruptions [17,18]. All aforementioned researches have demonstrated their capabilities of quantifying the system resilience and thus finding the best design of the system, e.g., expanding the existing ICI or solving the best layout for a new system. Yet existing methods have their drawbacks: either the model is ad hoc so that the generalization is not straightforward or the system models are simplified to have a tractable problem. For example, many optimization-based approaches have assumptions for the underlining physical constraints to derive a solvable model for optimum. Or probabilistic approaches only consider systems with around 30 components to simplify the solving process. As a result, those drawbacks limit the applicability of the existing methods on realistic large-scale ICIs.

To further complicate the pre-disruption design problems for ICIs, the system usually consists of heterogeneous components for satisfying the various needs in the same system. However, these different components add more constraints for solving the optimal design and make the decision-maker use ad hoc models to evaluate different online scenarios. Moreover, ICIs are usually discrete systems with up to tens of thousands of nodes and edges. Such a highly combinatorial system is nearly impossible to find the best design, without the help of intelligent methods. Thus, in this study, we focus on proposing a robust and intelligent generative framework to realize the design for ICIs that optimizes the resilience level.

The proposed generative design framework for ICIs is a model-free, data-driven method. It utilizes the graph learning algorithm to reconstruct candidate designs based on the input dataset of real-world ICIs. As a result, there is no need to make any assumptions about the ICIs to simplify the system operation constraints. Also, the optimal design does not rely on any specific mathematical model of the ICI. That is, the generative approach can learn the insights from the existing ICIs directly. And throughout the iterative training process, the generated candidates can be optimized towards predefined performance criteria, i.e., the system resilience level. As shown in Fig. 1, the framework includes two major components—the generator and the estimator. The design generator is a variational autoencoder (VAE) that directly outputs feasible designs for the ICI, while the estimator is pretrained to predict the performance of the candidate design efficiently. And an iterative process that blends the generated designs into the training dataset closes the gap of "target" driven generation to bias the generator towards outputting samples with high resilience. Finally, adequate post-processing step including more expensive post-disruption simulations pinpoints the best system design.

The contributions of this work are in two aspects: (1) to the authors' best knowledge, there is no "target" (resilience) driven generative algorithm for interconnected systems yet. In the machine learning community, various algorithms, such as graph recurrent neural network [19], graph recurrent attention network [20], and graph variational autoencoder (GVAE) [21], have been proposed to generate synthetic graph/discrete structures. And with the help of the graph neural network model, an estimator can be constructed to determine the resilience level of an ICI automatically. Combined with the generative part, the estimator further guides the generation process. (2) It is the first time adopting graph generation algorithms to real-world network systems with rich physical information. A physical system, for instance, the power grid, has much more complicated operational constraints than a social network or molecular structure. Those synthetic systems are the main applications of the generative algorithms proposed in the machine learning field. Thus, how to properly address the physical constraints and information when applying the graph algorithm to ICIs remains as a crucial challenge.

The rest of the paper is organized as follows: Sec. 3 explains the modeling of the design generator, and Sec. 4 presents the formulation of the design estimator in detail. Section 5 discusses the post-processing simulation step for re-evaluating the designs with more expensive metrics. A case study based on the power system design is used to illustrate the applicability of the proposed framework in Sec. 6. Section 7 concludes the study with brief discussions on the effectiveness of the developed design methodology.

## 2 Modeling of ICIs and System Resilience

In this study, the ICI is modeled as graphs to indicate the interdependence and inherent network structure of the system. Taking the commodity distribution system as an example, the warehouse, transportation hubs, or the final destinations can be modeled as nodes $\{i|i \in \mathbf{V}\}$. And the distribution paths are the edges $\{ij|ij \in \mathbf{E}\}$ between different nodes. Thus, the overall ICI is denoted as a graph $\mathcal{G} := (\mathbf{V}, \mathbf{E})$ as shown in Fig. 2. Other than the topological information illustrated in the figure, physical systems usually carry much more signals. Taking the power grid as an example, each node of the graph can have a specific level of operating voltages, frequencies, and magnitudes of generations/loads, while the edges can have physical constraints for capacities, distance, connection types, etc. Thus how to generatively design an ICI satisfying practical operating conditions becomes the major challenge.

On the other hand, to measure the system resilience, a typical resilience curve (real performance curve after disruptions) with four states is illustrated in Fig. 3. Note that the system performance curves could be different due to different strategies during the recovery process. Based on the resilience curve, this study defines the resilience level with respect to changes in the system performance after the disturbance. It can be measured by comparing the resilience curve with the nominal system performance curve. In other words, the resilience metric is derived from the ratio of the area under the resilience curve to the area under the nominal

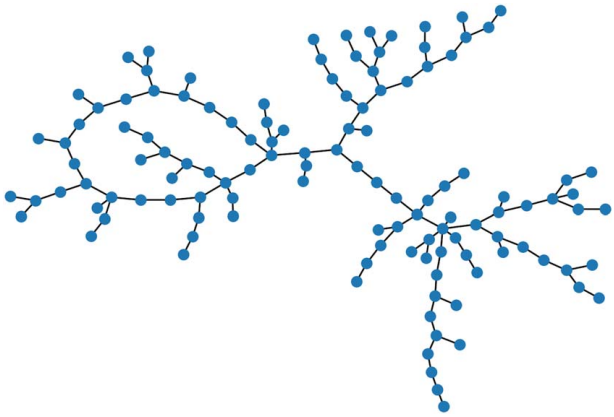


**Fig. 2 The graphical representation of a power distribution system consists of 123 buses and 123 distribution lines**

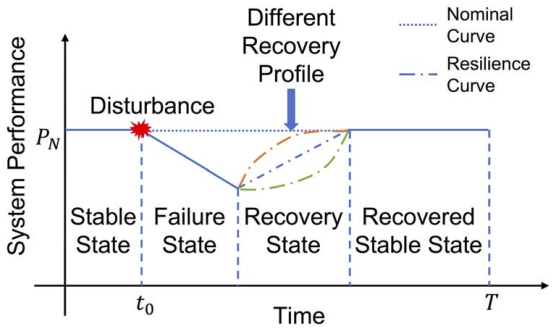



**Fig. 3 The resilience curve and four states in an ICI after with the disruptive event**

performance curve. The specific formulation is

$$\Phi = \frac{\int_{t_0}^{T} C_R(t)\,\mathrm{d}t}{\int_{t_0}^{T} C_N(t)\,\mathrm{d}t} \tag{1}$$

where $\Phi$ denotes the resilience level, and $C_R/C_N$ are the resilience curve and nominal performance curve, respectively. $t_0$ is the initial time before the occurrence of the disruption, while $T$ is the termination time of the recovery process and the system settles at a new stable state. Intuitively this is true since a larger area under the resilience curve generally means a smaller performance loss induced by the disruption, and thus the system is more resilient considering a given disruptive event. For classical resilience-base design problems, $C_R$ is further used in the objectives of the optimization models for finding the best design.

However, the above resilience term can only be measured after simulating a disruption event during the system nominal operation. This type of post-disruption information is hardly accessible during the design stage. Thus in order to bias the generative design process towards a more resilient system, a proxy is needed to represent the resilience level during the design stage. Section 6 discusses the selection for the proxy in detail. And the actual resilience levels of the candidate designs become accessible through more expensive simulations during the post-processing stage.

## 3 Modeling for Design Generator

Similar to discriminative methods, generative models have been well established for structured data, for example, the mixture model, the variational autoencoder, and the generative-adversarial network [22–24]. Different from the traditional discriminative approach, the generative model tries to learn the underlying representation of the training data and tune the parameters of the model to create realistic outputs that are similar to the input structure. And this rationale enables the research of generative methods in the design community. Various research has been proposed to use the deep learning model for generative designs [25–27]. But it was not until recently that the generative models have been extended to unstructured data inputs, i.e., graphs. In this study, we adopt the GVAE model, in which high-dimensional representations of the nodes' information are encoded as latent vectors to further reconstruct the original graph. Notice that the focus of this study is on ICIs, and their nodes/edges have physical information representing the operational condition. That information is unique and crucial for reconstructing a physical system.

**3.1 Modeling.** As a VAE model, the GVAE algorithm contains two consecutive steps—the encoding and decoding. Thus following sections discuss the modeling of such an algorithm from those two aspects as well as how to train the corresponding data-driven design model.

*3.1.1 Encoding.* Similar to the standard VAE model, the GVAE first needs to encode the high-dimensional node features $\mathbf{X}_v \in \mathbb{R}^f$ into latent vectors $z_v \in \mathbb{R}^d$, where $f$ denotes the number of features. And we use the gated graph neural network (GGNN) model as the nodes embedder [28]. The advantage of using the GGNN to embed the nodal information is that it can aggregate the neighborhood information in close proximity and preserve the information from the unique structure of each graph. And with the GGNN model, the input $\mathbf{X}_v$ is mapped into a multivariate diagonal Gaussian distribution in $d$-dimensional latent space, which are parameterized by $\mu_v$ and $\sigma_v$. And the latent representation of each node $z_v$ can be sampled from such a distribution. Following the convention of the standard VAE model, the regularization for encoding is the Kullback–Leibler (KL) divergence between the latent distribution and the standard Gaussian distribution. Thus, the loss term for the encoding step can be formulated as

$$\mathcal{L}_{\text{encode}} = \sum_{v \in G} \mathrm{KL}(\mathcal{N}(\mu_v, \mathrm{diag}(\sigma_v)^2) \| \mathcal{N}(0, \mathbf{I})) \tag{2}$$

*3.1.2 Decoding.* Different from the encoding part where the original graph information is embedded into latent vectors, the decoding aims to reconstruct a graph that is similar to the input data based on the encoded $z_v$. As for standard VAE with structured data inputs, the trained neural network for the decoder can automatically reconstruct the output through forwarding propagation. However, to reconstruct a graph, it is unclear how to directly output the whole graph in one single forward propagation process. Several studies have proposed to generate graph structures in an auto-regressive manner: starting from one node $i$, connect $i$ to nodes $j$, $k$, $m$, …, which have the highest probability for connection, then keep growing the graph node by node until termination [20,21]. Such an auto-regressive process involves two main decisions at each generation step: which node to connect and use what type of edge to connect the new node. To answer those two questions, the GVAE model used as the generator takes four steps: node initialization, edge selection, edge labeling, and node updating. And we discuss the formulations of these steps as follows.

During the node generation process, it is crucial to determine which information to be utilized for initializing the node representation. Since the encoding process has already preserved the physical features of each node by mapping them into a latent distribution, only the label information remains to be taken care of. And for physical systems, it is important to consider the type of each node during the reconstruction process. For example, the nodes inside a power system could have three different classes—the generations, the load bus, as well as transmissible node. And

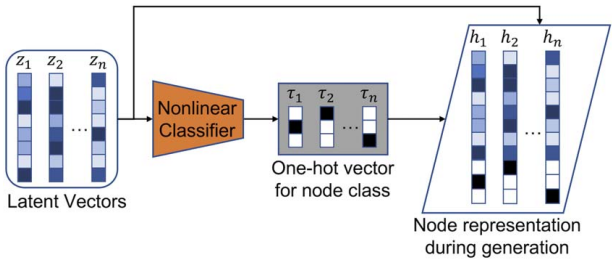



**Fig. 4 Node initialization for the decoding process: the latent vectors $z_v$ have been concatenated with their predicted nodal type, denoting as one-hot vectors**

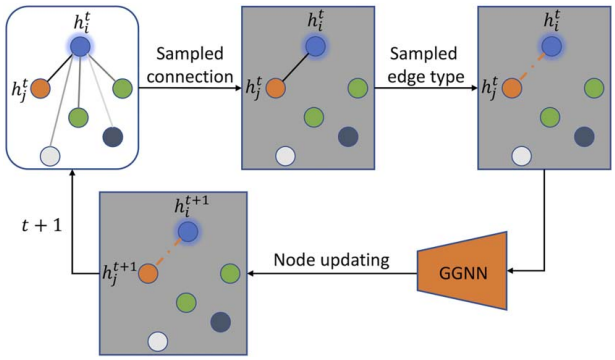



**Fig. 5 Illustration for the edge selection, edge labeling, and the node updating process**

the final generated design cannot randomly arrange disparate nodes violating physical constraints: they must follow the same pattern learned from the training dataset. Thus the node representation $h_v$ during the encoding step consists of two parts—the initial latent vector $z_v$ is concatenated with the one-hot class vector $\tau_v \in \mathbb{R}^c$ as shown in Fig. 4.$\tau_v = f(z_v)$ where $f(\cdot)$ could be any appropriate nonlinear classifier for deducing the node labels. And in this study, we use a three-layer fully connected neural network as the $f(\cdot)$.

Once each candidate node of the graphs has been initialized, the graph generation process can start by establishing edges between nodes. Considering the current starting node is $v_i$, the task is to derive the probability of connecting $v_i$ to all other candidate nodes $v_j$ by using an edge $ij_l$ (selection), where $\ell$ is the type of the connection (labeling). Notice that unlike traditional generative algorithms on graphs, where the edges are indifferent, here the edges in ICIs carry rich physical information. For example, in a power grid, the decision-maker should assign different capacities to edges according to the magnitude of power loads at both ends. And to accomplish the tasks for edge selection and labeling, the feature representation of the candidate edge can first be constructed by concatenating various vectors:

$$\phi_{i,j}^{(t)} := [h_i^{(t)}, h_j^{(t)}, d_{i,j}, H_{\text{init}}, H^{(t)}] \tag{3}$$

where $h_i$, $h_j$ are the nodal representations; $d_{i,j}$ is the distance measure between node $i$ and $j$, for instance, the electrical resistance between two buses in a power grid; $H_{\text{init}}/H$ is the initial/updated global graph feature, which is defined as the average of all nodal representations in this study:

$$H^{(t)} := \frac{1}{n}\sum_{i=1}^{n} h_i^{(t)} \tag{4}$$

Notice that all vectors in Eqs. (3) and (4) have an extra index for time-step $t$. This is due to the node updating procedure, which is discussed later. Equation (3) shows the advantage of the autoregressive generation process since the generation result not only depends on local information, e.g., $h_v$, but also considers the global state of the graph $H^{(t)}$. Once the feature vector for candidate edge is established, we can start to model the probability of connecting edge $ij$ using type $\ell$ connection. First, given the edge feature vector $\phi_{i,j}^{(t)} \in \mathbb{R}^f$, the distribution of choosing to connect $ij$ via edge type $\ell$ is expressed as the product of the probability of connecting $i$ to $j$ and the probability of using type $\ell$:

$$P(i \overset{\ell}{\leftrightarrow} j|\phi_{i,j}^{(t)}) = P(\ell|\phi_{i,j}^{(t)}, i \leftrightarrow j)P(i \leftrightarrow j|\phi_{i,j}^{(t)}) \tag{5}$$

These two probability terms can be further calculated by formulating softmax functions:

$$P(i \leftrightarrow j|\phi_{i,j}^{(t)}) = \frac{M_{i\leftrightarrow j}^{(t)} \exp\left[C(\phi_{i,j}^{(t)})\right]}{\sum_w M_{i\leftrightarrow w}^{(t)} \exp\left[C(\phi_{i,w}^{(t)})\right]} \tag{6}$$

$$P(\ell|\phi_{i,j}^{(t)}) = \frac{m_{i\overset{\ell}{\leftrightarrow} j}^{(t)} \exp\left[L_\ell(\phi_{i,j}^{(t)})\right]}{\sum_k m_{i\overset{k}{\leftrightarrow} j}^{(t)} \exp\left[L_k(\phi_{i,j}^{(t)})\right]} \tag{7}$$

In the above two probability terms, $C$ and $L_\ell$ represent two trainable, nonlinear functions, e.g., fully connected neural networks, that map the feature vector to a scalar score. Moreover, additional masking matrices $M_{i\leftrightarrow j}$ and $m_{i\overset{\ell}{\leftrightarrow} j}$ enforce any prior knowledge about the generated graph. For example, nodes $i$ and $j$ cannot be connected because they are from different regions, or nodes $i$ and $j$ can only be connected by a high-capacity edge $\ell$. After obtaining the probability distribution of the edge connection and labeling, new connection $ij_\ell$ can be sampled from the empirical distribution and the graph grows sequentially.

Notice that after choosing to connect node $i$ to node $j$, the graph structure has been changed due to the newly introduced edge and node. This changing in graph structure leads to changes in the node representation $h_i^{(t)}$, since nodes in the proximity of node $i$ are shifting. To update the node representation, GGNN can be used to re-derive $h_i^{(t)}$ in a recursive manner:

$$h_i^{(t)} = \text{GGNN}\left(h_i^{(t-1)}, \sum_{j|ij\in E} h_j^{(t-1)}\right) \tag{8}$$

where the initial node feature vector is $h_i^{(0)}$, and the updating aggregates all the feature vectors of nodes that are adjacent to $i$ in the current candidate graph design. In summary, the aforementioned edge selection, labeling, and node updating steps are illustrated in Fig. 5. And the corresponding decoding loss can be formulated as

$$\mathcal{L}_{\text{decode}} = \sum_{G\in\mathcal{D}} \log\left(P(G|G^{(0)}) \cdot P(G^{(0)}|z_v)\right) \tag{9}$$

where the loss term essentially measures the log-likelihood of reconstructing the graph $G$ given in the dataset $\mathcal{D}$, with the initial encoded latent vectors $z_v$. Due to space limitations, we have omitted some detail of the decoding process, including the termination condition for the graph generation. Readers can refer to Ref. [21] for more in-depth discussion.

**3.2 Performance Aware Generation.** So far, the basic loss terms for the encoding and decoding processes of the GVAE have been discussed. However, the conventional GVAE model with the $\mathcal{L}_{\text{encode}}$ and $\mathcal{L}_{\text{decode}}$ can only reconstruct candidate designs that are structurally similar to the samples in the input dataset. The established two loss terms only focus on quantifying the errors when encoding the latent design spaces and reconstructing from the encoding. To enable a generative design for resilience, the target performance criteria, i.e., the resilience level, needs to be correlated to the ICIs encoded in the structured, latent design space.

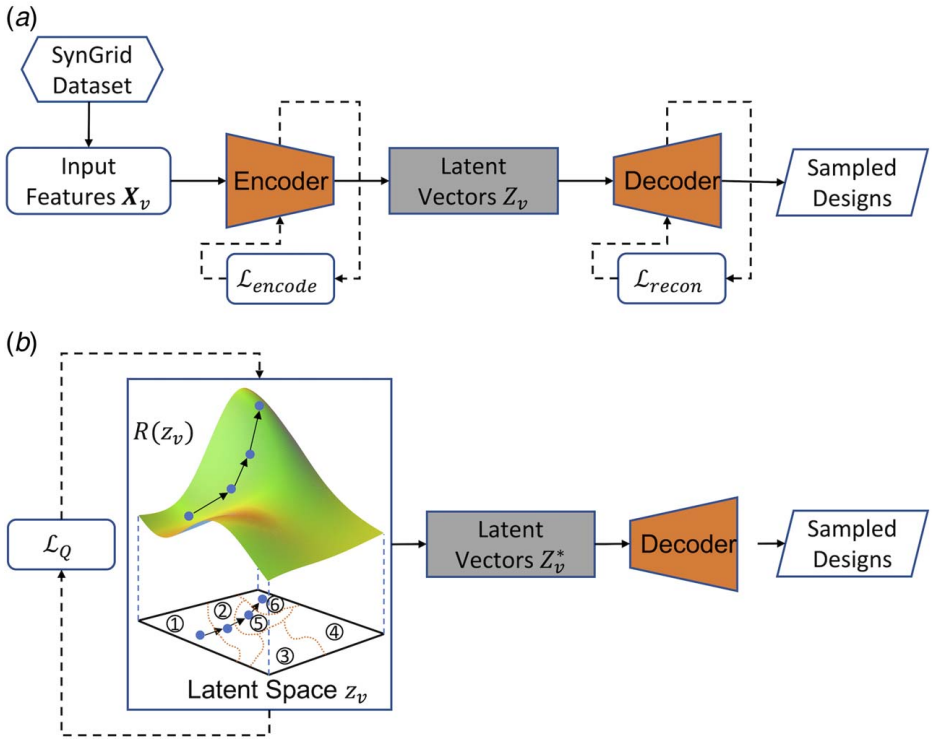



**Fig. 6 Flowchart for training the GVAE-based design generator: (*a*) Training the GVAE model with the standard encoding and decoding loss and (*b*) Performance aware generation by utilizing another surrogate $R(z_v)$ and gradient ascent with respect to $z_v$: the latent space has been divided into six parts for illustration purposes.**

And to achieve this, a nonlinear mapping between the latent vectors and the performance criteria can be constructed as

$$R(z_v) = \sum_v \sigma(f_1(z_v)) \cdot f_2(z_v) \tag{10}$$

where $f_1/f_2$ are two fully connected neural networks, and $\sigma$ is the sigmoid function. This formulation for estimating the performance based on latent vectors of the encoded ICIs adopts the idea from the GGNN graph-level regression model [28]. And it has two desirable properties:

- differentiable: $z_v$ can be optimized not only through the encoding/decoding loss but also via the gap between the intermediate design performance and the predefined performance criteria;
- soft attention mechanism: $\sigma(f_1\ (z_v))$ acts as an additional weight that decides which nodes are more relevant to the current graph-level estimation.

The loss term for finding the locally optimized $z_v^*$ is defined as

$$\mathcal{L}_Q = \|R(z_v) - Q\|_2^2 \tag{11}$$

where $Q$ is a predefined performance criterion. And Eq. (11) assembles a regression task on graphs. After training the $f_1$ and $f_2$ using the training dataset, gradient ascent with respect to the input latent vector $z_v$ for $R(z_v)$ can be performed to find promising new candidate designs. Particularly, starting from a latent vector of the encoded ICI, the design with better performance can be found by following the direction within the latent space to increase the performance metric, as shown in Fig. 6. The final design is decoded from the optimized $z_v^*$ by using the trained decoder model afterwards. In conclusion, with the encoding, decoding, as well as the performance aware generation process, the overall loss for training the generator is $\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{encode}} + \mathcal{L}_{\text{decode}} + \mathcal{L}_Q$.

## 4 Modeling for Design Estimator

Although the performance training enables the generator to output desirable designs that optimize the predefined performance criteria, the candidate designs are still confined to the latent design space of the input dataset. Moreover, the generator only uses a simple fully connected neural network model to predict the performance for efficiency. To further bias the design towards optimal design space and to fine-pick the "good" candidates, a more sophisticated performance estimator is needed. As a design estimator, it needs to learn the mapping between the input information (raw information of the ICI) and the output scalar, i.e., the performance metric. And deep learning framework is well known to have superb performance for predicting numerical labels from high-dimensional training data. As a result, in this study, we adopt the graph convolutional network (GCN) [29–31] to construct the performance estimator for the graph-like ICIs.

**4.1 Formulation.** The convolutional neural network (CNN) is well known for its excellent performance on image classification for nearly two decades [32]. However, the application of CNN on 2D Euclidean space is not extended to unstructured topological space until Bruna et al. [33] propose the generalization of CNN to signals defined on the graphical domains. This extension significantly improves the applicability of the convolution process on unstructured data, such as information generated from social networks or chemical compounds. After their work, the research about convolution on graphs diverges into two directions: one is working on the spatial domain of graphs while the other integrates the spectral graph theory into the convolution process. And the breakthrough works in Refs. [29,30] have greatly increased the potential of the spectral GCN by proposing an efficient first-order approximation of spectral graph convolutions. This study utilizes the latter approach, i.e., the spectral GCN. And the GCN takes the advantage of the underlying information embedded in the

adjacency matrix to perform the learning tasks for networks. Therefore, GCN has superb performance for the inference task on graph inputs.

It has been proved that the GCN can be derived from the traditional CNN by studying the spectral graph theory [29,30]. GCN can be treated as the generalized case of the CNN on arbitrary, unstructured space. And following the formulation in Ref. [31], the propagation rule or the convolutional operator for graphs requires the degree and weighted adjacency matrix as additional parameters. Different from conventional adjacency matrix with binary entries only, the weighted one can include other values than 0/1 representing the edge weights between different pairs of nodes. This characteristic enables the learning task fulfilled by GCN to take advantage of the rich graphical information of ICIs. On the other hand, the degree matrix $D$ is diagonal and aggregates the neighborhood information for each node. It can be obtained based on $A$: $D_{ii} = \sum_j A_{ij}$. With the degree and adjacency matrix on hand, the GCN propagation rule is defined as

$$\mathrm{GCN}(X) = D^{-1/2} A D^{-1/2} X \Theta \tag{12}$$

where $\Theta$ are the trainable weights of each convolutional layer. Therefore, the graph inference problem can be solved by constructing a deep neural network (DNN), for instance, $q = \sigma(f_2(f_1(\mathrm{GCN}_2(\mathrm{GCN}_1(X)))))$, where $\mathrm{GCN}_1/\mathrm{GCN}_2$ are the convolutional layers defined in Eq. (12), and $f_1/f_2$ are adequate nonlinear functions.

**4.2 Training.** As for using the GCN algorithm to evaluate the performance of generated designs, the specific training process needs to be discussed. Like conventional DNNs, the training process for the GCN-based estimator requires gradient information to optimize the trainable parameters as shown in Fig. 7. The dataset of ICIs contains the input feature $\mathbf{X} \in \mathbb{R}^{N \times f}$, where $N$ is the number of nodes inside the graph and $f$ is the number of features considered. For instance, for the 123-bus power grid shown in Fig. 2, $\mathbf{X}$ contains 123 rows and six columns that includes the information of the power demand magnitude, generation capacity level, the voltage angle, etc. Whereas the label $y$ used for training the estimator is a numerical value for each training sample. Since the generation process needs to be biased to have a more resilient system design, the label should represent the overall system resilience level. And the choice for the label $y$ is further discussed in Sec. 6. The loss term used to train the estimator is the mean squared error (MSE) between the estimated label $q$ and the actual performance $y$:

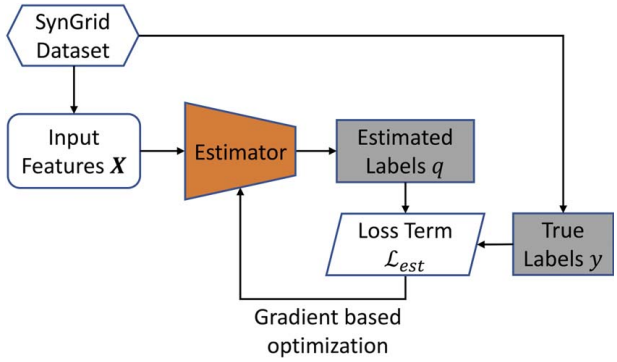



**Fig. 7 Flowchart for training the GCN-based design estimator**

$$\mathcal{L}_{\mathrm{est}} = \frac{1}{n} \sum_i (q_i - y_i)^2 \tag{13}$$

Notice that, unlike the design generator, the estimator can be pretrained on the existing training dataset so that estimating the performance of new designs is computationally cheap during the iterative generating process.

After training a GCN-based design estimator, we can combine the module with the generator to form an iterative process for refining the designs. As shown in Fig. 1, a fixed number of $z_v$ is sampled and are used to generate the corresponding designs. Then the estimator determines the best batch of designs in terms of metric $Q$. If the process is not converged, i.e., the difference between the last best design and the current optimal one is large, then the top $c$ generated designs are mixed with the original dataset to perform another round of generation. To improve gradually the

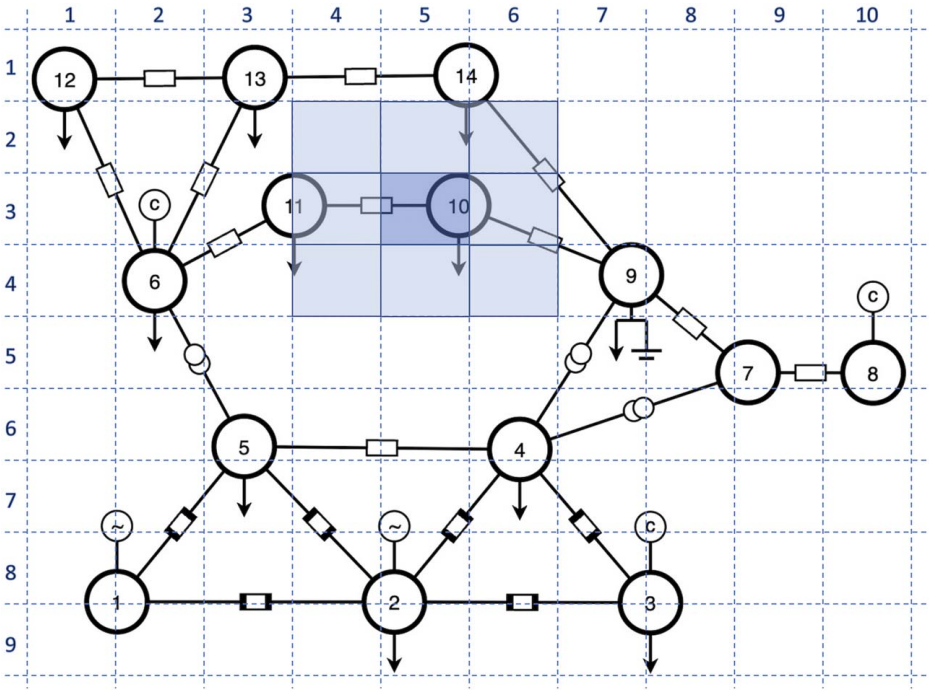



**Fig. 8 A small-scale power system mapped to an example meshgrid with a disruptive event sampled at cell (3,5)**

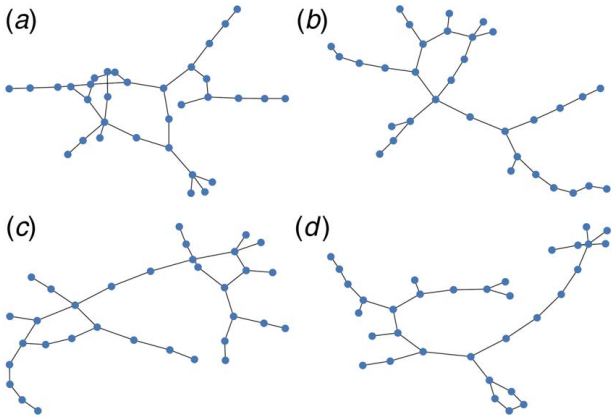



**Fig. 9 Examples of small-scale Watts–Strogatz graphs used as the training data**

generated samples, the batch of the best designs replace the group of the original training data that have been ranked as the lowest performance.

## 5 Post-Processing Simulation

So far the performance criterion $Q$ for training the design generator/estimator has been discussed. And for an efficient training process, the performance of each design is represented by an easy-to-evaluate scalar metric: rather than the actual resilience level, it is a proxy to access the system resilient performance. Recall the system resilience which mainly measures the system's capability toward unforeseen disruptions. And it is hard to obtain the resilience level without running experiments for systems undergoing disturbances. As a result, we add a post-processing stage to further evaluate the candidate designs filtered by the surrogate metric $Q$. Since the initial design space has been shrunk from millions of candidates to hundreds through the iterative biasing generation, it becomes tractable for running more complex, simulation-based evaluations during the post-processing stage.

Following the convention of the studies for ICI, here the metric used in the post-processing simulation is the expected demand not supplied (EDNS). And it is defined as

$$\text{EDNS} = \sum_{e_i \in S_{e_i}} P_{e_i} C_{e_i} \tag{14}$$

where $e_i$ denotes a possible disruptive event and $S_{e_i}$ is the set of all events simulated. $P_{e_i}$ represents the probability of having the event $e_i$, and $C_{e_i}$ is the amount of lost demands after $e_i$. In Fig. 3, the EDNS can be viewed as the area between $C_R$ and $C_N$, which quantifies the capability of the system withholding external disruptions. And the optimal design can be determined as the one with the smallest EDNS after acquiring the post-processing results.

To obtain the $C_{e_i}$ after each $e_i$, we establish meshgrids for mapping each candidate design with geological information. Figure 8 illustrates the mesh view of the IEEE 14-bus system with a disruptive event. Notice that, the mapped mesh view of the system is helpful for simulating realistic disruptive events since external disruptions are usually confined within a specific region. For example, as shown in Fig. 8, a storm with predefined $P_{e_i}$ is simulated to happen at cell (3,5), which leads to the highest probability of failure for node 10 and edge (10,11). Also, the components within the close proximity of cell (3,5), including node 10, 14 and edges around, incur smaller failure probability. Thus, based on the information from the meshgrid, the curtailed performance of the ICI caused by random disasters can be simulated. For instance, $C_{e_i}$ from solving the optimal power flow (OPF) problem for power systems, or obtaining the maximum flow solution for an impacted supply chain network, after disconnecting the damaged components. With each $P_{e_i}$ and the corresponding $C_{e_i}$, the overall EDNS of the candidate design can be found by performing several runs of simulations in a Monte Carlo manner.

## 6 Case Studies

To validate the proposed generative design method, experiments about designing different kinds of ICIs are considered. Section 6.1 presents the design results for small-scale synthetic network systems. And Sec. 6.2 further discusses the applicability of the proposed framework on large-scale IEEE test feeders. The design generator and estimator are implemented in Pytorch and Pytorch Geometric packages. The model is trained on a PC with a 10700K 8-core processor and NVIDIA GTX3080 GPU.

**6.1 Synthetic Network Systems.** First, we utilize the NetworkX package [34] in PYTHON to randomly generate 10,000 Watts–Strogatz small-world graphs as the design training dataset. And 10% of the dataset is used for validation. This type of random graphs has the small-world properties, which include high clustering and short average path length. Those properties enable researchers to use the Watts–Strogatz model to understand realistic networked systems, for instance, social networks and supply chain systems. Figure 9 demonstrates example Watts–Strogatz graphs generated for the case study. Here a small-scale system is considered, which includes 33 nodes and the same number of edges. We can see from Fig. 9 that the initial graphs used for training has few clusters and several nodes have higher degree comparing to remaining nodes. This is desirable since for a supply chain network, nodes with higher degree can represent transportation hubs, while clusters indicate different coverage for separated communities.

*6.1.1 Generative Network Design.* As for designing the ICI, we need to define the physical features of the nodes and edges. And in this case study, the nodes have a numerical feature to indicate the magnitude of demands/supplies as well as a categorical feature declaring the type of the node: demand, supply, and transfer. This categorical feature is pre-sampled based on the degree of each node. For example, a node with higher degree has much higher probability of being the supply node, while nodes with small degree are likely sampled as demand or transfer nodes. On the other hand, the edges have two numerical feature denoting the capacity and cost of each edge. Like classical network flow problems, the cost is measured in terms of per unit flow on the corresponding edge. And to quickly evaluate the design generated during the iterative process, the performance metric $Q$ is defined as the magnitude of maximum flow with the minimum cost inside the network, denoting as $f_{\max}$. In order to solve this multi-source multi-sink problem and derive the $f_{\max}$, two dummy nodes are added to the sampled Watts–Strogatz graphs. These nodes represent

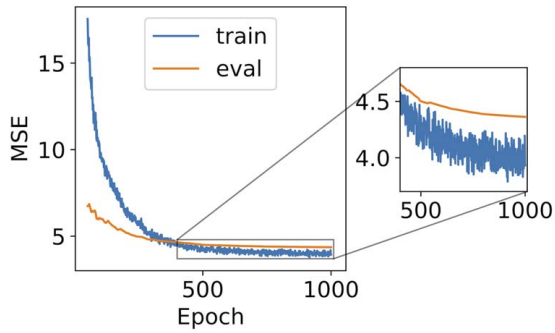



**Fig. 10 The training loss history for the estimator on synthetic network system data**

the pseudo-source and destination node, respectively. And here the Ford–Fulkerson algorithm is utilized to calculate $f_{\max}$ for each candidate design in $O(Ef)$ time, where $E$ is the number of edges and $f$ denotes the maximum flow magnitude.

Once the training dataset is ready, the estimator can be pretrained for quickly evaluating the designs during generation. Figure 10 shows the training history of the GCN-based estimator, where the $y$-axis represents the MSE between the ground-truth and the estimated performance metric. Notice that here the evaluation MSE is lower than that of the training data during the initial epochs. This is due to the fact that at each epoch, we train the model with stochastic gradient descent and sum up the training error of each sample. Then the training MSE is derived as the average of the overall error terms. As a result, the error for the first training sample could be significantly larger than that of the last few samples. On the other hand, the error of each validation sample is at least close to that of the last batch of training samples. Thus, the initial validation errors have smaller magnitude than the derived training errors.

Based on the training dataset, our goal for the design is to come up with a new and optimal design that satisfies the demands within the network while maintains steady performance after disturbances. We set the termination criterion for the training process as reaching 150 iterations. Figure 11 summarizes the distribution of the performance metric $Q$ of the training dataset and the 500 generated designs, as well as the best design evaluated by the estimator at each iteration. Moreover, to show the performance biasing design process for a resilient system, we also show the distribution of the EDNS of the 500 designs along with the generation metric $Q$. Notice that the in-production generative design process only needs to execute the post-processing step after the training has been converged. Here to demonstrate the correlation between the training metric $Q$ and the EDNS, we conduct post-processing simulations even during the middle of the generation.

According to the results shown in Fig. 11, the generative design method successfully biases the candidate designs toward predefined performance metric, especially for the training label $Q$. Though the changes in the resilience index EDNS are not significant at the same level as that of $Q$, there still exists a trend for improvement: the overall EDNS decreases significantly during the first 100 training iterations. On the other hand, take the best design generated at each iteration as an example; the first version of the design contains long branches and few redundant paths. This is not a good practice for designing a resilient ICI like supply chain network. After several iterations' training, the candidate designs become to have clustered communities as well as central hubs along with substations. Those hierarchical components lead to a much smaller EDNS. At the last iteration, the central skeleton of the network changes from a tree structure to a ring shape, which further introduces more redundancy to the already high clustering network. This shifting in design strategies also demonstrates the capability of the proposed framework on learning good proprieties from existing training samples.

*6.1.2 Comparative Study.* To demonstrate the advantage of the proposed generative design framework, a comparative study based on the synthetic network design dataset is first conducted with and without the performance aware routine. And Table 1 summarizes the results in terms of system performance as well as the computational cost. From the table, although the average training time per iteration is reduced after removing the performance aware loss, the convergence toward the optimal design is much slower. When the generator is trained with $\mathcal{L}_{\text{encode}}$ and $\mathcal{L}_{\text{decode}}$, the performance criterion as well as the EDNS at iteration 150 only surpasses that of the model trained with the additional $\mathcal{L}_Q$ at iteration 50. And this observation is consistent with the motivation of adding $\mathcal{L}_Q$ to

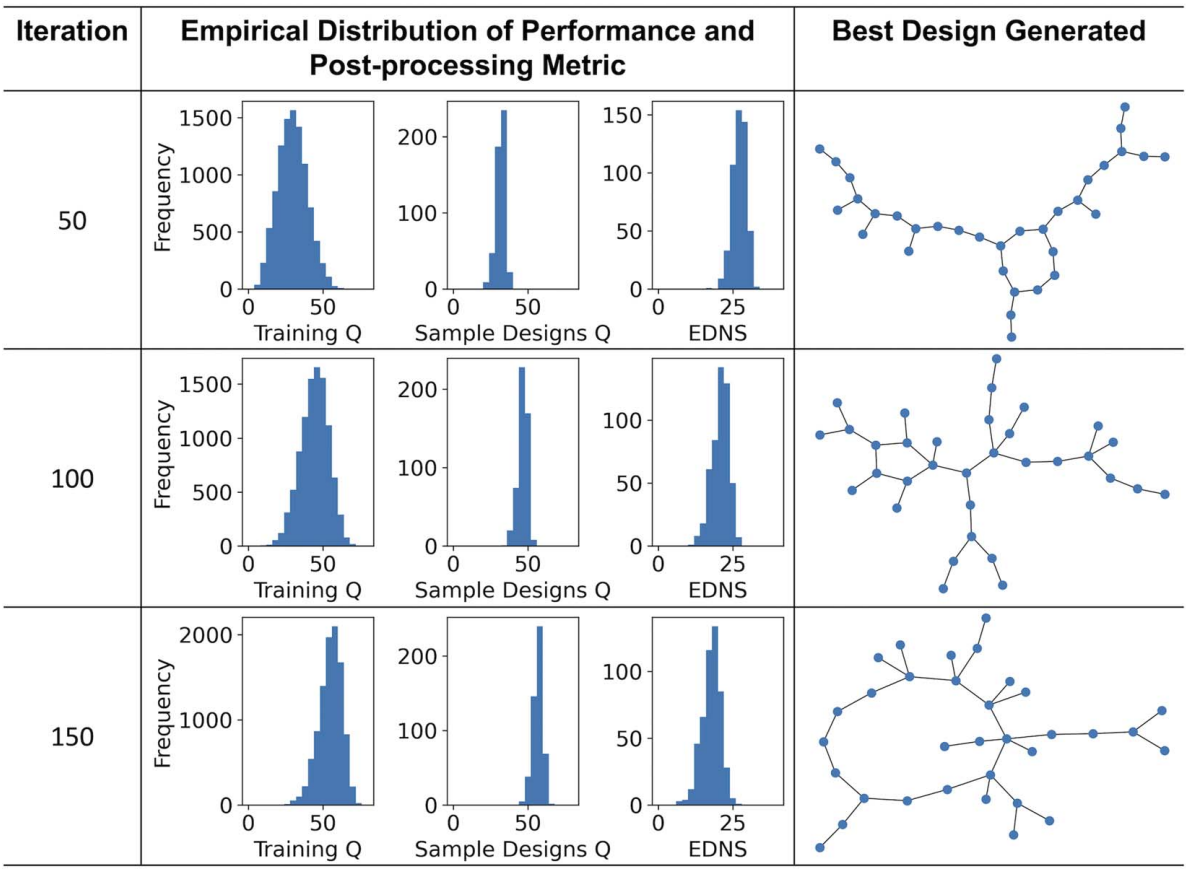



**Fig. 11 Snapshots of the empirical distribution of the metric Q of the training data and the best designs generated, as well as the post-processing EDNS metric at different training iterations for the synthetic network test case**

**Table 1 Comparative analysis results of the system performance and computational complexity for the synthetic network systems**

| Generator loss terms | Avg. training time/iter | Iteration | Training Q | | | Sample designs Q | | | EDNS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | P10 | P50 | P90 | P10 | P50 | P90 | P10 | P50 | P90 |
| $\mathcal{L}_{\text{encode}} + \mathcal{L}_{\text{decode}}$ | 219.3 s | 50 | 11.85 | 19.81 | 28.16 | 18.95 | 21.59 | 23.59 | 34.4 | 39.16 | 42.33 |
| | | 100 | 19.57 | 27.38 | 34.26 | 27.23 | 30.11 | 32.55 | 27.17 | 33.2 | 35.94 |
| | | 150 | 25.54 | 31.7 | 36.52 | 32.18 | 34.97 | 37.22 | 21.77 | 26.84 | 28.95 |
| $\mathcal{L}_{\text{encode}} + \mathcal{L}_{\text{decode}} + \mathcal{L}_Q$ | 237.9 s | 50 | 17.16 | 29.53 | 42.77 | 27.53 | 32.12 | 35.02 | 24.04 | 27.31 | 29.69 |
| | | 100 | 31.8 | 44.82 | 56.11 | 42.1 | 46.95 | 50.4 | 17.91 | 20.62 | 22.8 |
| | | 150 | 44.38 | 55.7 | 64.25 | 52.64 | 56.76 | 60.36 | 14.15 | 17.96 | 21.22 |

the generator. Since this performance aware generation loss make finding the best latent representation $z_v^*$ depends not only on the structure information but also on the resulting performance. Besides updating the training dataset, this enables a way to accelerate the convergence of the candidate design toward desirable performance criteria.

Furthermore, the training time results in Table 1 indicate that removing the training process for Eq. (11) can reduce the computational cost by around 19 s per iteration (8.0%). Compared to the cost for optimizing $\mathcal{L}_{\text{encode}}$ and $\mathcal{L}_{\text{decode}}$, this difference is not significant. Although deriving the performance regression loss $\mathcal{L}_Q$ requires training two additional neural networks $f_1$ and $f_2$ defined in Eq. (10), we simply set these two neural network models as three-layer fully connected networks for efficient training. And optimizing the generator through $\mathcal{L}_Q$ is a regression task consisting of only numerical backpropagation computations. Whereas the major computation burden for training the generator is coming from the decoding part. Since to interpolate a valid design from $z_v$ during training, the stakeholder needs to grow the candidate design node by node while updating the states of established nodes and the graph. Training this decoding process requires a more computationally expensive process defined by Eqs. (3)–(8).

Additionally, we compare the performance of the proposed framework to existing baseline algorithms, such as the resilience-based design (RBD) method proposed in Refs. [35–37]. Specifically, the heuristic optimization-based approach from Ref. [35] is reproduced because it adopts the same performance criteria $f_{\max}$ to approximate the system resilience and considers similar synthetic supply chain networks as the case studies. To compare the performance of a resilient design, systems with three different scales, 33, 100, and 250 nodes, are evaluated. For the RBD method, we establish candidate edges by setting the graph as fully connected. The governing equation for each edge is formulated using the parameters given in Ref. [35]. And for the generative method, three training datasets each of 10,000 synthetic samples are produced. The deep generative model is then trained for 150 iterations. Results shown in Table 2 are the ones with the best EDNS selected from the post-processing step.

Based on the numerical results for the system resilience, the RBD method can indeed optimize the proxy performance metric $Q$. However, the more accurate resilience index EDNS indicates the insufficiency of only optimizing toward $Q$. One potential reason for this discrepancy is that the design solved by RBD tends to have multiple long branches, instead of loops or clusters found in the designs from the generative framework. Since RBD only selects an edge that can attain a high level of $Q$ with the minimum cost, it does not reckon about potential redundancies presented in real-life systems. And the difference in the EDNS results for RBD and the generative method demonstrates the versatility of the latter approach. One advantage of the generative method is that it can mine intrinsic knowledge from the training data via the deep learning mechanism. On the contrary, the RBD is a discriminative method that is not context-aware. The design solution solely depends on the predefined objective and constraints, which lacks the global knowledge of a resilient design.

As for the computational cost, the RBD method does not require training but needs to run repeatedly for each test case. On the other hand, the proposed generative framework can be trained offline. And during the online stage, we can have an initial input $z_v$ as a random normal vector. Then the candidate design can be generated by gradient ascent and feeding $z_v^*$ into the trained decoder. Thus, the comparison for computational cost focuses on the time spent for outputting the best design solution only. During the generation stage, the most expensive part of RBD is solving the $f_{\max}$ in $O(Ef)$ time for each possible solution while optimizing the solution by a heuristic method. The corresponding computation cost is quadratically increasing with respect to the number of nodes. And this can be seen from the numerical results in Table 2. For the generative method, we include the post-processing time plus the time for testing/generation. And the computational cost for generation is slightly scaled with respect to the size of the network system.

**6.2 IEEE Power Grids Test Case.** Despite the small-scale test case based on synthetic networks, we also consider real-world dataset about the power systems. In order to train the design estimator and the generator, we first generate a power grid dataset consisting of 138,000 sample systems that range from 50 nodes to 150 nodes. The sample system designs are generated by the SynGrid package in MATPOWER [38]. And since the exact resilience level of a system design is hard to quantify without knowing the online response during disruptions, here we adopt a surrogate metric, the total capacity ratio of the system. This metric implicitly measures how well the system gonna behave towards external disruptions. And it is defined as

$$C^R = \frac{\sum_{ij|ij\in E} f_{ij}}{\sum_{ij|ij\in E} u_{ij}} \tag{15}$$

where $f_{ij}$ is the actual flow on edge $ij$ from the OPF results and $u_{ij}$ is the capacity assigned to $ij$. A lower capacity ratio means that the design can have more room for unpredictable flows. And this metric can indicate the tolerance to failures for an ICI during disruptive events, e.g., sudden load surges or line outages. However, the final performance metric requires additional compensations to prevent the generated designs from assigning unnecessary large capacity to every edges for obtaining a low $C^R$. Therefore, we take

**Table 2 Comparison results with the baseline algorithm for the synthetic network systems**

| Method | Test case | Sample design Q | EDNS | Solving time |
|---|---|---|---|---|
| RBD | 33 nodes | 64.11 | 17.89 | 18.30 s |
| | 100 nodes | 180 | 55.65 | 75.49 s |
| | 250 nodes | 346.3 | 150.7 | 233.2 s |
| Generative | 33 nodes | 64.24 | 10.16 | 83.77 s |
| | 100 nodes | 178.2 | 42.77 | 91.03 s |
| | 250 nodes | 337.4 | 130.5 | 108.6 s |

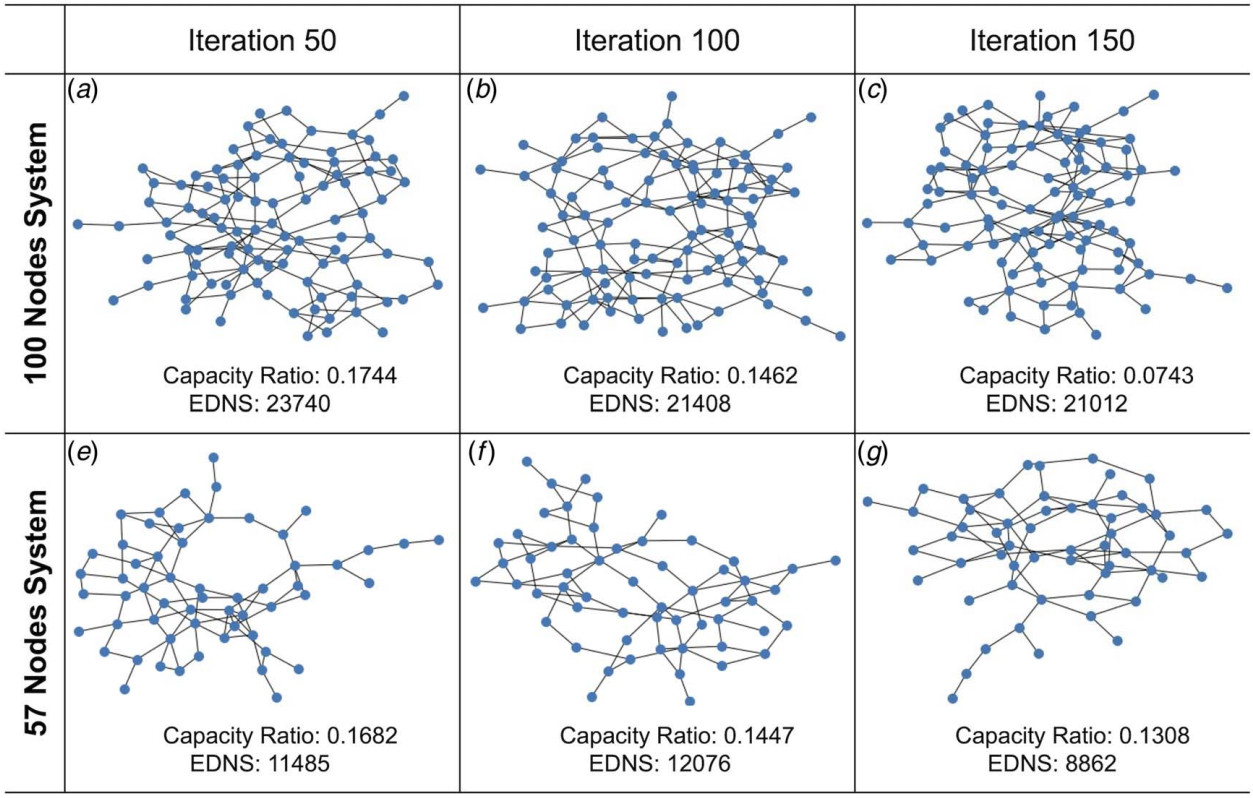



**Fig. 12 Illustrations of new designs generated as well as their capacity ratios and EDNS for the power grid test case**

the cost of having large edge capacity into consideration. And the total cost of establishing all edges with capacity $u_{ij}$ is denoted as

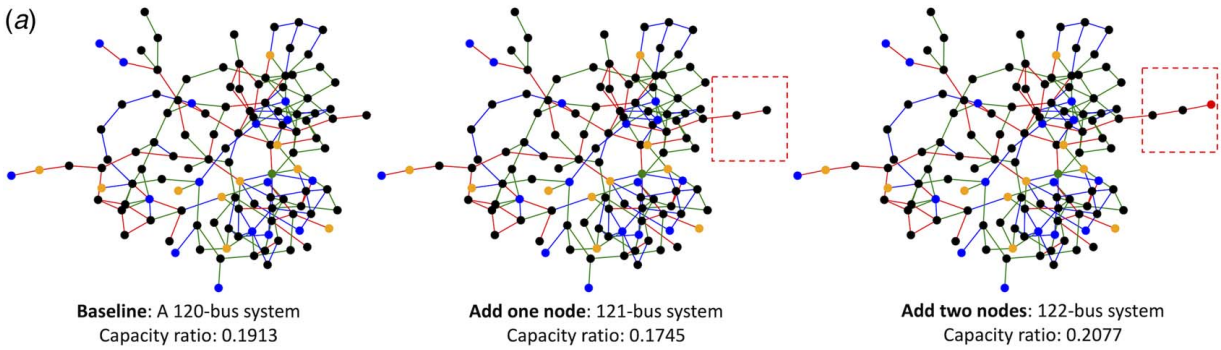
$$C^E = \sum_{ij|ij\in E} a \cdot u_{ij} \tag{16}$$

where $a$ is a predefined parameter measuring the unit cost of edge capacity. The training label is then defined as

$$Q = \alpha C^R + (1-\alpha)\frac{C^E}{K} \tag{17}$$

where $\alpha$ is the weight hyperparameter and $K$ is a constant for normalizing. The training task is thus to minimize $Q$. Notice that the proposed framework is not limited to any specific form of metric, and it can adopt other appropriate resilience-driven metrics by changing the formulation of the $Q$.

To show the applicability of the design framework, we trained the model on the SynGrid dataset and followed the steps depicted in Sec. 3.2 to generate brand new designs with a low level of capacity ratio. Figure 12 shows the samples from the progress of the gradient ascent and the designs decoded from the embedded




(b)

| Node Attributes | Black | Blue | Green | Orange | Purple | Red |
|---|---|---|---|---|---|---|
| | Load Bus | Hydro | Wind | Natural Gas | Coal Fire | Nuclear |

| Edge Attributes: Rated Capacity | Red | Green | Blue |
|---|---|---|---|
| | >= 1000 MVA | 100 – 1000 MVA | < 100 MVA |

**Fig. 13 The solution of an expansion problem generated from the framework and the specific physical information considered: (*a*) the system is expanded by sequentially adding nodes and edges outlined in the dotted box, and (*b*) the nodes have six types while the edges have three different categories**

$z_v$. Here, a mid-scale 57 nodes system and a large-scale 100 nodes system are considered. Although the generative design process only uses a surrogate metric instead of the actual system resilience like EDNS, both the proxy metric used for training and the more expensive resilience index show the similar trend throughout the iterations. This finding suggests that the relatively cheap performance metric $Q$ is suitable for deriving a resilient design via the generative process.

Moreover, different from designing a brand new system, the expansion problem sometimes is more prevalent for ICIs. For instance, determining where to add new buses in an existing power grid to enhance the current service or how to allocate new transportation hubs in a supply chain network to improve the coverage. In order to solve an expansion problem, we can include the prior knowledge constraints, i.e., existing connection information, as masking matrices $M_{i\leftrightarrow j}$ and $m_{i\overset{\ell}{\leftrightarrow} j}$ shown in Eqs. (6) and (7). Such an information injection during the generation process is also useful when the stakeholder requires to put hard constraints on the final design.

Figure 13 illustrates the expansion process for a 120-bus system after adding two additional nodes. The figure also shows the physical information for the original ICI and the expanded system. Six types of nodes are considered and the edges have three categories based on their rated capacities. For this test case, the objective is to enhance the system generation capacity by adding a nuclear power plant with an additional transitive bus. So the design solution is the optimal location to connect the new components and the corresponding connection type. The three networks shown in Fig. 13 are the original system, the best solution for expanding after one generation iteration, and the final expanded system. And the second network is constructed by connecting the extra transitive node with a high-capacity edge, denoted as the additional line in the dotted box. This intermediate result includes additional redundancy without increasing the steady-state power flow $f_{ij}$. As a result, the capacity ratio decreases based on Eq. (15). However, the final expanded system has the nuclear generation connected to the new transitive bus. Compared to other types of generation nodes with a few hundred megavolt-ampere (MVA) capacity, the nuclear power plant usually has the larger generation capacity (above 1000 MVA). And this addition enlarges the $f_{ij}$ significantly across the entire system: average $f_{ij}$ increases from 150 MVA to 170 MVA in this case. Therefore, the capacity ratio increases after adding nuclear generation, and the numerical results here show a non-monotonic pattern in terms of the proxy metric. Notice that the generated designs from the GVAE model only contain categorical information for each node and edge, e.g., a node is assigned as a coal-fired power plant and an edge is allocated with high capacity. To conduct OPF simulations in the MATPOWER and to obtain the capacity ratio, we have sampled numerical ratings for generations and edges, based on the statistics in the SynGrid package.

## 7 Conclusion

In this study, a generative design framework for interdependent network systems has been developed. Different from traditional model-based design methods, the developed approach utilizes advanced graph learning algorithms to enable a data-driven network design solution, therefore eliminating the requirements for developing explicit mathematical models for the interdependent networked systems. The proposed design framework is capable of mining useful properties from existing system designs and identifying candidate designs that meet predefined performance criteria. Moreover, prior knowledge about the system can be conveniently included for design during the generation process through masking information included in the generator module of the framework. Case study results based on various scales of synthetic network systems along with power systems have shown the applicability of the developed generative design approach. For future studies, it would be meaningful to investigate directly coupling of the general resilience metrics with the system specific design performance criteria. Additionally, the case studies presented in this paper considered limited types of interdependent networks such as the power systems, and other engineering system design applications can be explored using the developed generative design approach.


## Acknowledgment

This research is partially supported by the U.S. Department of Energy's Office of Nuclear Energy under Award No. DE-NE0008899 and the National Science Foundation (NSF) Engineering Research Center for Power Optimization of Electro-Thermal Systems (POETS) with cooperative agreement EEC-1449548.


## Conflict of Interest

There are no conflicts of interest.

## Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

## Nomenclature

### Parameters

$\mathcal{G}$ = graph
$H$ = graph representation in the generator
$h_i$ = node representation in the generator
$\mathbf{z}_v$ = latent feature vector for node $v$
$\mathbf{X}_v$ = feature vector for node $v$
$i, j$ = node index in the graph
$\tau_v$ = class of node $v$
$\phi_{ij}$ = feature vector for edge generation

### Variable

$C, L_\ell, f_1, f_2$ = trainable neural networks

## References


[1] DeAngelis, D. L., 1980, "Energy Flow, Nutrient Cycling, and Ecosystem Resilience," Ecology, **61**(4), pp. 764–771.
[2] Goerger, S. R., Madni, A. M., and Eslinger, O. J., 2014, "Engineered Resilient Systems: A Dod Perspective," Procedia Comput. Sci., **28**, pp. 865–872, 2014 Conference on Systems Engineering Research.
[3] Walker, B., Holling, C., Carpenter, S., and Kinzig, A., 2004, "Resilience, Adaptability and Transformability in Social-Ecological Systems," Conserv. Ecol., **9**(2), p. 5.
[4] Wu, J., and Wang, P., 2020, "Risk-Averse Optimization for Resilience Enhancement Under Uncertainty," Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Virtual, Online, Aug. 17–19, ASME, p. V11AT11A042.
[5] Ouyang, M., and Fang, Y., 2017, "A Mathematical Framework to Optimize Critical Infrastructure Resilience Against Intentional Attacks," Comput. Aided Civil Infrastruct. Eng., **32**(11), pp. 909–929.
[6] Wu, J., and Wang, P., 2021, "Post-Disruption Performance Recovery to Enhance Resilience of Interconnected Network Systems," Sustain. Resilient Infrastruct., **6**(1–2), pp. 107–123.
[7] Wu, J., and Wang, P., 2021, "Risk-Averse Optimization for Resilience Enhancement of Complex Engineering Systems Under Uncertainties," Reliab. Eng. Syst. Saf., **215**, p. 107836.
[8] Chen, C., Wang, J., Qiu, F., and Zhao, D., 2016, "Resilient Distribution System by Microgrids Formation After Natural Disasters," IEEE Trans. Smart Grid, **7**(2), pp. 958–966.
[9] Wu, J., Chen, X., Badakhshan, S., Zhang, J., and Wang, P., 2022, "Spectral Graph Clustering for Intentional Islanding Operations in Resilient Hybrid Energy Systems," IEEE Trans. Ind. Inform., pp. 1–9.
[10] Ambia, M. N., Meng, K., Xiao, W., and Dong, Z. Y., 2021, "Nested Formation Approach for Networked Microgrid Self-Healing in Islanded Mode," IEEE Trans. Power Deliv., **36**(1), pp. 452–464.

[11] Dall'Anese, E., and Giannakis, G. B., 2014, "Sparsity-Leveraging Reconfiguration of Smart Distribution Systems," IEEE Trans. Power Deliv., **29**(3), pp. 1417–1426.
[12] Wu, J., and Wang, P., 2019, "A Comparison of Control Strategies for Disruption Management in Engineering Design for Resilience," ASCE-ASME J. Risk Uncert. Eng. Syst. Part B Mech. Eng., **5**(2), p. 020902.
[13] Yodo, N., and Wang, P., 2016, "Resilience Allocation for Early Stage Design of Complex Engineered Systems," ASME J. Mech. Des., **138**(9), p. 091402.
[14] Sharma, N., Tabandeh, A., and Gardoni, P., 2018, "Resilience Analysis: A Mathematical Formulation to Model Resilience of Engineering Systems," Sustain. Resil. Infrastruct., **3**(2), pp. 49–67.
[15] Bourennani, F., Rahnamayan, S., and Naterer, G. F., 2015, "Optimal Design Methods for Hybrid Renewable Energy Systems," Int. J. Green Energy, **12**(2), pp. 148–159.
[16] Li, D., Wu, J., Zhang, J., and Wang, P., 2021, "Co-Design Optimization of a Combined Heat and Power Hybrid Energy System," Proceedings of the ASME 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Virtual, Online, Aug. 17–19, ASME, p. V03AT03A028.
[17] Yodo, N., and Wang, P., 2016, "Resilience Modeling and Quantification for Engineered Systems Using Bayesian Networks," ASME J. Mech. Des., **138**(3), p. 031404.
[18] Yodo, N., Wang, P., and Zhou, Z., 2017, "Predictive Resilience Analysis of Complex Systems Using Dynamic Bayesian Networks," IEEE Trans. Reliab., **66**(3), pp. 761–770.
[19] You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J., 2018, "GraphRNN: Generating Realistic Graphs With Deep Auto-Regressive Models," Proceedings of the 35th International Conference on Machine Learning, J. Dy and A. Krause, eds., Stockholm, Sweden, July 10–15, PMLR, Vo1. 80, pp. 5708–5717.
[20] Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W. L., Duvenaud, D., Urtasun, R., and Zemel, R., 2019, *Efficient Graph Generation With Graph Recurrent Attention Networks*, Curran Associates Inc., Red Hook, NY, pp. 4255–4265.
[21] Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. L., 2018, "Constrained Graph Variational Autoencoders for Molecule Design," Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, Canada, Dec. 3–8, Curran Associates Inc., pp. 7806–7815.
[22] Bailey, T., and Elkan, C., 1994, "Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Biopolymers," Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology, Stanford, CA, Aug. 14–17, Vol. 2, pp. 28–36.
[23] Kingma, D. P., and Welling, M., 2014, "Auto-Encoding Variational Bayes," Proceedings of the 2nd International Conference on Learning Representations, Banff, Canada, Apr. 14–16.
[24] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014, "Generative Adversarial Nets," Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, Canada, Dec. 8–13.
[25] Heyrani Nobari, A., Chen, W., and Ahmed, F., 2021, "Range-Gan: Range-Constrained Generative Adversarial Network for Conditioned Design Synthesis," Proceedings of the ASME 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Virtual, Online, Aug. 17–19, Vol. 85390, American Society of Mechanical Engineers, p. V03BT03A039.
[26] Qian, C., Tan, R. K., and Ye, W., 2022, "An Adaptive Artificial Neural Network-Based Generative Design Method for Layout Designs," Int. J. Heat Mass Transfer, **184**, p. 122313.
[27] Oddiraju, M., Behjat, A., Nouh, M., and Chowdhury, S., 2022, "Inverse Design Framework With Invertible Neural Networks for Passive Vibration Suppression in Phononic Structures," ASME J. Mech. Des., **144**(2), p. 021707.
[28] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S., 2016, "Gated Graph Sequence Neural Networks," Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico, May 2–4.
[29] Hammond, D. K., Vandergheynst, P., and Gribonval, R., 2011, "Wavelets on Graphs Via Spectral Graph Theory," Appl. Comput. Harmonic Anal., **30**(2), pp. 129–150.
[30] Defferrard, M., Bresson, X., and Vandergheynst, P., 2016, "Convolutional Neural Networks on Graphs With Fast Localized Spectral Filtering," Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, Dec. 5–10, Curran Associates Inc., pp. 3844–3852.
[31] Kipf, T. N., and Welling, M., 2017, "Semi-Supervised Classification With Graph Convolutional Networks," Proceedings of the 5th International Conference on Learning Representations, Toulon, France, Apr. 24–26.
[32] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., 1998, "Gradient-Based Learning Applied to Document Recognition," Proc. IEEE, **86**(11), pp. 2278–2323.
[33] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y., 2014, "Spectral Networks and Locally Connected Networks on Graphs," Proceedings of the 2nd International Conference on Learning Representations, Banff, Canada, Apr. 14–16.
[34] Hagberg, A. A., Schult, D. A., and Swart, P. J., 2008, "Exploring Network Structure, Dynamics, and Function Using Networkx," Proceedings of the 7th Python in Science Conference, Pasadena, CA, Aug. 19–24, G. Varoquaux, T. Vaught, and J. Millman, eds., pp. 11–15.
[35] Zhang, X., Mahadevan, S., Sankararaman, S., and Goebel, K., 2018, "Resilience-Based Network Design Under Uncertainty," Reliab. Eng. Syst. Saf., **169**, pp. 364–379.
[36] Cimellaro, G. P., Villa, O., and Bruneau, M., 2015, "Resilience-Based Design of Natural Gas Distribution Networks," J. Infrastruct. Syst., **21**(1), p. 05014005.
[37] Suribabu, C., 2017, "Resilience-Based Optimal Design of Water Distribution Network," Appl. Water Sci., **7**(7), pp. 4055–4066.
[38] Wang, Z., Scaglione, A., and Thomas, R. J., 2010, "Generating Statistically Correct Random Topologies for Testing Smart Grid Communication and Control Networks," IEEE Trans. Smart Grid, **1**(1), pp. 28–39.