Article

Not peer-reviewed version

Analyzing the Robustness of Complex Networks with Attack Success Rate

Fangqun Yang and Yisong Wang *

Posted Date: 9 October 2023

doi: 10.20944/preprints202310.0451.v1

Keywords: complex network; robustness; quasi-Monte Carlo; attack success rate

Article

Analyzing the Robustness of Complex Networks with Attack Success Rate

Fangqun Yang and Yisong Wang *

State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; gs.yangfq21@gzu.edu.cn (F.Y)

* Correspondence: yswang@gzu.edu.cn

Abstract: Analyzing network robustness against random failures or malicious attacks is a critical research issue in network science as it helps to enhance the robustness of beneficial networks or efficiently disintegrate harmful networks. Most previous studies commonly neglect the impact of the attack success rate (ASR) and assume that attacks on the network will always be successful. However, in real-world scenarios, an attack may not always succeed. Therefore, this paper proposes a novel robustness measure called RASR, which utilizes mathematical expectations to assess network robustness when considering the ASR of each node. To efficiently compute the RASR for large-scale networks, a parallel algorithm named PRQMC is presented, which leverages randomized quasi-Monte Carlo integration to approximate the RASR with a faster convergence rate. Additionally, a new attack strategy named HBnnsAGP is introduced to better assess the lower bound of network RASR. Finally, the experimental results on 6 representative real-world complex networks demonstrate the effectiveness of the proposed methods compared with the state-of-the-art baselines.

Keywords: complex network; robustness; quasi-Monte Carlo; attack success rate

1. Introduction

Complex networks can effectively represent many real-world networks, such as the Internet, social networks, power grids, and so on. Most networks are beneficial to people and bring many positive effects. However, some networks also have negative effects, with the most important examples being terrorism and disease transmission networks [1,2]. Whether beneficial or harmful, these networks substantially influence the functioning and development of our society. In recent decades, the study of diverse complex networks has gained significant attention from researchers across various fields such as computer science, statistical physics, systems engineering, and applied mathematics [3–7]. One hot topic point in these studies is the error and attack tolerance of complex networks [8–16], a concept referred to as *robustness* within the context of this paper.

The robustness of a network refers to its ability to keep functioning when some of its components, such as nodes or edges, malfunction due to random failures or malicious attacks [12,17,18]. The study of network robustness is valuable from two main perspectives. Firstly, the failure of components can lead to the breakdown of beneficial networks and result in significant economic losses. A typical example is the Northeast blackout of 2003 [19,20]. Analyzing network robustness aids in developing methods to enhance it. On the other hand, for harmful networks, such as terrorist networks [21] or COVID-19 transmission networks [22], analyzing their robustness assists in developing effective attack strategies to dismantle them. Therefore, analyzing network robustness is of great importance.

To analyze the robustness of the network, it is necessary to choose a suitable metric to evaluate how robust a network is. Since almost all network applications are typically designed to operate in a connected environment [23], network connectivity is selected as the primary indicator to assess network robustness in this study.

The robustness of a network depends not only on its structural features but also on the mechanisms of random failures or malicious attacks. In random failures, nodes or edges are attacked with equal probability, while malicious attacks target nodes or edges in decreasing order of their importance.

Typically random failures are less severe than malicious attacks [24,25]. Evaluating the impacts of node or edge removal using various malicious attack strategies is a crucial approach to analyzing network robustness. Determining the lower bound of network robustness is critical as it allows for analysis of network robustness under worst-case scenarios, identification of the most vulnerable components, and development of robustness improvement methods. An effective approach to addressing this issue involves identifying an optimal attack strategy that inflicts maximum damage on the network [26].

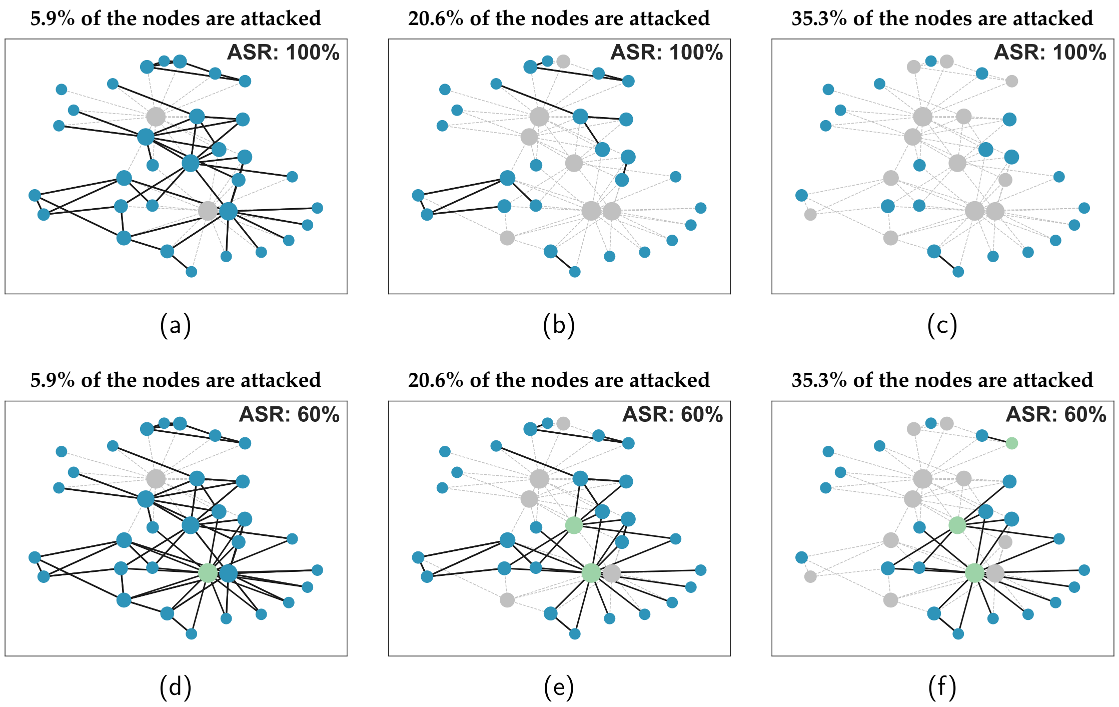
Extensive research has been conducted on the robustness of complex networks. Albert et al. [8] studied the robustness of scale-free networks and found that while these networks are robust to random failures, they are extremely vulnerable to malicious attacks. Iyer et al. [9] conducted a systematic examination of the robustness of complex networks by employing simultaneous and sequential targeted attacks based on various centrality measures such as degree, betweenness, closeness, and eigenvector centrality. Fan et al. [10] proposed a deep reinforcement learning algorithm, FINDER, to effectively identify critical network nodes. Wang et al. [11] introduced region centrality and proposed an efficient network disintegration strategy based on this concept, which combines topological properties and geographic structure in complex networks. Ma et al. [12] conducted a study on the robustness of complex networks against incomplete information. They employed link prediction methods to restore missing network topology information and identify critical nodes. Lou et al. [14] introduced LFR-CNN, a CNN-based approach that utilizes learning feature representation for predicting network robustness, which exhibits excellent predictive performance notably smaller prediction errors.

However, the aforementioned research generally assumes that attacks on the network will always be successful, neglecting the important factor of attack success rate (ASR). In fact, an attack may not succeed in real-world scenarios. For example, even if the enemy forces launch an attack on a target within a military communication network, there is no guarantee of successfully destroying it. Figure 1 illustrates the main process of network disintegration under varying ASR. Moreover, selecting an optimum attack strategy that can lead to maximal destructiveness to the network is challenging due to the NP-hard nature of this problem [10]. Existing methods often encounter difficulties in achieving a desirable balance between effectiveness and computational efficiency.

Therefore, the purpose of this paper is to analyze network robustness when considering ASR under an optimal attack strategy. To achieve this purpose, a novel robustness measure called Robustness-ASR (RASR) is introduced, which utilizes mathematical expectations to evaluate network robustness when considering ASR. In addition, an efficient algorithm called PRQMC is proposed to calculate the RASR for large-scale networks. Furthermore, to assess the lower bound of network RASR, a new attack strategy, named HBnnsAGP, is proposed. The main contributions of this study are as follows:

- We introduce and define a novel robustness measure called RASR, which utilizes mathematical expectations to assess network robustness when considering the ASR of each node.
- To efficiently calculate the RASR for large-scale networks, we propose the PRQMC algorithm. PRQMC leverages randomized quasi-Monte Carlo (QMC) integration to approximate the RASR with a faster convergence rate and utilizes parallelization to speed up the calculation.
- To assess the lower bound of network RASR, we present a new attack strategy, named HBnnsAGP. In HBnnsAGP, a novel centrality measure called BCnns is proposed to quantify the importance of a node.
- The experimental results on 6 representative real-world networks demonstrate the effectiveness of the proposed methods compared with the baselines.

The rest of this paper is organized as follows. Section 2 provides an introduction to the preliminaries, including classical centrality measures, traditional network robustness measures, and the principles of Monte Carlo (MC) and QMC integration. Section 3 presents the proposed methods for analyzing network robustness when considering ASR, including the RASR, the PRQMC algorithm, and the HBnnsAGP attack strategy. The experiments and results are demonstrated in Section 4. Finally, Section 5 concludes the paper.



5.9% of the nodes are attacked
ASR: 100%
(a)
20.6% of the nodes are attacked
ASR: 100%
(b)
35.3% of the nodes are attacked
ASR: 100%
(c)
5.9% of the nodes are attacked
ASR: 60%
(d)
20.6% of the nodes are attacked
ASR: 60%
(e)
35.3% of the nodes are attacked
ASR: 60%
(f)

Figure 1. An example of network disintegration process under different ASR. Gray nodes indicate successful attacks, green nodes represent unsuccessful attacks, and blue nodes denote unattacked nodes.

2. Preliminaries

A complex network can be modeled as an unweighted undirected graph $G = (V, E)$, where $V(|V| = N)$ and $E(|E| = M)$ represent the set of nodes and the set of edges in the network G, respectively. The network G can be also represented as an adjacency matrix $A = (a_{ij})_{N\times N}$, if node i and node j are connected, $a_{ij} = 1$, otherwise $a_{ij} = 0$.

2.1. Centrality Measures

The concept of a centrality measure attempts to quantify how important a node is [27]. Here we introduce two classical centrality measures: degree centrality and betweenness centrality.

2.1.1. Degree centrality (DC)

DC is the simplest measure of centrality. The DC of a node is defined by its degree, that is, its number of edges. The DC is formally defined as follows.

Definition 1. *Given a network $G = (V, E)$, $A = (a_{ij})_{N\times N}$ is the adjacency matrix of the network G. The DC of node i is defined as:*

$$DC(i) = \sum_{j\in V} a_{ij}. \tag{1}$$

The DC is frequently a reliable and effective measure of a node's importance. A higher DC value typically signifies a more critical node.

2.1.2. Betweenness centrality (BC)

BC quantifies the number of shortest paths passing through a particular node in a network[28]. BC characterizes the extent to which a node acts as a mediator among all other nodes in a network[27].

Nodes that lie on numerous shortest paths are likely to play a crucial role in information transmission, exhibiting higher BC values. The BC is defined as follows.

Definition 2. *Given a network $G = (V, E)$. The BC of node v in G is defined as:*

$$BC(v) = \sum_{s,t \in V} \frac{\sigma(s, t \mid v)}{\sigma(s, t)}, \tag{2}$$

where, $v \in V$, $\sigma(s, t)$ is the total number of shortest paths from node s to node t and $\sigma(s, t \mid v)$ is the number of those paths that pass through node v. $\sigma(s, t) = 1$, if $s = t$. $\sigma(s, t \mid v) = 0$, if $v \in s, t$.

2.2. Accumulated Normalized Connectivity

Traditionally, network robustness has been evaluated by calculating the size of the giant connected component (GCC) after the network has endured attacks. The Accumulated Normalized Connectivity (ANC), also known as R, is a well-known measure of network robustness for node attacks [10,17,29]. The ANC is defined as follows.

Definition 3. *For a network $G = (V, E)$, $|V| = N$. Given an attack sequence of nodes $(v_1, v_2, \ldots, v_N)$, where $v_i \in V$ indicates the ith node to be attacked, the ANC of G under this attack sequence is defined as:*

$$ANC(v_1, v_2, \ldots, v_N) = \frac{1}{N} \sum_{k=1}^{N} \frac{\sigma_{gcc}(G \backslash \{v_1, v_2, \ldots, v_k\})}{\sigma_{gcc}(G)}, \tag{3}$$

here, $\sigma_{gcc}(G \backslash \{v_1, v_2, \ldots, v_k\})$ is the size of the GCC of the residual network after the sequential removal of nodes from the set $\{v_1, v_2, \ldots, v_k\}$ in G, and $\sigma_{gcc}(G)$ the initial size of the GCC of G before any nodes are removed. The normalization factor $\frac{1}{N}$ ensures that the robustness of networks with different sizes can be compared.

A larger ANC value indicates a higher level of network robustness against attacks. Additionally, the ANC can be used to assess the destructiveness of attacks, lower ANC values correspond to more destructive attack strategies. The ANC value can be viewed as an estimate of the area beneath the ANC curve, which is plotted with the horizontal axis as k/N and the vertical axis as $\sigma_{gcc}(G \backslash \{v_1, v_2, \ldots, v_k\})/\sigma_{gcc}(G)$.

2.3. Monte Carlo Integration

Monte Carlo (MC) integration is a numerical technique that is particularly useful for higher-dimensional integrals[30]. Caflisch[31] provides a comprehensive review of this method. The integral of a Lebesgue integrable function $f(\boldsymbol{X})$ can be expressed as the average or expectation of the function evaluated at random locations. Considering $\boldsymbol{X}$ as a random variable uniformly distributed on the one-dimensional unit interval $[0, 1]$, the integration of $f(\boldsymbol{X})$ over this interval can be represented as follows:

$$I[f] = E[f(\boldsymbol{X})] = \int_{[0,1]} f(\boldsymbol{X}) dP(\boldsymbol{X}), \tag{4}$$

in which $P(\boldsymbol{X})$ is the probability measure of $\boldsymbol{X}$ on the interval $[0, 1]$, then

$$dP(\boldsymbol{X}) = d\boldsymbol{X}, \tag{5}$$

therefore

$$I[f] = E[f(\boldsymbol{X})] = \int_{[0,1]} f(\boldsymbol{X}) d\boldsymbol{X}. \tag{6}$$

Similarly, for an integral on the unit hypercube $[0,1]^N$ in N dimensions,

$$I[f] = E[f(\boldsymbol{X})] = \int_{[0,1]^N} f(\boldsymbol{X}) d\boldsymbol{X}, \tag{7}$$

in which $\boldsymbol{X} = (x_1, x_2, \ldots, x_N)$ is a uniformly distributed vector in $[0,1]^N$, where $x_i \in [0,1], i \in \{1, 2, \ldots, N\}$. Given that the hyper-volume of $[0,1]^N$ is equal to 1, thus $[0,1]^N$ can be viewed as the total probability space.

The MC integration method approximates definite integrals utilizing random sampling. It draws K uniform samples from $[0,1]^N$, in turn generating a points set $\{\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_K}\}$. The empirical approximation of the integral $I[f]$ is then procured by computing the mean of the K sample outcomes $f(\boldsymbol{X}_i)$, which can be expressed as follows:

$$I[f] \approx I_K[f] = \frac{1}{K} \sum_{i=1}^{K} f(\boldsymbol{X}_i). \tag{8}$$

According to the Strong Law of Large Numbers [32], this approximation is convergent with probability 1, that is,

$$\lim_{K \to \infty} P\left(|I_K[f] - I[f]| = 0\right) = 1. \tag{9}$$

Figure 2 illustrates the application of the MC integration method in approximating definite integrals over a one-dimensional unit interval. As shown in Figure 2a, MC integration approximates the area under the curve of the integral by summing the areas of the bars corresponding to the sampled points. The bars are rearranged sequentially to avoid overlap on the $\boldsymbol{X}$-axis, as shown in Figure 2b.

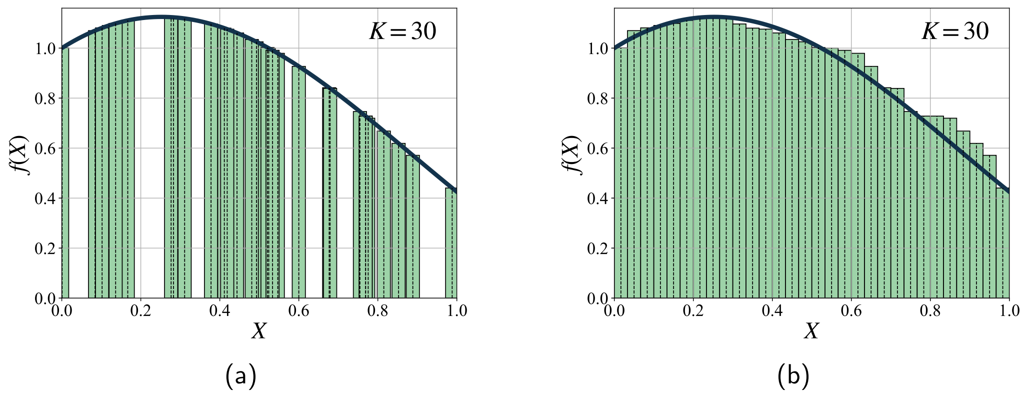


Figure 2. An example of MC integration method for approximating a definite integral over a one-dimensional unit interval. (**a**) illustrates the approximation of the integral by summing the areas of bars that correspond to the sampled points. Each bar's height represents the value of $f(\boldsymbol{X})$ at $\boldsymbol{X}_i$ and its width is $1/K$, where K denotes the total number of samples. (**b**) demonstrates the sequential rearrangement of the bars to prevent overlapping on the X-axis, ensuring a clear visualization of the areas.

The error of MC integration is

$$\varepsilon_K = |I_K[f] - I[f]|\,. \tag{10}$$

By the Central Limit Theorem [32], for any a, b where $a < b$, we have

$$\lim_{K \to \infty} P(a < \frac{\varepsilon_K}{\sigma / \sqrt{K}} < b) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = P(a < v < b), \tag{11}$$

where v is a standard normal random variable and σ is the square root of the variance of f, given by

$$\sigma = (\int_{[0,1]^N} (f(\boldsymbol{X}) - I[f])^2 d\boldsymbol{X})^{1/2}. \tag{12}$$

When K is sufficiently large, we have

$$\varepsilon_K \approx \sigma K^{-1/2} v. \tag{13}$$

This implies that the order of error convergence rate of the MC integration is $O(K^{-1/2})$ [33], which means that the accuracy of the integral error decreases at a rate proportional to the total number of samples K increases. That is, "an additional factor of 4 increase in computational effort only provides an additional factor of 2 improvements in accuracy" [31].

In practical applications, the MC integration method draws K uniform samples from an N-dimensional pseudo-random sequence (PRS) generated by a computer to obtain the points set $\{\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_K}\}$.

2.4. Quasi-Monte Carlo Integration

The quasi-Monte Carlo (QMC) integration is a method of numerical integration that operates in the same way as MC integration, but instead uses a deterministic low-discrepancy sequence (LDS) [34] to approximate the integral. The advantage of using LDSs is a faster rate of convergence. QMC integration has a rate of convergence close to $O(K^{-1})$, which is much faster than the rate for the MC integration, $O(K^{-1/2})$ [35]

Using the QMC integration method for approximating definite integrals is similar to the MC integration method. This can be expressed as:

$$I[f] = \int_{[0,1]^N} f(\boldsymbol{X}) d\boldsymbol{X} \approx \frac{1}{K} \sum_{i=1}^{K} f(\boldsymbol{Y}_i), \tag{14}$$

where $\{\boldsymbol{Y_1}, \boldsymbol{Y_2}, \ldots, \boldsymbol{Y_K}\}$ is a points set obtained by combining the first K points from an N-dimensional LDS. Each $\boldsymbol{Y_i}$ is an N-dimensional point, with $\boldsymbol{Y_i} = (y_1^{\{i\}}, y_2^{\{i\}}, \ldots, y_N^{\{i\}})$ for $i \in \{1, 2, \ldots, K\}$, and $y_j^{\{i\}} \in [0,1]$ for $j \in \{1, 2, \ldots, N\}$.

The error order of the QMC integration can be determined by the Koksma-Hlawka inequality [36,37], that is,

$$\varepsilon_K = \left| \int_{[0,1]^N} f(\boldsymbol{X}) d\boldsymbol{X} - \frac{1}{K} \sum_{i=1}^{K} f(\boldsymbol{Y}_i) \right| < V(f) D_K^*, \tag{15}$$

where $V(f)$ is the Hardy–Krause variation of the function f, D_K^* is the star discrepancy of $\{\boldsymbol{Y_1}, \boldsymbol{Y_2}, \ldots, \boldsymbol{Y_K}\}$, and is defined as:

$$D_K^* = \sup_{Q \subset [0,1]^N} \left| \frac{M(\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_K)}{K} - \lambda_N(Q) \right|, \tag{16}$$

where $M(\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_K)$ is the number of points in $\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_K\}$ inside the region Q, and $\lambda_N(Q)$ is the Lebesgue measure of region Q in the unit hypercube $[0,1]^N$. For more detailed information, please refer to [31].

For an N-dimensional LDS comprising K points, the star discrepancy of the sequence is $O(K^{-1}(\log K)^N)$. Consequently, for a function F with $V(F) < \infty$, a QMC approximation based on this sequence yields a worst-case error bound in Equation (28) converging at a rate of $O(K^{-1}(\log K)^N)$ [38]. Since $\log K \ll K$, the QMC integration convergence rate approaches $O(K^{-1})$ for low-dimensional cases [33], which is asymptotically superior to MC.

Figure 3 illustrates the clear differences between MC and QMC integration methods. The subfigures provide a visual representation of their respective point distributions and demonstrate their application for approximating definite integrals over a one-dimensional unit interval. The points generated from an LDS exhibit greater uniformity than the points generated by a PRS. Consequently, with the same number of sampling points, LDS has the ability to uniformly fill the integration space, resulting in a faster convergence rate.

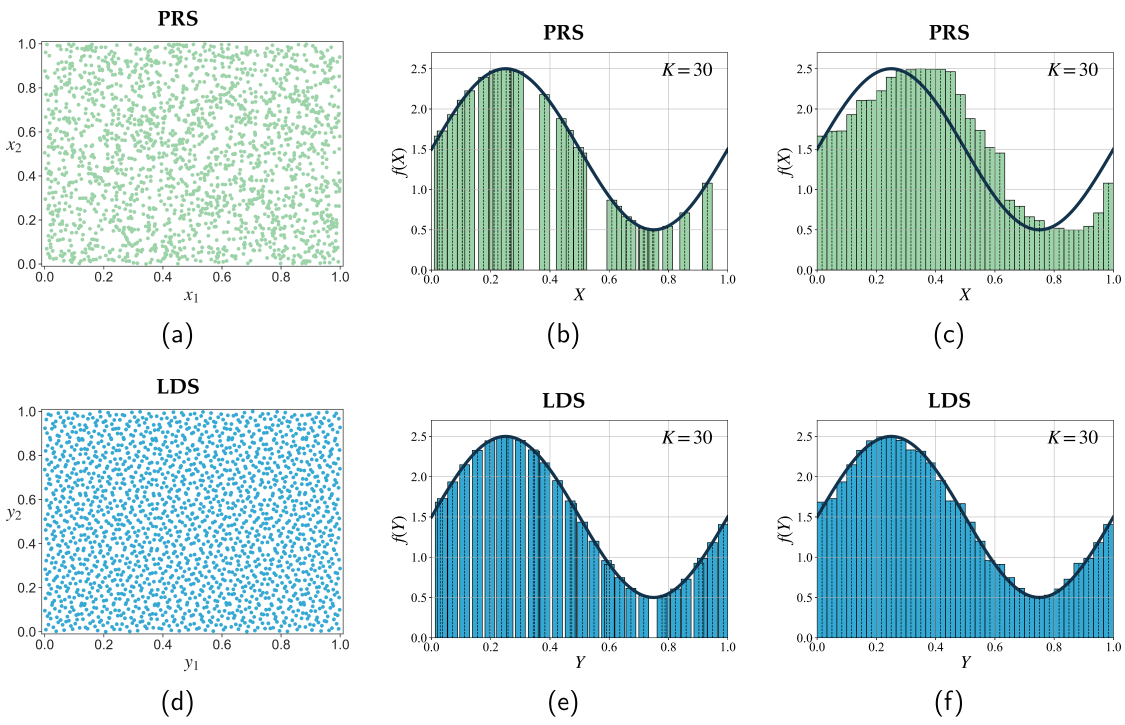


Figure 3. A comparison of MC and QMC integration methods. (**a**) and (**d**) show the two-dimensional projections of a PRS and an LDS (a Sobol sequence) respectively. (**b**) and (**c**) depict the MC integration for approximating a definite integral over a one-dimensional unit interval, while (**e**) and (**f**) present the QMC integration for approximating a definite integral over a one-dimensional unit interval.

3. Methods

In this section, we will first introduce the major problem we focus on in this paper. Then, we give the details of the proposed methods for analyzing network robustness when considering ASR, including the RASR, the PRQMC algorithm, and the HBnnsAGP attack strategy.

3.1. Problem Formalization

Typically, it is assumed that removing a node will also remove all of its connected edges. Therefore, in this paper, we only consider node attack strategies.

For a network $G = (V, E)$, $|V| = N$. A node attack strategy can be represented as a sequence $\boldsymbol{Seq} = (v_1, v_2, \ldots, v_N)$, where $v_i \in V$ indicates the ith node to be attacked. Given a predefined metric $\Phi(\boldsymbol{Seq})$ to measure network robustness against attacks. The primary goal is to evaluate the lower bound of network robustness. Therefore, the objective is to minimize $\Phi(\boldsymbol{Seq})$, as presented below:

$$Minimize \;\; \Phi(\boldsymbol{Seq}). \tag{17}$$

To achieve this objective, it is crucial to determine the optimal node attack strategy that will minimize the $\Phi(\boldsymbol{Seq})$.

3.2. The Proposed Robustness Measure RASR

The ANC, as defined in Definition 3, does not consider the ASR, or it is a special case where the ASR of each node is 100%. To this end, the proposed robustness measure RASR utilizes mathematical expectations to assess network robustness when considering ASR. Before introducing the RASR, we will first present a weighted ANC (named ANCw), which takes into account both the state of the attack sequence state and the associated attack cost.

For a network $G = (V, E)$ with N nodes, $\boldsymbol{Seq} = (v_1, v_2, \ldots, v_N)$ is an attack sequence, where $v_i \in V$. The state of $\boldsymbol{Seq}$ is denoted as a random variable $\boldsymbol{S} = (s_{v_1}, s_{v_2}, \ldots, s_{v_N})$, where

$$s_{v_i} = \begin{cases} T, & \text{if the attack on } v_i \text{ succeeded} \\ F, & \text{otherwise} \end{cases}. \tag{18}$$

Then, the ANCw is defined as follows.

Definition 4. *The ANCw of G under an attack sequence* $\boldsymbol{Seq}$ *is defined as:*

$$ANCw(\boldsymbol{Seq}, \boldsymbol{S}) = \frac{1}{\text{N+1}} \sum_{k=0}^{N} \frac{\sigma_{gcc}(G \backslash \{v_i | s_{v_i} = T, i = 1, 2, \cdots, k\})}{\sigma_{gcc}(G)} \varphi(v_k), \tag{19}$$

here σ_{gcc} is the same as defined in Definition 3. When k = 0, it indicates that no nodes have been attacked. $\varphi(v_k)$ is a weighted function, that is,

$$\varphi(v_k) = \begin{cases} 0, & \textit{if } v_k \textit{ is an isolated node} \\ 1, & \textit{otherwise} \end{cases}. \tag{20}$$

There are two main reasons for using the weighted function $\varphi(v_k)$. Firstly, it is important for an attacker to choose an optimal attack strategy at a minimum attack cost to efficiently disintegrate the network [11,26]. Secondly, as illustrated in Figure 1, with an increased number of nodes removed, the network will eventually fragment into isolated nodes, thereby losing its functionality as a network. Therefore, this paper sets the attack cost of an isolated node to 0.

Let $\boldsymbol{P_v} = (p_{v_1}, p_{v_2}, \ldots, p_{v_N})$ represent the ASR of each node corresponding to $\boldsymbol{Seq}$, where p_{v_i} represents the ASR of node v_i. Assuming that attacks on different nodes are independent, then the probability of $\boldsymbol{S}$ is

$$p(\boldsymbol{S}) = \prod_{i=1}^{N} p(s_{v_i}), \tag{21}$$

where

$$p(s_{v_i}) = \begin{cases} p_{v_i}, & \text{if } s_{v_i} = T \\ 1 - p_{v_i}, & \text{otherwise} \end{cases}. \tag{22}$$

Based on the above formulas, the proposed RASR can defined as follows.

Definition 5. *Considering the ASR of each node, the robustness of a network G against an attack sequence* $\boldsymbol{Seq}$ *can be quantified by the RASR, which is defined as:*

$$RASR = E(ANCw(\boldsymbol{Seq}, \boldsymbol{S})) = \sum_{\boldsymbol{S} \in \Omega} ANCw(\boldsymbol{Seq}, \boldsymbol{S}) p(\boldsymbol{S}), \tag{23}$$

where $\boldsymbol{S}$ is a random variable representing the state of $\boldsymbol{Seq}$, Ω is the sample space of $\boldsymbol{S}$, $E(ANCw(\boldsymbol{Seq}, \boldsymbol{S}))$ is the expectation of the ANCw.

In theory, the value of RASR can be calculated using Equation (23) once all the samples of S are obtained in the sample space Ω. However, it confronts “the curse of dimensionality" [39] when applied to networks with a large number of nodes. In such cases, the size of Ω grows exponentially to 2^N. As a result, the analytical approach becomes infeasible when N is significantly large.

3.3. The Proposed PRQMC Algorithm

To efficiently calculate the RASR for large-scale networks, the PRQMC algorithm is proposed, which leverages randomized QMC integration to approximate the RASR with a faster convergence rate and utilizes parallelization techniques to speed up the calculation. In the following, we first introduce the RASR calculation model based on QMC integration and then give the PRQMC algorithm.

3.3.1. RASR Calculation Model Based on QMC Integration

The RASR of a network G, as defined in Definition 5, can be expressed using Lebesgue integration based on the principle of MC integration (see Section 2), that is,

$$RASR = E(ANCw(\boldsymbol{Seq}, \boldsymbol{S})) = \int_{\Omega} ANCw(\boldsymbol{Seq}, \boldsymbol{S}) dP(\boldsymbol{S}), \tag{24}$$

where $\boldsymbol{S} = (s_{v_1}, s_{v_2}, \ldots, s_{v_N})$ denotes a random variable representing the state of an attacking sequence $\boldsymbol{Seq}$, Ω is the sample space of $\boldsymbol{S}$, $P(\boldsymbol{S})$ is the probability measure of $\boldsymbol{S}$.

Let $\boldsymbol{P_v} = (p_{v_1}, p_{v_2}, \ldots, p_{v_N})$ represent the ASR of each node corresponding to $\boldsymbol{Seq}$, $\boldsymbol{X} = (x_1, x_2, \ldots x_N)$ is a uniformly distributed vector in $[0,1]^N$, where $x_i \in [0,1], i \in \{1, 2, \ldots, N\}$. Then, $\boldsymbol{S} = (s_{v_1}, s_{v_2}, \ldots, s_{v_N})$ can be represented as follows:

$$\boldsymbol{S} = G(\boldsymbol{X}), \tag{25}$$

where

$$s_{v_i} = G_i(x_i) = \begin{cases} T, & \text{if } x_i \leq p_{v_i} \\ F, & \text{otherwise} \end{cases}, i \in \{1, 2, \ldots, N\}. \tag{26}$$

When the $\boldsymbol{Seq}$ is determined, the $ANCw(\boldsymbol{Seq}, \boldsymbol{S})$ can be represented as a function of $\boldsymbol{X}$, that is,

$$F(\boldsymbol{X}) = ANCw(\boldsymbol{Seq}, G(\boldsymbol{X})) = ANCw(\boldsymbol{Seq}, \boldsymbol{S}). \tag{27}$$

By substituting Equation (27) into Equation (24) and transforming the integral space from Ω to $[0,1]^N$, we obtain the following expression for RASR:

$$RASR = E[F(\boldsymbol{X})] = \int_{[0,1]^N} F(\boldsymbol{X}) dP(\boldsymbol{S}). \tag{28}$$

This equation represents the integration of $F(\boldsymbol{X})$ with respect to the probability measure $P(\boldsymbol{S})$ over the N-dimensional unit hypercube $[0,1]^N$.

For the given network G, the sample space Ω has a size of 2^N. Let the state of $\boldsymbol{Seq}$ be $\boldsymbol{S}_i$, where $i \in \{1, 2, 3, \ldots, 2^N\}$. Based on $\boldsymbol{P_v}$, the unit hypercube $[0,1]^N$ can be divided into 2^N regions denoted by Q_i, where region Q_i corresponds to state $\boldsymbol{S}_i$, $i \in \{1, 2, \ldots 2^N\}$. Figure 4 illustrates this process for the case when $N = 2$. Then, the integral in Equation (28) can be transformed into:

$$\int_{[0,1]^N} F(\boldsymbol{X}) dP(\boldsymbol{S}) = \sum_{i=1}^{2^N} \int_{Q_i} F(\boldsymbol{X}^i) dP(\boldsymbol{S}_i), \tag{29}$$

where $\boldsymbol{X}^{\{i\}}$ is a vector uniformly distributed within region Q_i.

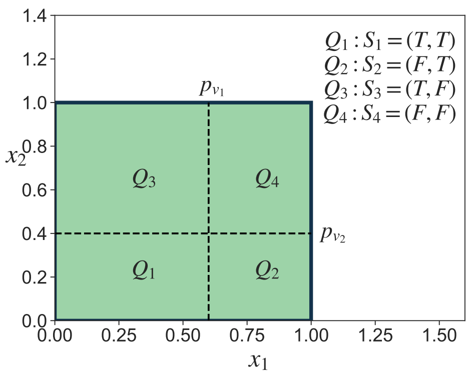


Figure 4. An example to illustrate the division of the unit hypercube, where $N = 2$ and $P_v = (p_{v_1}, p_{v_2})$. The unit hypercube $[0,1]^2$ is divided into 4 regions, namely Q_1, Q_2, Q_3, Q_4, where each region corresponds to a state of Seq, denoted by S_1, S_2, S_3, S_4.

The Lebesgue measure of region Q_i in $[0,1]^N$, denoted by $\lambda_N(Q_i)$, is equivalent to the probability measure of $\boldsymbol{S_i}$, denoted as $P(\boldsymbol{S}_i)$. Based on the principle of MC integration, we have:

$$\sum_{i=1}^{2^N} \int_{Q_i} F(\boldsymbol{X}^{\{i\}}) dP(\boldsymbol{S}_i) = \sum_{i=1}^{2^N} \int_{Q_i} F(\boldsymbol{X}^{\{i\}}) d\boldsymbol{X}^{\{i\}} = \int_{[0,1]^N} F(\boldsymbol{X}) d\boldsymbol{X}. \tag{30}$$

Combining Equation (28), Equation (29), and Equation (30), we obtain:

$$RASR = E[F(\boldsymbol{X})] = \int_{[0,1]^N} F(\boldsymbol{X}) d\boldsymbol{X}. \tag{31}$$

By referencing Equation (14) and Equation (31), the RASR of a network can be approximated using the QMC integration method. The approximation of RASR, denoted by $\widehat{R}$, is defined as follows.

Definition 6. *Consider a network $G = (V, E)$ with N nodes. Suppose a sequence of nodes $\boldsymbol{Seq} = (v_1, v_2, \ldots, v_N)$ is targeted for attack, $\boldsymbol{P_v} = (p_{v_1}, p_{v_2}, \ldots, p_{v_N})$ signifies the ASR of each node. The RASR of the network G can be approximated by $\widehat{R}$, which is defined as:*

$$\widehat{R} = \frac{1}{K} \sum_{i=1}^{K} F(\boldsymbol{Y}_i) \approx RASR. \tag{32}$$

Here, $\{\boldsymbol{Y_1}, \boldsymbol{Y_2}, \ldots, \boldsymbol{Y_K}\}$, as specified in Equation (14), represents a set of points obtained from an N-dimensional LDS. K is the total number of samples. The function $F(\boldsymbol{X})$ is defined in Equation (27).

The error bound of the QMC integral is determined by the star discrepancy of the chosen LDS, making the selection of LDSs important for improving the accuracy of approximations. Two frequently used LDSs are the Halton sequence and the Sobol sequence [40]. In this research, the Sobol sequence is adopted, as it demonstrates better performance in higher dimensions compared to the Halton sequence [41].

3.3.2. Parallel Randomized QMC (PRQMC) Algorithm

Despite the faster convergence rate of the QMC integration method compared to MC integration, it still necessitates a large number of samples to calculate the average value. Furthermore, the calculation of function $ANCw(\boldsymbol{Seq}, \boldsymbol{S})$, typically done through attack simulations, demands considerable computational resources, especially for large-scale networks [42]. Consequently, the computational process of obtaining $\widehat{R}$ for large-scale networks remains time-consuming. Additionally, due to the deterministic nature of the LDS, the QMC integration method can be seen as a deterministic algorithm, thus presenting challenges in assessing the reliability of numerical integration results and potentially leading to being stuck in local optima. In light of these issues, the PRQMC algorithm capitalizes on the benefits of the Randomized QMC method and parallelization.

The PRQMC algorithm improves computational efficiency through parallelization. This is because the computational cost of sampling the attack sequence's state $\boldsymbol{S}$ is significantly lower than that of computing the function $ANCw(\boldsymbol{Seq}, \boldsymbol{S})$. Therefore, by initially sampling the attack sequence's state $\boldsymbol{S}$ and obtaining a sufficient number of samples, it is possible to calculate the $\widehat{R}$ by parallelizing the computation of the function $ANCw(\boldsymbol{Seq}, \boldsymbol{S})$ with various samples. This approach effectively accelerates the calculation process by distributing the task across multiple processors or computing nodes.

Additionally, the PRQMC algorithm enhances randomness by randomly sampling points from the LDS, providing unbiased estimation and improved variance reduction capabilities. This is particularly advantageous in high-dimensional problems, where RQMC often outperforms QMC in terms of accuracy and efficiency [43].

The procedure of the PRQMC algorithm is presented in Algorithm 1, which consists of two main steps: "sampling state" and "paralleling stage". In the sampling stage, we first randomly sample K points $\{\boldsymbol{Y_1}, \boldsymbol{Y_2}, \ldots, \boldsymbol{Y_K}\}$ from an N-dimensional Sobol sequence, then determine K states of the attack sequence, $\{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K\}$, by comparing the values of each dimension of the sampled points with the ASR of each node. In the paralleling stag, we parallelize the computation of the function $ANCw(\boldsymbol{Seq}, \boldsymbol{S}_i)$, then obtain $\widehat{R}$ by calculating the average value of $ANCw(\boldsymbol{Seq}, \boldsymbol{S}_i)$.

Algorithm 1 PRQMC($G, \boldsymbol{Seq}, \boldsymbol{P}, K$)

```
Input:  $G = (V, E)$: a network with $N$ nodes,
        $\boldsymbol{Seq} = (v_1, v_2, \ldots, v_N)$: an attacking sequence of $G$,
        $\boldsymbol{P} = (p_{v_1}, p_{v_2}, \ldots, p_{v_N})$: ASR of each node in $\boldsymbol{Seq}$,
        $K$: the total number of samples.
Output: $\widehat{R}$: the approximate value of the RASR of $G$.
    Step1: Sampling stage.
  1 sampling $K$ points $\{\boldsymbol{Y_1}, \boldsymbol{Y_2}, \ldots, \boldsymbol{Y_K}\}$ randomly from an $N$-dimensional Sobol sequence,
    where $\boldsymbol{Y_i} = (y_1^{\{i\}}, y_2^{\{i\}}, \ldots, y_N^{\{i\}})$ for $i \in \{1, 2, \ldots, K\}$;
  2 let $State = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K\}$, where $\boldsymbol{S_i} = (s_{v_1}^{\{i\}}, s_{v_2}^{\{i\}}, \ldots, s_{v_N}^{\{i\}})$ for $i \in \{1, 2, \ldots, K\}$;
  3 for $i = 1$ to $K$ do
  4     $s_{v_j}^{\{i\}} = \begin{cases} T, y_j^{\{i\}} \le p_{v_j} \\ F, y_j^{\{i\}} > p_{v_j} \end{cases}, j \in \{1, 2, \cdots, N\}$;
    Step2: Paralleling stage.
  5 let $Res = \{\widehat{R}_1, \widehat{R}_2, \ldots, \widehat{R}_K\}$;
  6 parallel for all $\boldsymbol{S_i} \in State$ do
  7     $\widehat{R}_i = ANCw(\boldsymbol{Seq}, \boldsymbol{S}_i)$;
  8 $\widehat{R} = \frac{1}{K} \sum_{i=1}^{K} \widehat{R}_i$;
  9 return $\widehat{R}$.
```

3.4. The Proposed HBnnsAGP Attack Strategy

To assess the lower bound of network RASR, a new attack strategy called the High BCnns Adaptive GCC-Priority (HBnnsAGP) is presented. In HBnnsAGP, a novel centrality measure called BCnns is proposed to quantify the significance of a node, and GCC-priority attack strategy is utilized to improve attack effectiveness. Algorithm 2 describes the procedure of HBnnsAGP, which contains two steps: "obtaining the first part of $\boldsymbol{Seq}$" and "obtaining the second part of $\boldsymbol{Seq}$". In the first step, the algorithm obtains the first part of the attack sequence by iteratively removing the node with the highest BCnns in GCC and recalculating BCnns for the remaining nodes until only isolated nodes remain in the residual network. In the second step, the algorithm arranges these isolated nodes in descending order according to their DC values in the initial network to obtain the second part of the attack sequence. This procedure is aimed at improving the effectiveness of attacks when the ASR is below 100%. It is important to note that isolated nodes when the ASR is 100% may no longer remain isolated, as depicted in, as shown in Figure 1. Additionally, previous research has shown that there is minimal difference in destructiveness between simultaneous attacks and sequential attacks based on DC [9]. Therefore, by sorting these isolated nodes in descending order based on their DC values from the initial network (similar to the approach used in simultaneous attacks), the second step further improves the effectiveness of attacks when the ASR is less than 100%.

In the following, we first introduce the BCnns and then give the GCC-priority attack strategy.

Algorithm 2 HBnnsAGP(G, N_1, N_2)

```
Input: $G = (V, E)$: a network with $N$ nodes,
       $N_1$ and $N_2$: sampling numbers.
Output: $\boldsymbol{Seq}$: an attacking sequence of $G$.
 1 let $\boldsymbol{Seq}$ be an empty list;
 2 $G_0 = (V_0, E_0) \leftarrow G$;
     Step1: Obtaining the first part of $\boldsymbol{Seq}$.
 3 while $E \neq \emptyset$ do
 4     $G_c = (V_c, E_c) \leftarrow$ get the $GCC$ of $G$;
 5     $S, T \leftarrow$SelectST$(G_c, N_1, N_2)$
 6     for all $v \in V_c$ do
 7         $BC_{nns}(v) = \sum_{s \in S, t \in T} \frac{\sigma(s,t|v)}{\sigma(s,t)}$;
 8     $attack\_node \leftarrow \arg\max_{v \in V_c}(BC_{nns}(v))$;
 9     append $attack\_node$ to the end of $\boldsymbol{Seq}$;
10     $G \leftarrow G \setminus \{attack\_node\}$;
     Step2: Obtaining the second part of $\boldsymbol{Seq}$.
11 $V_r \leftarrow (V_0 \setminus \boldsymbol{Seq})$;
12 sort the nodes of $V_r$ decreasing by $DC$ values from $G_0$;
13 $\boldsymbol{Seq} \leftarrow \boldsymbol{Seq} + V_r$;
14 return $\boldsymbol{Seq}$.
```

3.4.1. Non-central Nodes Sampling Betweenness Centrality (BCnns)

Contrasted with BC (see Definition 2), which evaluates a node's role as a mediator in the network based on the count of shortest paths it traverses for all node pairs. BCnns quantifies the importance of nodes acting as bridge nodes between different network communities by counting the number of shortest paths that pass through a node for specific pairs of non-central nodes (nodes located in the periphery of the network and with less importance). These bridge nodes typically serve as mediators for non-central nodes across different communities. The BCnns is defined as follows.

Definition 7. *For a network $G = (V, E)$ with N nodes, the BCnns of node v in network G is:*

$$BC_{nns}(v) = \sum_{s \in S, t \in T} \frac{\sigma(s, t \mid v)}{\sigma(s, t)}, \tag{33}$$

where $S, T \subset V_{nns}$, V_{nns} is the set of non-central nodes sampled from V, $V_{nns} \subset V$, $S \cap T = \emptyset$. The $\sigma(s, t)$ and $\sigma(s, t \mid v)$ have the same meaning as defined in Definition 2.

By selecting the appropriate pairs of non-central nodes, BCnns can more effectively measure the significance of nodes as bridges between different communities in a network. While these bridge nodes may not have the highest BC value, they are crucial for maintaining overall network connectivity and could potentially have the highest BCnns value.

The Definition of BCnns highlights the importance of selecting suitable nodes for sets S and T. Thus, we proposed an algorithm called SelectionST for node selection. Algorithm 3 describes the procedure of SelectionST. Initially, the nodes are sorted in ascending order based on their DC values, and the first N_1 nodes with lower DC values are selected to create the non-central nodes set V_{nns}. This is because nodes with lower DC values typically have lower centrality and are considered non-central nodes. Next, in order to achieve a more balanced sampling, V_{nns} is divided into two subsets: V_{nns}^{odd} containing nodes at odd indices and V_{nns}^{even} containing nodes at even indices. Lastly, N_2 nodes are randomly sampled from V_{nns}^{odd} to create set S, and N_2 nodes are similarly sampled from V_{nns}^{even} to form set T.

Algorithm 3 Selection$ST(G_c, N_1, N_2)$

```
Input: $G_c = (V_c, E_c)$: the $GCC$ of network $G$,
       $N_1$ and $N_2$: sampling numbers.
Output: $S$ and $T$: the sets of sampling nodes.
 1 if $|V_c| <= N_1$ then
 2     $N_1 \leftarrow \lfloor 0.85 * |V_c| \rfloor$;
 3 sort the nodes of $V_c$ increasing by $DC$ values;
 4 $V_{nns} \leftarrow$ choose first $N_1$ nodes of $V_c$;
 5 $V_{nns}^{odd} \leftarrow$ choose nodes at odd indices of $V_{nns}$;
 6 $V_{nns}^{even} \leftarrow$ choose nodes at even indices of $V_{nns}$;
 7 $N_2 \leftarrow \mathrm{Min}(\left|V_{nns}^{odd}\right|, |V_{nns}^{even}|, N_2)$
 8 $S \leftarrow$ choose $N_2$ nodes of $V_{nns}^{odd}$ randomly;
 9 $T \leftarrow$ choose $N_2$ nodes of $V_{nns}^{even}$ randomly;
10 return $S, T$.
```

The N_1 and N_2 are chosen based on the size of the network and the node degree distribution. Typically, both N_1 and N_2 are much smaller compared to the total number of nodes N. Therefore, BCnns have higher computational efficiency compared to BC, especially for large-scale networks.

Figure 5 demonstrates the differences between BC and BCnns. Specifically, Figure 5a identifies the non-central nodes in red, Figure 5b showcases node sizes based on BC values, and Figure 5c adjusts node sizes based on their BCnns values. Notably, node 14 plays a critical bridging role between two communities, a role that BCnns captures more accurately than BC.

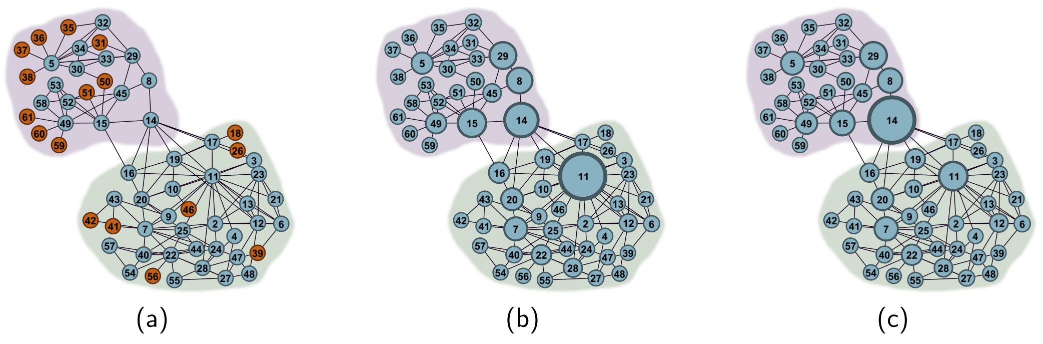


Figure 5. An illustrative example of non-central nodes and comparison of BC and BCnns. In this figure, (**a**) highlights non-central nodes in red, (**b**) showcases node sizes based on BC, and (**c**) showcases node sizes based on BCnns.

3.4.2. GCC-Priority Attack Strategy

As the attack progresses, the network fragments into connected components of varying sizes. The importance of these components varies within the residual network. The GCC refers to the largest connected component containing the most nodes. The destruction of the GCC accelerates the collapse of the network. The GCC-priority attack strategy enhances the attack's effectiveness by targeting nodes within the GCC at each stage of the attack process.

4. Experimental Studies

In this section, we present a series of experiments to verify the effectiveness of our proposed methods. Firstly, we introduce the experimental settings including network datasets and baselines. Next, we compare the proposed PRQMC method with the baselines. Additionally, we demonstrate the effectiveness of the proposed HBnnsAGP attack strategy. Finally, we present further discussions on network robustness when considering the ASR.

4.1. Experimental Settings

4.1.1. Datasets

In our experiments, we selected six real-world classic complex networks of different scales, including Karate [44], Krebs [10], Airport [45], Crime [46], Power [47], Oregon1 [48]. Table 1 provides a detailed summary of these networks, with N and M representing the number of nodes and edges, respectively, and $< k >$ denoting the average degree of the network.

Table 1. Basic Information of 6 Real-World Networks. N and M Represent the Number of Nodes and Edges, Respectively, and $< k >$ Denotes the Average Degree of the Network.

Network	Description	N	M	$< k >$
Karate [44]	Karate club network	34	78	4.59
Krebs [10]	Terrorist network	62	159	5.13
Airport [45]	Aviation network	332	2126	12.81
Crime [46]	Criminal network	829	1473	3.55
Power [47]	Power grid	4941	6594	2.67
Oregon1 [48]	AS peering network	10670	22002	4.12

4.1.2. Comparison Methods

To show the effectiveness of the proposed PRQMC algorithm, we compare it with MC and QMC methods.

- **MC**: It calculates the estimated value of $\widehat{R}$ using original MC integration and generates a set of points from a PRS.
- **QMC**: It calculates the estimated value of $\widehat{R}$ using original QMC integration and generates a set of points from an LDS.

To show the effectiveness of the proposed HBnnsAGP attack strategy, we compare it with three representative baseline attack strategies, including HDA[49], HBA[28], and FINDER[10].

- **High Degree Adaptive (HDA)**: HDA is an adaptive version of the high degree method that ranks nodes based on their DC and sequentially removes the node with the highest DC. HDA recomputes the DC of the remaining nodes after each node removal and is recognized for its superior computational efficiency.
- **High Betweenness Adaptive (HBA)**: HBA is an adaptive version of the high betweenness method. It operates by iteratively removing the node with the highest BC and recomputing BC for the remaining nodes. HBA has long been considered the most effective strategy for

the network dismantling problem in the node-unweighted scenario [50]. However, the high computing cost prohibits its use in medium and large-scale networks.

- **FINDER**: FINDER is notable as an algorithm based on deep reinforcement learning, which achieves superior performances in terms of both effectiveness and efficiency.

We implemented the proposed algorithm and baselines using the Python programming language. All experiments are performed on a server AMD EPYC 7742 64-Core Processor @ 2.25GHz, with memory (RAM) 1024 GB, running Linux ubuntu 11.10 Operating System.

4.2. Comparison of the PRQMC with Baselines

This subsection presents the comparison results to demonstrate the effectiveness of the proposed algorithm, PRQMC, on six real-world complex networks. Specifically, we compare PRQMC with two baselines: MC and QMC. All experiments use the same attack strategy, and the ASR of each node is randomly generated.

We first compare PRQMC with the baselines on two small-scale networks (Karate and Krebs). This is because precise values of RASR can be calculated analytically for small-scale networks. Then, for large-scale networks (Airport, Crime, Power, and Oregon1), we utilize the standard deviation curve as the convergence criterion, as the analytical method is not applicable to large-scale networks. Figure 6 and Figure 7 present the comparison of the convergence and error between PRQMC and baselines. The figure clearly illustrates that PRQMC achieves faster convergence and better accuracy with fewer samples compared to the baselines.

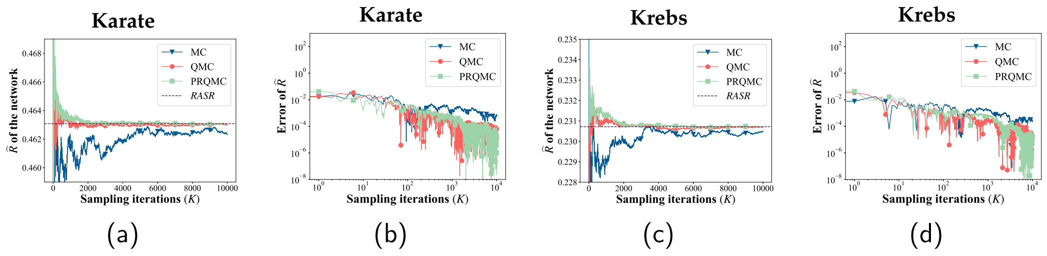


Figure 6. Comparison of the convergence and error of the PRQMC, QMC, and MC methods in assessing robustness for two smaller-scale networks.

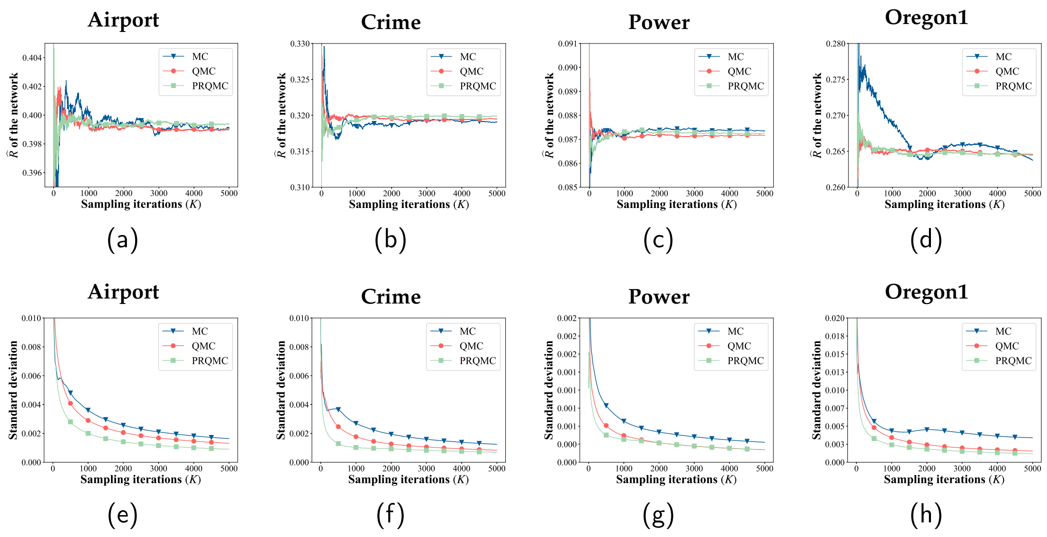


Figure 7. Comparison of the convergence and standard deviation of the PRQMC, QMC, and MC methods in assessing robustness for four larger-scale networks.

Additionally, Table 2 presents a comparison of the computational efficiency of PRQMC and the baselines, each with 5000 sampling iterations. In the PRQMC method, the number of parallel computing processes is set based on the network size, assigning 25 processes to the Karate and Krebs, and 100 processes to the other networks. The results in Table 2 indicate that the PRQMC method outperforms in terms of computational efficiency. Specifically, the PRQMC method operates nearly 50 times faster than the QMC and MC methods on the Oregon1.

Table 2. Computational Time Comparison of PRQMC, QMC, and MC Methods(s)

Network	MC	QMC	PRQMC
Karate	1.8	1.7	**0.4**
Krebs	4.5	4.3	**0.6**
Airport	106.7	104.9	**3.0**
Crime	518.2	520.6	**7.4**
Power	20,525.7	20,529.1	**343.7**
Oregon1	213,748.2	213,758.1	**4,262.4**

4.3. Comparison of the HBnnsAGP with Baselines

In this subsection, we will demonstrate the effectiveness and efficiency of the proposed HBnnsAGP attack strategy. Specifically, we will compare HBnnsAGP with HDA, HBA, and FINDER on six real-world complex networks, while considering different ASR conditions. Initially, we will employ various attack strategies to generate corresponding attack sequences. Subsequently, we will utilize the PRQMC method to calculate the $\widehat{R}$ value under the following ASR distribution scenarios.

1. *ASR = 100%*: The ASR of each node is set to 100%.
2. *ASR = 50%*: The ASR of each node is set to 50%.
3. *ASR = 50% for the first 30% of nodes*: In the attack sequence generated by different attack strategies, the ASR of the first 30% of nodes is set to 50%.
4. *Random ASR*: The ASR of each node is randomly set between 50% and 100%. To obtain more reliable results, the average of 10 experimental outcomes is taken.

The sample numbers (N_1 and N_2) for different networks used in HBnnsAGP are presented in Table 3. Table 4 presents the $\widehat{R}$ values of networks in the four specified scenarios. The data suggests that HBnnsAGP performs better than other attack strategies in terms of destructiveness. The destructiveness of HBnnsAGP, on average, has increased by 6.76%, 4.03%, and 7.26% in comparison to the FINDER, HBA, and HDA strategies, respectively.

Table 5 presents a comparison of computation times for HBnnsAGP and the baselines. As the network size increases, the computation time for the HBA method becomes excessively long. In contrast, the HBnnsAGP method maintains commendable computational efficiency even for larger-scale networks. For the Oregon1 network, HBnnsAGP is approximately 28 times faster than HBA. While the computational efficiency of HBnnsAGP slightly lags behind that of FINDER and HDA for larger-scale networks, it surpasses them in terms of attack destructiveness.

Figure 8 represents the ANCw curves of the networks under various attack strategies when the ASR of each node is set to 100%. In this scenario, the state of the attack sequence is unique. The figure shows that HBnnsAGP excels at identifying critical nodes in the network, leading to the effective disruption of the network structure compared to other methods. Hence the effectiveness of the proposed HBnnsAGP attack strategy is verified.

Table 3. The Sample Numbers (N_1 and N_2) for Different Networks Used in HBnnsAGP

Network	N_1	N_2
Karate	16	8
Krebs	30	16
Airport	100	60
Crime	120	80
Power	1300	80
Oregon1	2300	80

Table 4. The Robustness of Networks Under Different ASR. All $\widehat{R}$ Values Are Multiplied by 100

Network	HBnnsAGP	FINDER	HBA	HDA
1. ASR=100%				
Karate	**12.77**	14.12	15.04	15.04
Krebs	**12.26**	16.26	14.21	17.23
Airport	**7.53**	10.25	7.93	11.10
Crime	**9.90**	11.04	10.14	11.54
Power	**0.91**	5.02	1.01	5.23
Oregon1	**0.68**	1.06	0.73	1.01
Avg score	**7.34**	9.63	8.18	10.19
2. ASR=50%				
Karate	**59.90**	60.55	60.88	61.41
Krebs	**57.61**	58.38	57.74	58.33
Airport	**59.45**	60.32	60.39	60.52
Crime	**57.59**	59.50	59.50	59.08
Power	**16.54**	19.73	17.31	19.93
Oregon1	**49.68**	51.58	51.28	51.33
Avg score	**50.13**	51.69	51.18	51.77
3. ASR = 50% for the first 30% of nodes				
Karate	**48.61**	50.38	49.76	50.57
Krebs	**45.51**	47.16	45.78	47.38
Airport	**48.78**	50.68	51.00	50.47
Crime	**41.84**	48.17	46.90	47.12
Power	**14.87**	17.79	16.32	17.86
Oregon1	**41.26**	42.91	42.83	43.14
Avg score	**40.15**	42.91	42.10	42.76
4. Random ASR				
Karate	**35.12**	36.29	36.50	37.57
Krebs	**30.39**	32.99	31.02	33.36
Airport	**36.95**	38.58	38.67	38.99
Crime	**27.90**	30.96	30.48	30.42
Power	**5.17**	8.46	5.18	8.80
Oregon1	**21.61**	24.23	23.52	23.86
Avg score	**26.19**	28.56	27.55	28.79

Table 5. The Computation Time of Different Attack Strategies(ms)

Network	HBnnsAGP	FINDER	HBA	HDA
Karate	1.6	16.3	1.9	**0.5**
Krebs	3.6	36.6	4.6	**2.3**
Airport	82.3	218.3	211.0	**11.1**
Crime	552.1	369.3	4,434.6	**49.1**
Power	6,760.7	1,397.9	78,119.9	**1,796.8**
Oregon1	15,799.1	8,641.5	477,802.8	**2,065.9**

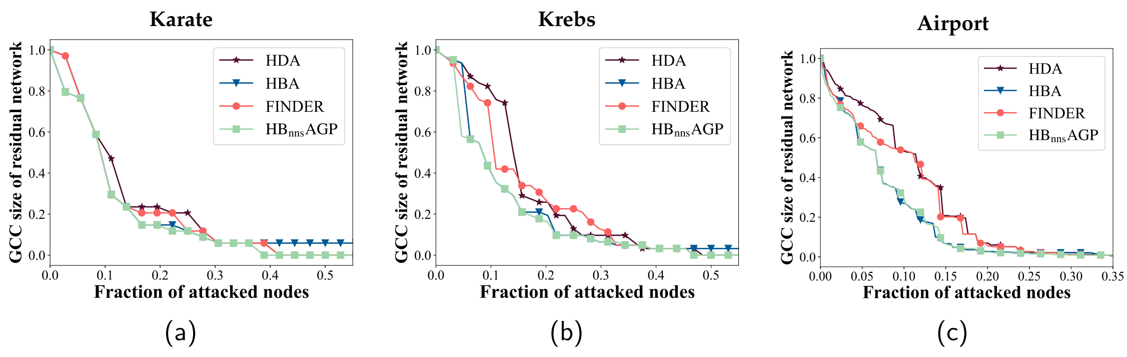


Figure 8. *Cont.*

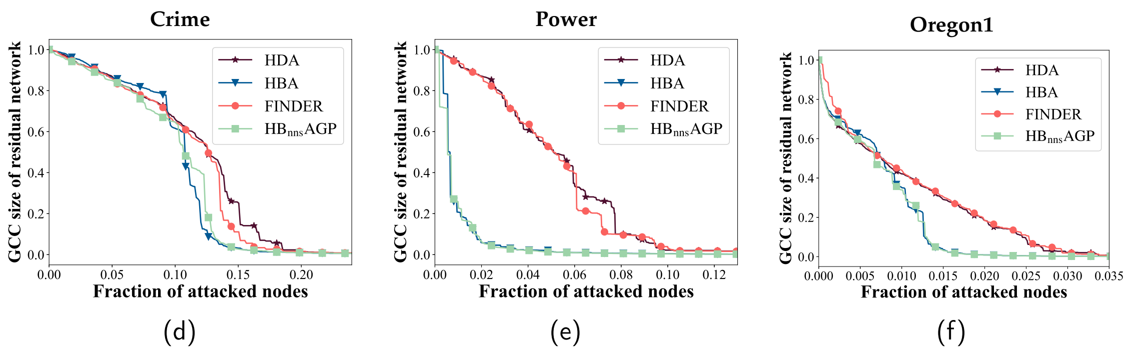


Figure 8. The ANCw curves of networks under different attack strategies.

4.4. Further Discussions About Network Robustness

The data presented in Table 4 indicates that reducing the ASR can significantly enhance network robustness. Generally, this reduction can be achieved through reinforcing node protection. Comparing Scenario 2 and Scenario 3, it is apparent that simply reducing the ASR of the first 30% nodes in the attack sequence (Scenario 3) effectively enhances network robustness. This improvement is approximately 78.25% of that in Scenario 2. Therefore, enhancing the protection of a small subset of crucial nodes in the network can effectively enhance its robustness.

5. Conclusion

In this paper, we conducted a study to analyze the robustness of networks when considering ASR. Firstly, we introduce a novel metric called RASR to assess network robustness in this scenario. Then, we propose the PRQMC algorithm to efficiently calculate the RASR for large-scale networks. PRQMC utilizes RQMC integration to approximate the RASR with a faster convergence rate and employs parallelization to speed up the calculation. Next, we propose a new attack strategy called HBnnsAGP to evaluate the lower bound of network RASR. In HBnnsAGP, we quantify the significance of a node using BCnns and enhance the destructiveness of the attack using the GCC-priority attack strategy. Experimental results on six representative real-world networks demonstrate the effectiveness of the proposed methods. Furthermore, our work demonstrates that reinforcing the protection of a small subset of critical nodes significantly improves network robustness. These findings offer valuable insights for devising more robust networks. The efficiency of the proposed methods can be further enhanced, particularly when analyzing ultra-large-scale networks. In future research, we aim to explore efficient algorithms to enhance the network RASR and devise promising methods for analyzing ultra-large-scale networks.

References

1. Eiselt, H. Destabilization of terrorist networks. *Chaos, Solitons & Fractals* **2018**, *108*, 111–118.
2. Pastor-Satorras, R.; Castellano, C.; Van Mieghem, P.; Vespignani, A. Epidemic processes in complex networks. *Reviews of modern physics* **2015**, *87*, 925.
3. Barabási, A.L. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **2013**, *371*, 20120375.
4. Newman, M. *Networks*; Oxford university press, 2018.
5. Albert, R.; Barabási, A.L. Statistical mechanics of complex networks. *Reviews of modern physics* **2002**, *74*, 47.
6. Berahmand, K.; Haghani, S.; Rostami, M.; Li, Y. A new attributed graph clustering by using label propagation in complex networks. *Journal of King Saud University-Computer and Information Sciences* **2022**, *34*, 1869–1883.
7. Li, Z.; Ma, W.; Ma, N. Partial topology identification of tempered fractional-order complex networks via synchronization method. *Mathematical Methods in the Applied Sciences* **2023**, *46*, 3066–3079.
8. Albert, R.; Jeong, H.; Barabási, A.L. Error and attack tolerance of complex networks. *nature* **2000**, *406*, 378–382.

9. Iyer, S.; Killingback, T.; Sundaram, B.; Wang, Z. Attack robustness and centrality of complex networks. *PloS one* **2013**, *8*, e59613.
10. Fan, C.; Zeng, L.; Sun, Y.; Liu, Y.Y. Finding key players in complex networks through deep reinforcement learning. *Nature machine intelligence* **2020**, *2*, 317–324.
11. Wang, Z.G.; Deng, Y.; Wang, Z.; Wu, J. Disintegrating spatial networks based on region centrality. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **2021**, *31*, 061101.
12. Ma, W.; Fang, J.; Wu, J. Analyzing robustness of complex networks against incomplete information. *IEEE Transactions on Circuits and Systems II: Express Briefs* **2022**, *69*, 2523–2527.
13. Ma, L.; Zhang, X.; Li, J.; Lin, Q.; Gong, M.; Coello, C.A.C.; Nandi, A.K. Enhancing Robustness and Resilience of Multiplex Networks Against Node-Community Cascading Failures. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2022**, *52*, 3808–3821. https://doi.org/10.1109/TSMC.2021.3073212
14. Lou, Y.; Wu, R.; Li, J.; Wang, L.; Li, X.; Chen, G. A Learning Convolutional Neural Network Approach for Network Robustness Prediction. *IEEE Transactions on Cybernetics* **2023**, *53*, 4531–4544. https://doi.org/10.1109/TCYB.2022.3207878
15. Sun, G. Robustness Analysis of an Urban Public Traffic Network Based on a Multi-Subnet Composite Complex Network Model. *Entropy* **2023**, *25*. https://doi.org/10.3390/e25101377
16. Zelenkovski, K.; Sandev, T.; Metzler, R.; Kocarev, L.; Basnarkov, L. Random Walks on Networks with Centrality-Based Stochastic Resetting. *Entropy* **2023**, *25*. https://doi.org/10.3390/e25020293
17. Zhou, M.; Liu, J. A two-phase multiobjective evolutionary algorithm for enhancing the robustness of scale-free networks against multiple malicious attacks. *IEEE transactions on cybernetics* **2016**, *47*, 539–552.
18. Tian, M.; Dong, Z.; Wang, X. Reinforcement learning approach for robustness analysis of complex networks with incomplete information. *Chaos, Solitons & Fractals* **2021**, *144*, 110643.
19. Boccaletti, S.; Latora, V.; Moreno, Y.; Chavez, M.; Hwang, D.U. Complex networks: Structure and dynamics. *Physics reports* **2006**, *424*, 175–308.
20. Barrat, A.; Barthelemy, M.; Vespignani, A. *Dynamical processes on complex networks*; Cambridge university press, 2008.
21. Arulselvan, A.; Commander, C.W.; Elefteriadou, L.; Pardalos, P.M. Detecting critical nodes in sparse graphs. *Computers & Operations Research* **2009**, *36*, 2193–2200.
22. Oka, T.; Wei, W.; Zhu, D. The effect of human mobility restrictions on the COVID-19 transmission network in China. *PloS one* **2021**, *16*, e0254403.
23. Lalou, M.; Tahraoui, M.A.; Kheddouci, H. The Critical Node Detection Problem in networks: A survey. *Computer Science Review* **2018**, *28*, 92–117. https://doi.org/https://doi.org/10.1016/j.cosrev.2018.02.002
24. Bastian, M.; Heymann, S.; Jacomy, M. Gephi: an open source software for exploring and manipulating networks. In Proceedings of the international AAAI conference on web and social media, 2009, Vol. 3, pp. 361–362.
25. Xia, Y.; Fan, J.; Hill, D. Cascading failure in Watts–Strogatz small-world networks. *Physica A: Statistical Mechanics and its Applications* **2010**, *389*, 1281–1285.
26. Zhang, L.; Xia, J.; Cheng, F.; Qiu, J.; Zhang, X. Multi-objective optimization of critical node detection based on cascade model in complex networks. *IEEE Transactions on Network Science and Engineering* **2020**, *7*, 2052–2066.
27. Manoj, B.; Chakraborty, A.; Singh, R. *Complex networks: A networking and signal processing perspective*; Prentice Hall Communications Engineering and Emerging Technologies, Pearson, 2018.
28. Freeman, L.C. A set of measures of centrality based on betweenness. *Sociometry* **1977**, pp. 35–41.
29. Schneider, C.M.; Moreira, A.A.; Andrade Jr, J.S.; Havlin, S.; Herrmann, H.J. Mitigation of malicious attacks on networks. *Proceedings of the National Academy of Sciences* **2011**, *108*, 3838–3841.
30. Press, W.H.; Farrar, G.R. Recursive stratified sampling for multidimensional Monte Carlo integration. *Computers in Physics* **1990**, *4*, 190–195.
31. Caflisch, R.E. Monte carlo and quasi-monte carlo methods. *Acta numerica* **1998**, *7*, 1–49.
32. Feller, W. *An introduction to probability theory and its applications, Volume 2*; Vol. 81, John Wiley & Sons, 1991.
33. Liu, X.; Zheng, S.; Wu, X.; Chen, D.; He, J. Research on a seismic connectivity reliability model of power systems based on the quasi-Monte Carlo method. *Reliability Engineering & System Safety* **2021**, *215*, 107888.
34. Hou, T.; Nuyens, D.; Roels, S.; Janssen, H. Quasi-Monte Carlo based uncertainty analysis: Sampling efficiency and error estimation in engineering applications. *Reliability Engineering & System Safety* **2019**, *191*, 106549.
35. Asmussen, S.; Glynn, P.W. *Stochastic simulation: algorithms and analysis*; Vol. 57, Springer, 2007.

36. Koksma, J. Een algemeene stelling uit de theorie der gelijkmatige verdeeling modulo 1. *Mathematica B (Zutphen)* **1942**, *11*, 43.
37. Hlawka, E. Discrepancy and Riemann integration. *Studies in Pure Mathematics* **1971**, *3*.
38. L'Ecuyer, P. Quasi-Monte Carlo methods with applications in finance. *Finance and Stochastics* **2009**, *13*, 307–349.
39. Bellman, R.E. *Dynamic programming*; Princeton university press, 2010.
40. Kocis, L.; Whiten, W.J. Computational investigations of low-discrepancy sequences. *ACM Transactions on Mathematical Software (TOMS)* **1997**, *23*, 266–294.
41. Morokoff, W.J.; Caflisch, R.E. Quasi-monte carlo integration. *Journal of computational physics* **1995**, *122*, 218–230.
42. Lou, Y.; Wu, R.; Li, J.; Wang, L.; Chen, G. A convolutional neural network approach to predicting network connectedness robustness. *IEEE Transactions on Network Science and Engineering* **2021**, *8*, 3209–3219.
43. L'Ecuyer, P. Random number generation and quasi-Monte Carlo. *Wiley StatsRef: Statistics Reference Online* **2014**, pp. 1–12.
44. Zachary, W.W. An information flow model for conflict and fission in small groups. *Journal of anthropological research* **1977**, *33*, 452–473.
45. Colizza, V.; Pastor-Satorras, R.; Vespignani, A. Reaction–diffusion processes and metapopulation models in heterogeneous networks. *Nature Physics* **2007**, *3*, 276–282.
46. Rossi, R.; Ahmed, N. The network data repository with interactive graph analytics and visualization. In Proceedings of the AAAI conference on artificial intelligence, 2015, Vol. 29.
47. Watts, D.J.; Strogatz, S.H. Collective dynamics of 'small-world'networks. *nature* **1998**, *393*, 440–442.
48. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* **2007**, *1*, 2–es.
49. Pastor-Satorras, R.; Vespignani, A. Epidemic spreading in scale-free networks. *Physical review letters* **2001**, *86*, 3200.
50. Holme, P.; Kim, B.J.; Yoon, C.N.; Han, S.K. Attack vulnerability of complex networks. *Physical review E* **2002**, *65*, 056109.