GANI: Global Attacks on Graph Neural Networks via Imperceptible Node Injections

Junyuan Fang ©, Student Member, IEEE, Haixian Wen ©, Jiajing Wu ©, Senior Member, IEEE, Qi Xuan ©, Senior Member, IEEE, Zibin Zheng ©, Fellow, IEEE, and Chi K. Tse ©, Fellow, IEEE

Abstract—Graph neural networks (GNNs) have found successful applications in various graph-related tasks. However, recent studies have shown that many GNNs are vulnerable to adversarial attacks. In a vast majority of existing studies, adversarial attacks on GNNs are launched via direct modification of the original graph such as adding/removing links, which may not be applicable in practice. In this article, we focus on a realistic attack operation via injecting fake nodes. The proposed global attack strategy via node injection (GANI) is designed under the comprehensive consideration of an imperceptible perturbation setting from both structure and feature domains. Specifically, to make the node injections as imperceptible and effective as possible, we propose a sampling operation to determine the degree of the newly injected nodes, and then generate features and select neighbors for these injected nodes based on the statistical information of features and evolutionary perturbations obtained from a genetic algorithm, respectively. In particular, the proposed feature generation mechanism is suitable for both binary and continuous node features. Extensive experimental results on benchmark datasets against both general and defended GNNs show strong attack performance of GANI. Moreover, the imperceptibility analyses also demonstrate that GANI achieves a relatively unnoticeable injection on benchmark datasets.

Index Terms—Graph adversarial attacks, graph neural networks (GNNs), node injections, robustness, unnoticeable perturbations.

I. INTRODUCTION

Graphs, where nodes represent individuals and links represent relationships between individuals, are ubiquitous in real-world systems. With the remarkable success of deep graph learning, various powerful graph neural network (GNN) models have been proposed to deal with graph data-based tasks, such as node/graph classification, community detection, and link prediction [1], [2], [3], [4], [5], [6], [7].

Recently, Goodfellow et al. [8] found that current neural networks are vulnerable to the unnoticeable adversarial samples generated by the attackers. Similarly, though GNNs have shown great potential on graph learning, researchers are also curious about the robustness of current GNNs when facing malicious attacks, namely graph adversarial attacks [9], [10], [11], [12]. Prior work has proposed to evaluate the robustness of current GNNs against perturbations based on the gradient of classifiers [13], metalearning [14], reinforcement learning [15], etc. Specifically, corresponding experiments have demonstrated the vulnerability of most current GNNs, even under small perturbations.

For the vast majority of existing studies on graph adversarial attacks [13], [14], [15], [16], [17], a common assumption is that the attacker has the privilege to modify the original data, such as adding/removing links or modifying features of the original graph data. However, this assumption may not be valid in many realistic situations. Taking social networks as an example, it is rather difficult for an attacker to tamper with the relationship or the personal information of other users. Instead, an alternative strategy is to make an adversarial attacker generate some vicious nodes (i.e., adversarial nodes) and associate them with the original nodes to achieve the attacks. These kinds of adversarial attacks can be referred to as node injection attacks (NIAs).

In the NIA scenario, the attacker needs to solve two major problems. The first problem is to generate the feature of the adversarial nodes, while the other is to decide the neighbors of these newly injected nodes. Although a few NIA methods have been proposed, there still exist two main limitations.

1) Feature inconsistency: Ignoring the diversity of feature spaces in different realistic networks, current work usually assumes the attacker generates binary features (0/1) to the new nodes. Other feature generation mechanisms like averaging features or combining with Gaussian distribution will also lead to inconsistent problems on ranges or types.

2) Budget constraint: Current NIA methods usually define the degree of each new node as a constant value such as the average degree of original nodes. Therefore,
considering the above limitations, the NIAS in most existing work may cause obvious disturbance to the feature and topology domains of the original network, making the attacks easy to be detected.

To address the above concerns and conduct effective and unnoticeable NIA, we propose a new NIA strategy GANI to achieve global attacks via node injections. We focus on global attacks (i.e., decreasing the global performance) of the node classification task via injecting some fake nodes into the original graph. Specifically, the proposed GANI can be divided into the following two parts. In the feature generation procedure, the statistical feature information of the original nodes is utilized to make our new nodes similar to the normal nodes. In the link generation procedure, we employ evolutionary algorithms to help find the optimal links which can obtain the best attack performance. Moreover, we utilize the decrease of node homophily (NH) [18], [19] to reduce the randomness of the optimal neighbor selections. Finally, we verify the superiority on both attack performance and imperceptibility of the proposed attack strategy in datasets with different feature types through extensive experiments. Our major contributions are as follows.

1) We present new insights on promoting the imperceptibility of NIA from both topology and feature domains which were seldom discussed in previous studies.
2) We propose a powerful NIA method by combining the statistical information-based feature generation step with evolutionary algorithm-based neighbor selection step on the global poisoning attack scenario.
3) We conduct extensive experiments on several datasets with different types of features to verify the effectiveness and imperceptibility of the proposed method.

The remainder of this work is organized as follows. Section II discusses the related work of current attack and defense on GNNs. Some preliminaries about GNNs, NIAS, and the definition of homophily are given in Section III. Then, we introduce the setting of attack budgets, methods for feature generation, and neighbor selection of injected nodes in Section IV. In Section V, we present the corresponding experiments and discussions on both attack performance and the imperceptible effect of perturbations in detail. Finally, we conclude our work in Section VI.

II. RELATED WORK

In this work, we aim at achieving global attacks, which is to reduce the overall effectiveness of the models, rather than focusing on influencing some target nodes. Moreover, we concentrate on poisoning attacks on graphs, for which we will apply our attack strategy on the retraining models, rather than on the fixed models with the same parameters before and after the attack (i.e., evasion attacks). Compared with evasion attacks, poisoning attacks will be more realistic but difficult due to the model retraining. In the following, we will briefly introduce current efforts on typical adversarial attacks and NIAS, respectively. Then, we will briefly introduce current (defended) methods on graphs.

A. Typical Adversarial Attacks on GNNs

In most existing studies, attacks are launched by modifying the links or features of the original nodes. Zügner et al. [13] proposed the first graph adversarial attack method, Nettack, to point out the security concern in current GNNs by utilizing the classification margin of the nodes predicted in different classes. Dai et al. [15] put forward a reinforcement learning-based strategy, RL-S2V, to achieve the targeted evasion attacks. In order to achieve global attacks, Zügner and Günnemann [14] further proposed metattack (META for short in this work) by utilizing the meta gradient of loss functions. However, all of the above strategies are designed based on the assumption that the attackers can easily modify the original graphs, which is not realistic in the real life. Moreover, we also find that directly extending current methods to the node injection scenario cannot achieve the same performance, which will be shown in our experiments later.

B. NIAS on GNNs

To address the above concerns, some NIA methods have been proposed recently. Sun et al. [20] introduced a strong NIA strategy named node injection poisoning attack (NIPA) which utilizes the deep Q-learning method to achieve the global poisoning attacks. Aiming at targeted poisoning attacks, Wang et al. [21] proposed approximate fast gradient sign method (AFGSM) by considering the gradient of loss function of targeted nodes. Zou et al. [22] introduced TGDIA, a global evasion injection method combining topological defective edge selection strategy and smooth feature generations. Moreover, Tao et al. [23] proposed generalizable node injection attack (GNIA) by modeling the attack process via a parametric model to preserve the learned patterns and further be utilized in the inferring phase. Recently, Chen et al. [24] proposed harmonious adversarial objective (HAO) by utilizing the homophily ratio of nodes as the unnoticeable constraint to further improve the unnoticeability of adversarial attacks. Ju et al. [25] modeled the NIA process as a Markov decision process and developed a reinforcement learning-based attack method, gradient-free graph advantage actor critic (G2A2C), in the black-box targeted evasion scenarios.

However, the two concerns (i.e., feature inconsistency and budget constraint) still have not been well addressed in prior work. Specifically, NIPA generates the adversarial features through the average features of all nodes plus Gaussian distribution. Since different datasets may have different types of features (i.e., binary and continuous features), and the features of each node usually will be an extremely sparse vector, the Gaussian noise may dominate the feature of new nodes, causing...
the situation that the features of newly injected nodes will be very different from those original nodes. Moreover, AFGSM simply generates binary features for the new nodes, while TDGIA only generates continuous features and fails to limit the feature-attack budget. Among them, only G-NIA and G²A2C solve the feature consistent problem. At the same time, none of the above work tackles the degree problems of the new nodes. Though HAO retains the homophily ratio of corresponding nodes during injections in a local perspective, it may break other global network properties or constraints such as degree distributions, the range/type of node features which can be easily detected.

Therefore, current NIA methods may break the unnoticeable principle as the new nodes have a totally different range distribution or belong to different types (i.e., binary or continuous format) of features. They may also make the injection more obvious by assuming that all new nodes have the same degree. In order to make the NIA more realistic, we aim to address the above issues that exist in current strategies. Particularly, to make the adversarial attacks more general, we focus on a more practical and difficult scenario, namely global poisoning attacks.

C. Defense on GNNs

Standard GNNs usually utilize the neighborhood aggregation mechanism to help obtain a better representation of the central node from its neighbors, such as classic graph convolution network (GCN) [26], graph attention network (GAT) [27] by giving different aggregation weights to different neighbors via a masked self-attention mechanism, and simplifying graph convolution network (SGC) [28] by removing the activation functions in the middle layers.

Since current GNNs are vulnerable to adversarial attacks, some other models are proposed to improve the robustness of GNNs through the preprocessing of data, redesigning the aggregation functions, etc. For example, Wu et al. [29] came up with Jaccard by removing the links with relatively lower feature similarity before employing the GNNs. Zhu et al. [30] designed robust graph convolutional networks (RGCN) via modeling the low-dimensional representation of nodes as Gaussian distribution and further employing a variance-based attention mechanism, and simplifying graph convolution network (SimPGCN) [28] by removing the activation functions in the middle layers.

In this section, we will give some preliminaries on GNNs and NIAAs, together with the definition of NH.

III. PRELIMINARIES

A. GNNs

Generally, we use \( G = (A, X) \) to denote the graph, where \( A \) and \( X \) are the adjacency matrix and feature matrix, respectively. Assuming \( G \) has \( n \) nodes and each of them has \( d \) features, the adjacency matrix \( \tilde{A} \in \{0,1\}^{n \times n} \) represents the connection relationship between each node-pair, where \( \tilde{A}_{ij} = 1 \) means there is a link between nodes \( i \) and \( j \), and 0 otherwise. The feature matrix \( \{X_v\}^{n \times d} \) denotes the features of each node, and \( X_i \) represents the feature vector of node \( i \). As mentioned before, the feature vector could be either binary or continuous type. Following the classic work on node classification [26], a general two-layer GCN can be represented by

\[
Z = f_\theta(A, X) = \text{softmax}(\tilde{A}\sigma(\tilde{A}XW^{(1)}))W^{(2)}
\]

where \( \tilde{A} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}} \), \( \tilde{D} \) and \( I \) represent the diagonal degree matrix and identity matrix, respectively, \( \sigma(\cdot) \) is the activation function like rectified linear unit (ReLU), and \( W \) is the weight matrix to be optimized. Specifically, \( Z_{v,c} \) denotes the probability that node \( v \) belongs to class \( c \). The optimization goal of node classification is to assign the node a higher probability to the true class, i.e.,

\[
\min_\theta L = -\sum_{v \in V_{\text{train}}} \ln Z_{v,c_v}
\]

where \( c_v \) indicates the ground truth label of node \( v \), \( V_{\text{train}} \) represents the nodes of training set, and \( \theta = \{W^{(1)}, W^{(2)}\} \) is the learning parameter to be optimized.

B. NIAAs

Instead of directly modifying the original adjacency matrix \( A \) and feature matrix \( X \), NIA will inject some malicious nodes into the original graph and generate the features of these injected nodes and then connect them to the original graph to conduct the attack. The adversarial graph after being injected \( n_{\text{in}} \) fake nodes is given as

\[
\begin{align*}
A' &= \begin{bmatrix} A & A_p^T \\ A_p & O \end{bmatrix} \\
X' &= \begin{bmatrix} X \\ X_p \end{bmatrix}
\end{align*}
\]

where matrices \( \{A_p\}^{n_{\text{in}} \times n} \) and \( \{A_p^T\}^{n_{\text{in}} \times n} \) represent the adversarial matrices between the fake nodes and normal nodes and corresponding transpose matrix, respectively. \( \{X_p\}^{n_{\text{in}} \times d} \) is the new generated features of the new nodes.

Specifically, we focus on global poisoning attacks on node classification, and the final goal of the attacker is to decrease the overall classification accuracy of the targeted GNNs, which can be represented as

\[
\max_\theta \sum_{v \in V_{\text{test}}} \mathbb{1}(\arg\max_{\theta} f_\theta(A', X')_v \neq c_v)
\]

s.t. \( \theta^* = \arg\max_\theta \sum_{v \in V_{\text{train}}} \ln f_\theta(A', X')_v, c_v \)
where $\theta^*$ indicates the optimal training parameters trained on the perturbed graph, $\arg\max_{\theta^*} \ln f_{\theta^*}(A, X)_v$ is the predicted label of node $v$ based on $\theta^*$, and $I(\cdot)$ is an indicator function with output 1 when the argument is true, and 0 otherwise.

C. NH

Similar to some previous studies [18], [19], we define NH of a specific node based on the ratio of neighbors whose label is the same as the central nodes. The detailed calculation of NH and the average NH of the whole graph is given by

$$NH = \frac{\#\ \text{neighbors with same label}}{\#\ \text{all neighbors}}$$  \hspace{1cm} (6)

$$\text{Average NH} = \frac{1}{n} \sum_{i=1}^{n} NH_i$$  \hspace{1cm} (7)

where $n$ is the total number of nodes in the graph.

To get a better understanding of NH, we analyze average NH of the largest connected component of four benchmark datasets, whose statistical information is given in Table II in Section V. As shown in Table I, we can observe that all these datasets have an extremely high average NH closing to 0.9, meaning that most neighbors will have the same label as the central nodes. The above finding is also consistent with the design of the neighborhood aggregation mechanism on GNNs which utilizes the information of neighbors to help the central nodes obtain a better representation. Therefore, we try to utilize NH to design our attack strategy in the next section.

IV. PROPOSED METHOD

In this section, we describe the proposed method for solving the following issues. First, to keep our attacks from being detected, we need to limit our adversarial perturbations on both structure and feature domains to reasonable budgets. Then, we need to solve the two major challenges in NIA, including feature generation and neighbor selection.

A. Attack Budgets

In order to make the attacks as unnoticeable as possible, we need to define the attack limitation appropriately. Specifically, in the NIA scenario, we decide both the feature-attack budget $\Delta F$ and link-attack budget $\Delta L = \{\Delta L_0, \Delta L_1, \cdots, \Delta L_{n_{in} - 1}\}$ as we will inject some fake nodes into the original graph. For the feature-attack budgets, since the features of nodes are usually extremely sparse vectors, we follow the setting of the previous work [21] by assigning the average number of nonzero features of normal nodes to the newly injected nodes. For the link-attack budget, previous work usually assigns the average degree of original nodes to all new nodes. Yet we believe such settings will cause an obvious change to the graph’s structural properties as many new nodes having the same degree will be injected at the same time, which can be easily detected. Therefore, to keep a higher similarity in the structural properties between the graph before and after the attacks, we propose a degree sampling operation to decide the link-attack budget of each newly injected node. Specifically, we randomly select an original node first, then assign the degree of this node to the current new node, and repeat this procedure until all $n_{in}$ nodes have been assigned. Particularly, we set the maximal link-attack budget of each injected node to $2 \times$ average degree, aiming to avoid the extremely large perturbations on a single node. Since the degree with a higher frequency in the original graph will have a higher probability to be sampled in the above procedure, the similarity of graphs before and after the attacks can be retained to a great extent.

B. Feature Generation of Injected Nodes

To avoid the attacks being detected, we prefer the newly injected nodes to be similar to the original nodes. Therefore, from the perspective of features, we employ a statistical operation to generate the adversarial features. During the feature generation of each new node, we first assign a targeted label $L'$ randomly, and then calculate the total nonzero appearance of features in each index among $d$ dimensions of all original nodes whose label belongs to class $L'$. Further, we choose the top $\Delta F_i$ feature indices with the highest appearance as the nonzero indices of the current injected node. After that, for a specific feature index $\Delta F_i$ of new node $k$, the specific feature value will be calculated as

$$X'[k, \Delta F_i] = \frac{\sum_{Y[v] = L'} X[v, \Delta F_i]}{\sum_{Y[v] = L'} I(X[v, \Delta F_i] \neq 0)}$$  \hspace{1cm} (8)

where $Y$ is the label matrix and $X$ is the original feature matrix. It is worth noting that the above feature generation mechanism is suitable for both binary and continuous features. For binary features, the calculated feature values on the corresponding nonzero indices $X'[k, \Delta F_i]$ will also be 1. Since $X$ is usually a sparse matrix, (8) can promise the newly injected features having the same range of value as normal nodes because averaging is only done by the nonzero features in this index, rather than all nodes. After the above feature generation operation, the newly generated node will have a relatively higher similarity with the normal nodes with the same targeted label in the feature domain. In addition, we also can apply some randomness to this feature generation procedure to make it more unnoticeable.

C. Neighbor Selection of Injected Nodes

After generating the adversarial features to the newly injected nodes, our next challenge is to assign appropriate neighbors to achieve the best attack performance. According to (5), the principal goal of the proposed model is to decrease the overall prediction accuracy of testing nodes. In other words, we want to figure out what kind of changes in the adversarial
adjacency matrix $A'$ and feature matrix $X'$ can achieve the best attack performance. Based on the analysis of Table I, we know that higher NH usually helps the GNNs to obtain a better performance in the downstream task as the neighbors have a positive effect on obtaining the representation of central nodes during aggregations. Therefore, an intuitive idea is to decrease the homophily of nodes, thereby influencing the neighborhood aggregation procedure of GNNs.

For a specific injected node, we need to select its neighbors that can improve the optimization of (5). Interestingly, the neighborhood selection can be modeled as a combinatorial optimization problem. Evolutionary algorithms designed by imitating the biological evolution processes have proved to be an effective tool to solve complicated combinatorial optimization problems [34], [35], [36], [37]. In the graph adversarial learning domain, previous work also utilizes the evolutionary algorithms to achieve optimal perturbations on the graph modification problems [34], [35], [36], [37]. In the graph adversarial learning domain, previous work also utilizes the evolutionary algorithms to achieve optimal perturbations on the graph modification problem. Evolutionary algorithms designed by imitating the biological evolution processes have proved to be an effective tool to solve complicated combinatorial optimization problems [34], [35], [36], [37].

In the experiments, we find that the outputs of different $A$'s that have the same value of $f$ can be the ground truth or the predicted label from $f$. Also, $c_v$ could be the ground truth or the predicted label from surrogate models under different settings.

In the experiments, we find that the outputs of different $A'$ in (10) can only differ slightly. In other words, there may be numerous different $A'$s that have the same value of $f$. If we randomly select one of them as the optimal adversarial adjacency, our method may be unstable due to the randomness on neighbor selection. Therefore, we introduce another sorting index, the decrease of NH (DNH), to reduce the randomness of the above selection. The definition of DNH of a specific node $i$ is as

$$\text{DNH}_i = \text{NH}_i^\text{before} - \text{NH}_i^\text{after}$$

where $\text{NH}_i^\text{before}$ and $\text{NH}_i^\text{after}$ refer to the NH before and after node $i$ being attacked, respectively. Moreover, for a specific individual $I = \{n_{v_1}, n_{v_2}, \ldots, n_{v_k}\}$ that consist of $k$ injected links in GA, the total DNH (labeled as TDNH) of individual $I$ is given as

$$\text{TDNH}_I = \text{DNH}_{n_{v_1}} + \text{DNH}_{n_{v_2}} + \cdots + \text{DNH}_{n_{v_k}}$$

where DNH$_{n_i}$ is the specific decrease of NH of node $n_i$.

Following the definition of DNH, we further define a two-level sorting mechanism by combining the original fitness function $f_{in}$ and the total decrease of NH TDNH in each individual. For instance, for two different $A'$ with the specific value $(f_{in1}, \text{TDNH}_1)$ and $(f_{in2}, \text{TDNH}_2)$, the proposed sorting mechanism will first order them based on the priority of original fitness value if $f_{in1} \neq f_{in2}$. However, if $f_{in1} = f_{in2}$, we will retain the individual with a higher TDNH. Based on this principle, the major steps of GA for a specific node $n_v$ are as follows.

1) **Parameter definition:** We define a specific individual as $I = \{n_{v_1}, n_{v_2}, \ldots, n_{v_k}\}$ where $(n_{v_i}, n_{v_j})$ is a link connecting the adversarial node $n_{v_i}$ with a normal node $n_{v_j}$. The population is $P = \{I_0, I_1, I_2, \ldots\}$, candidate rate is $\alpha$, crossover rate, and mutation rate are $p_c$ and $p_m$, respectively.

2) **Candidates selection:** For current node $v_i$ with the targeted label $L'$, as the searching space is extremely large due to the inclusion of all possible nodes as $n_i$, we first remove the nodes having label $L'$ as we want to decrease the NH. Then, we employ single-link attacks under the guide of $f_{in}$ and TDNH. We will only retain the nodes in the top $\alpha$ as the final candidates which may be selected later.

3) **Population initialization:** From the obtained candidates, we generate an initial population by randomly selecting the node from candidates as node $n_{i(j)}$ in each individual.
4) **Crossover and mutation:** Under the crossover rate $p_c$ and mutation rate $p_m$, we will exchange a part of adversarial links between every two individuals in the crossover operation and replace an adversarial link with a new randomly generated link in each individual in the mutation operation.

5) **Fitness calculation:** We inject each individual to the original graph by constructing current adversarial adjacency matrix $A$. Together with the features $X$ generated from feature generation step, we can obtain the fitness list $F_{\text{fit}}$ by recording each $f_{\text{fit}}$ and TDNH calculated through (10) and (12).

6) **Elite selection:** Finally, to maintain the scale of $P$, we utilize the tournament selection mechanism to retain the better individuals. The individuals with higher $f_{\text{fit}}$ and TDNH will be more likely to be retained.

Specifically, steps 4–6 will repeat until the predefined maximum iteration is exceeded.

**D. Complexity Analysis**

In this section, we provide the complexity analysis of GANI to demonstrate its efficiency. Specifically, based on the discussion of Section IV, the time complexity of GANI is dominated by the calculation of the attack effect of a specific individual (i.e., adversarial link sequence) using (9)–(12), especially for the matrix multiplication in (9). Hence, we employ the frequency of invoking (9) to analyze the complexity of GANI in the following.

We first briefly give some parameters related to the frequency of invoking (9), i.e., the number of times (9) is invoked. Suppose we have an original graph $G$ with $n$ nodes where these nodes are uniformly distributed to $C$ classes. Moreover, we have a total of $n_{\text{in}}$ injected nodes. For the parameters of neighborhood selection phase, we denote the maximum number of iterations by $T$ and the size of population by $|P|$. Therefore, for the injection of each node, the number of times (9) is invoked primarily arises from the candidates selection step and fitness calculation step. Specifically, for the candidates selection step, we filter out unnecessary candidates that share the same label as the currently injected node. Thus, (9) is invoked $n(C - 1)/C$ times, as the nodes are uniformly distributed to each class. For the fitness calculation step, we need to obtain the attack effect of each individual. Since the size of population will be doubled after the crossover operation and the maximum number of iterations is $T$, the number of times (9) is invoked during this process is thus equal to $2|P|T$. Finally, as we will inject $n_{\text{in}}$ nodes, the total number of times (9) is invoked becomes $n_{\text{in}}((n(C - 1)/C) + 2|P|T)$.

From the above analysis, we can observe that, as the number of injected nodes $n_{\text{in}}$, the size of population $|P|$, and the maximum number of iterations $T$ are typically much smaller than the number of nodes $n$ in the original graph, the computational cost of GANI is determined the network scale. It is worth noting that, under a fixed value of $T$, the proposed candidates selection step narrows the focus of GA to explore high-quality candidates rather than tackling the overwhelmingly large search space consisting of all nodes. Compared to the latter, our GANI demonstrates the potential to converge to better solutions within the same maximum number of iterations $T$.

**E. Overall Framework**

To sum up, the overall framework of the proposed GANI on global poisoning attacks on GNNs consists of the GANI attack phase and evaluation phase, as illustrated in Fig. 2. During the attack phase, we first assign the link-attack budgets to each newly injected node through degree sampling operation, while the feature-attack budget is restricted by the average nonzero feature indices of original nodes. Then, we generate the specific features for the current adversarial node based on the proposed statistical information of features in the feature generation step. Next, we will utilize GA to find the near-optimal neighbors on neighbor selection step. This procedure will repeat until all $n_{\text{in}}$ adversarial nodes have successfully been injected, so the proposed GANI is a sequential generation method rather than a one-time injection, which will be more flexible if we want to increase the number of injection nodes. Finally, in the evaluation phase, we input the constructed adversarial graph to the GNNs for training as we focus on poisoning attacks and further verify the attack performance of GANI.

**V. EXPERIMENTS**

In this section, we verify the effectiveness of GANI via comprehensive experiments. We will first introduce the corresponding setups including the statistics of datasets, detailed baseline methods, the GNNs that will be attacked, and parameter settings. Then, we present corresponding experimental results and analysis to verify the performance of the proposed method.

**A. Experimental Setup**

1) **Datasets:** We choose four representative benchmark datasets including Cora, Citeseer, PubMed [43], and Cora-ML [44] to conduct the experiments. Among them, Cora and Citeseer have binary features, while the other two have continuous features. Adopting the setting of previous work [13], we use the largest connected component of these datasets in our experiments. The statistics are shown in Table II.

2) **Baselines:** For the baselines, we mainly compare our GANI with the extension of general targeted and global attack methods on NIA scenarios, and recent targeted and global NIA methods. Specifically, the compared methods can be divided into the following three groups: the first group is the extension of two strong general attack methods, including Nettack [13] and META [14]; the second group is the targeted NIA methods including AFGSM [21], G-NIA [23], and $G^2$A2C [25]; and the last group is the global NIA method, TDGIA [22]. We do not compare our method with NIPA [20] as no official codes are available.

Since current attack methods are not directly designed to the global poisoning attacks under NIA scenarios, we need to make some necessary modifications to extend their default designs, for example, from general attacks to NIAS and from the targeted...
attacks to global attacks. To ensure fairness, for the feature generation of newly injected nodes, we will randomly copy the features of the original nodes to the newly generated nodes if the corresponding baselines do not solve the feature consistent problem under a given feature-attack budget.

The detailed description of baselines and corresponding extended designs of them are as follows.

a) Nettack: It [13] is a powerful targeted attack method based on modifying the original graph and the feedback of classification margins. To extend the original Nettack to the NIA scenario, we first randomly sample $n_{\text{in}}$ feature vectors and corresponding labels of original nodes as the newly injected nodes, then initialize the original neighbors following the generation of the BA scale-free model [45] by giving a higher connected probability to the original nodes with a higher degree. Moreover, as Nettack is a targeted attack method while we focus on global attacks, we borrow the idea from the comparison between Nettack and the global attack method META (which will be introduced later). Specifically, during the injection of each node, we randomly choose a node in the test set as the targeted node, then employ the original Nettack but limit the adversarial link such that it only occurs between the targeted node and the new $n_{\text{in}}$ adversarial nodes.

b) META: It [14] is a global attack strategy which modifies the original graph based on meta gradients. To extend the original META to the NIA scenario, we also randomly sample $n_{\text{in}}$ in feature vectors and corresponding labels of original nodes as the newly injected nodes, then initialize the original neighbors following the generation of the BA scale-free model, which is the same as the extension of Nettack. After that, we employ the original META but only allow the attack operations to happen between the newly injected $n_{\text{in}}$ nodes and normal nodes.

c) AFGSM: It [21] is a NIA method utilizing the gradients of loss functions of targeted nodes. Since AFGSM focuses on a single node attack once a time, we also extend the original AFGSM to be a global attack method like Nettack does. Specifically, for each new adversarial node, we randomly choose a node in the test set as the targeted node and then employ the original AFGSM to achieve targeted attacks. Moreover, as AFGSM can only generate binary features, we will assign the randomly selected feature vectors of original nodes to the adversarial nodes on two datasets (i.e., Cora-ML and PubMed) with continuous features.

d) G-NIA: It [23] is a targeted evasion NIA method by utilizing a parametric model to learn the attack patterns. Since the designs of G-NIA cannot only retain the feature consistency but also promise a given unnoticeable feature-attack budget, we only need to extend it to global attacks by randomly selecting a node in the test set for each injected budget, which is the same as the change on Nettack and AFGSM.
e) \( G^2 \text{A2C} \): It [25] is a strong reinforcement learning-based targeted evasion NIA method that models the NIA as a Markov decision process and achieves remarkable attack performance in a black-box scenario. Similar to G-NIA, \( G^2 \text{A2C} \) is also adaptive to the diverse feature consistency requirements. However, \( G^2 \text{A2C} \) ignores the sparsity property on datasets with continuous features. Therefore, besides extending it to the global attack scenario, we will assign the randomly selected feature vectors of original nodes to adversarial nodes during the attack on Cora-ML and PubMed.

f) TDGIA: It [22] is a well-performed global NIA method combining both the topological defective edge selections and smooth feature generations. However, the original TDGIA fails to promise a given feature-attack budget, which ignores the situation that the features of nodes may be extremely sparse vectors. Moreover, TDGIA is only suitable for generating continuous features, causing feature inconsistent issues once the target graph has binary features. Therefore, we will first assign the randomly selected feature vectors of original nodes to the adversarial nodes to ensure the imperceptible feature generations, then employ the original neighbor selections of TDGIA.

3) Target Models: For the target GNNs, we employ GANI and other baselines to attack two general GNNs including GCN [26] and SGC [28], and two defended or robust GNNs including Jaccard [29] and SimPGCN [31].

a) GCN: It [26] is one of the most representative GNNs. GCN learns the high-level representation of central nodes by aggregating the features of local neighbors layer by layer.

b) SGC: It [28] is the simplified version of GCN via omitting the activation functions such as ReLU in the middle layers. SGC achieves a competitive performance as original GCN while having a much smaller parameter complexity.

c) Jaccard: It [29] is a inspectors GNN by employing a preprocess operation on the basic GCN. Specifically, Jaccard first calculates the Jaccard similarity of connected node pairs and only preserves those links with higher similarity, based on the assumption that connected node pairs with low similarity usually will be noise. Particularly, as the features with continuous type cannot directly utilize the original Jaccard similarity, we use the cosine similarity as a replacement on Cora-ML and PubMed datasets.

d) SimPGCN: It [31] is a recently proposed defended method for improving the aggregator designs of GNN. Specifically, SimPGCN will adaptively preserve both the feature similarity and structural similarity during the aggregation process to resist structural attacks.

Since GANI is optimized through the simple SGC model, the performance of attacking the other three GNN models indicates the generality of our method.

4) Parameter Settings: Following the previous work [13], the training, validation, and test ratio of all datasets are 10%, 10%, and 80%, respectively. All experiments take the average performance together with standard deviation under five different splits. All attack methods and attacked models are adopted from the default parameter settings. Specifically, for the proposed method, the candidate ratio \( \alpha \) is set to 50%. The population size and maximal iteration are set to 40 and 100, respectively. Moreover, the crossover rate \( p_c \) and mutation rate \( p_m \) are set as 0.5 and 0.3 based on the grid search process, respectively. In addition, to have a fair comparison, we employ the same link-attack budget \( \Delta L \) to all baselines. It is worth noting that Nettack and META will actually have \( 2 \ast \Delta L \) link-attack budget because of the initialization of the BA model. Particularly, to speed up the calculation of GA in the large-scale searching space, GANI is implemented via parallel computing by employing the message passing interface [46]. The codes are publicly available at https://github.com/alexfanjn/GANI.

B. Experimental Analysis

1) Overall Attack Performance on GNNs: The attack results on four GNNs of different baselines by injecting 5% nodes are shown in Table III. Among them, GANI achieves the best attack performance almost in all cases (with the overall best average ranking among all rows). We interestingly find that the extended methods also can obtain a remarkable attack performance on some cases, especially for Nettack. However, compared with the significant attack performance on Cora and Citeseer, Netack performs worse on Cora-ML and PubMed datasets, indicating that it may not be suitable for the datasets with continuous features. Methods like G-NIA and \( G^2 \text{A2C} \) cannot achieve satisfactory attack performance even when the injected ratio is large. Since Nettack, AFGSM, G-NIA, and \( G^2 \text{A2C} \) are extended from targeted attack scenarios, it is possible that they may not be suitable for global attacks considered in our problem. For instance, the classification performance of GCN on the perturbed graph after we adopted G-NIA is even worse than the classification performance on the clean graph. Moreover, the proposed GANI can obtain the best attack performance almost in all injection ratios, which is consistent with the above analysis.

In addition, Figs. 4–6 also draw a similar conclusion as above. The proposed GANI can achieve an overall better attack performance in most cases in the remaining three target GNNs. Though Nettack and G-NIA perform slightly better than GANI...
Table III

| Datasets | Attack | Clean | Nettack | META | AFGSM | G-NIA | G^2A2C | TDGIA | GANI | GANI(L) |
|----------|--------|-------|--------|------|-------|-------|--------|-------|------|---------|
| Cora     | GCN    | 83.60 | 81.83  | 81.00 | 82.40 | 81.32 | 81.32  | 82.97 | 82.66 | 77.25   |
|          | SGC    | 83.40 | 81.21  | 81.29 | 81.61 | 80.10 | 82.26  | 82.55 | 78.06 | 75.39   |
|          | Jaccard| 83.55 | 81.84  | 81.58 | 81.90 | 81.52 | 82.99  | 82.69 | 78.35 | 75.46   |
|          | SimPGCN| 8294  | 81.18  | 80.94 | 81.14 | 82.27 | 81.55  | 83.84 | 77.92 | 71.12   |
| Citeseer | GCN    | 72.87 | 70.80  | 71.49 | 71.73 | 71.46 | 71.98  | 71.74 | 70.27 | 68.70   |
|          | SGC    | 72.48 | 70.75  | 71.25 | 70.79 | 67.08 | 71.66  | 70.68 | 70.51 | 68.62   |
|          | Jaccard| 72.94 | 70.96  | 72.01 | 71.76 | 71.50 | 72.60  | 71.08 | 70.53 | 68.46   |
|          | SimPGCN| 75.47 | 73.87  | 72.73 | 74.15 | 74.38 | 74.42  | 74.21 | 70.80 | 70.22   |
| Cora-ML  | GCN    | 85.36 | 84.23  | 85.29 | 85.19 | 84.39 | 84.77  | 84.22 | 80.82 | 78.30   |
|          | SGC    | 84.80 | 83.16  | 84.50 | 83.30 | 83.75 | 84.43  | 83.59 | 80.76 | 77.44   |
|          | Jaccard| 85.39 | 84.20  | 85.23 | 85.15 | 84.62 | 84.91  | 84.27 | 80.94 | 78.32   |
|          | SimPGCN| 84.85 | 83.44  | 84.32 | 84.55 | 84.63 | 85.04  | 83.93 | 81.03 | 79.40   |
| PubMed   | GCN    | 86.49 | 84.98  | 85.29 | 85.19 | 84.39 | 84.77  | 84.22 | 80.82 | 80.98   |
|          | SGC    | 85.16 | 83.15  | 84.22 | 83.01 | 83.59 | 84.33  | 83.59 | 81.19 | 77.00   |
|          | Jaccard| 86.52 | 85.02  | 85.81 | 87.21 | 87.10 | 87.66  | 87.51 | 81.96 | 81.30   |
|          | SimPGCN| 87.88 | 87.30  | 87.51 | 87.64 | 87.69 | 87.46  | 87.30 | 87.30 | 86.71   |

Average Ranking / 2.69 6.83 4.63 4.00 6.22 4.13 1.09 /

Note: GANI (L) refers to the results of GANI with ground truth label. The best result of each row except for method GANI(L) is boldfaced. Here, ‘OOM’ denotes out of memory.

Fig. 3. Change in accuracy (in percent) of GCN w.r.t. different ratios of injected nodes (1%–10%) in four datasets: (a) Cora; (b) Citeseer; (c) Cora-ML; and (d) PubMed.

Fig. 4. Change in accuracy (in percent) of SGC w.r.t. different ratios of injected nodes (1%–10%) in four datasets: (a) Cora; (b) Citeseer; (c) Cora-ML; and (d) PubMed.

in a few cases, their performance fluctuates on a large scale among different GNNs and datasets.

3) With or Without Label Information: In the default setting of GANI, we only utilize the predicted label obtained from surrogate SGC model $M_s$ to provide the label information. We want to verify whether the ground truth label can help to improve the attack performance. Therefore, we replace the predicted label of SGC with the ground truth label on the optimization of GANI. By adopting this change, our GANI can obtain correct label information on feature generations,
calculation of DNH, and elite selections. Based on the results [labeled as GANI(L)] from the last column in Table III and Figs. 3–6, we know that the ground truth label information can largely improve the attack performance as it supplies a 100% correct guideline for GANI. This also indicates that the label information plays a critical role in the attack procedure. The proposed method performs better if there are more accurately labeled data.

4) Unnoticeable Perturbations: Then, we analyze whether the adversarial graphs obtained from the proposed GANI are unnoticeable from both topology and feature aspects. Specifically, we plot the degree distribution under different injection ratios and the t-SNE [47] visualization of features. As shown in Fig. 7, the degree distributions of adversarial graphs after injecting malicious nodes under different ratios are similar to the degree distributions of clean graphs. Moreover, the adversarial graphs have a lower frequency for lower degree nodes than the clean graphs, showing that GANI prefers to connect the newly injected node with those nodes having a relatively lower degree. The above finding is also consistent with the fact that it is harder to influence the representation or classification of high-degree nodes than low-degree nodes.

Fig. 8 shows the corresponding t-SNE visualization comparison between the generated features and original features. Since our GANI generates the adversarial features based on the statistical feature information of different classes, the adversarial nodes with the same assigned label will have the same features. Therefore, the number of the clusters of injected nodes in Fig. 8 represents the number of classes. We can observe that the generated features are projected into a similar latent place as the original features, rather than some outliers.

Besides testing the imperceptibility of GANI in terms of both degree distribution and feature visualization, we also want to investigate whether the injected nodes generated from GANI are detectable when a detection algorithm is employed. Therefore, we further verify the imperceptibility of GANI from the
of the clean graph, and graph outlier detection methods could hardly detect the injected nodes. Thus, GANI is unnoticeable enough and suitable for large ratio injections.

5) Case Study: To make a better comparison between the proposed GANI and previous strategies, we also conduct a case study on Cora-ML dataset on both topology and feature domains. For the topology domain, we select two methods from target and global NIA scenarios including AFGSM and TDGIA and compare the degree distribution of adversarial graphs generated by GANI and them. Specifically, original TDGIA and AFGSM assume the degree of all injected nodes as the average degree of the original graph. As shown in Fig. 9(a), we can find that, compared to AFGSM and TDGIA, the degree distribution of GANI is closer to the degree distribution of the clean dataset without being attacked. AFGSM and TDGIA have large gaps on several degree values, especially for values closer to the average degree of Cora-ML dataset (i.e., 5).

For the feature domain, we compare the feature generation of GANI and other two commonly used mechanisms in previous articles, namely the average noise- and binary noise-based generation, which are shown in Fig. 9(b)–9(d). We can easily find that the features generated via average noise and binary noise are more likely to be projected on the fringe area of the latent space, especially for the average noise, while the features generated via GANI are closer to the main area. The above visualizations of our case study support that the proposed GANI can retain a better imperceptibility than previous methods on both topology and feature aspects.

6) Ablation Study: Finally, to further verify the effectiveness of GANI, we analyze the influence of each part of GANI through ablation study. As our GANI majorly includes the feature generation module and neighbor selection module, we conduct an ablation study by removing one or all of them to verify their power on NIA. The specific different combinations are as follows: Random indicates we will randomly decide
the features and neighbors of the adversarial nodes. GANI_{links} indicates we will retain the feature generation part of GANI, but randomly design the adversarial links on the neighbor selection procedure. GANI_{features} represents that we will randomly assign the adversarial features on the feature generation part, but retain the GA-based neighboring selection procedure.

As shown in Table V, compared with the feature generation part, the neighbor selection step contributes more to the final performance. The above results are reasonable since the adversarial link will decide the final aggregation target of nodes. If we cannot generate high-quality adversarial links, the malicious features will fail to contribute to attack performance. Also, the overall better performance of GANI indicates that the combination of appropriate features and neighbors can help to improve the final performance. Particularly, we find that GANI with random features slightly outperforms the original GANI on PubMed dataset. We believe the possible reason is that this setting (i.e., GANI_{features}) may randomly select the nodes whose nonzero indices have far exceeded our feature-attack budget. Combining with our GA-based neighbor selection mechanism, this setting is possible to obtain comparable performance to GANI.

VI. CONCLUSION

In this work, we study the global adversarial attacks on GNNs via node injections. Compared with previous studies, we design a more imperceptible yet effective node injection method, GANI, by considering the topology properties and the feature consistent aspects. Specifically, we first propose a degree sampling operation to retain the structural similarity of graphs. For the feature generation step, considering the difference between the binary and continuous features, we utilize the statistical information of nodes in each class to make the fake nodes similar to the original nodes in the feature domain. For the neighbor generation step, we select the optimal neighbors via the evolutionary perturbations obtained from GA. Moreover, we induce a two-level sorting mechanism considering both the basic fitness function and the decrease of NH on the sorting procedure to further improve our method. Extensive experimental results demonstrate the superiority of the proposed GANI on the attack performance for different kinds of GNN models and the imperceptibility on both structure and feature properties. Particularly, the corresponding property investigation and outlier detection on the newly injected nodes also illustrate the imperceptible effect of the proposed method for the unnoticeable injections.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2020.
[2] J. Zhou et al., “Graph neural networks: A review of methods and applications,” AI Open, vol. 1, pp. 57–81, 2020.
[3] Z. Zhang, P. Cui, and W. Zhu, “Deep learning on graphs: A survey,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 1, pp. 249–270, Jan. 2022.
[4] S. Wu, F. Su, W. Zhang, and B. Cui, “Graph neural networks in recommender systems: A survey,” 2020, arXiv:2011.02260.
[5] W. Jiang and J. Luo, “Graph neural network for traffic forecasting: A survey,” 2021, arXiv:2101.11174.
[6] H. Cai, V. W. Zheng, and K. C.-C. Chang, “A comprehensive survey of graph embedding: Problems, techniques, and applications,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 9, pp. 1616–1637, Sep. 2018.
[7] W. Fan et al., “A graph neural network framework for social recommendations,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 5, pp. 2033–2047, May 2022.
[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in Proc. Int. Conf. Learn. Representations, 2015, pp. 220–225.
[9] L. Sun et al., “Adversarial attack and defense on graph data: A survey,” 2020, arXiv:1812.10528.
[10] W. Jin et al., “Adversarial attacks and defenses on graphs: A review, a tool and empirical studies,” 2020, arXiv:2003.00653.
[11] L. Chen et al., “A survey of adversarial learning on graphs,” 2020, arXiv:2003.05730.
[12] S. Freitas, D. Yang, S. Kumar, H. Tong, and D. H. Chau, “Graph vulnerability and robustness: A survey,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 6, pp. 5915–5934, Jun. 2023, doi: 10.1109/TKDE.2022.3163672.
[13] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in Proc. KDD, 2018, pp. 2847–2856.
[14] D. Zügner and S. Günnemann, “Adversarial attacks on graph neural networks via meta learning,” in Proc. Int. Conf. Learn. Representations, 2018.
[15] H. Dai et al., “Adversarial attack on graph structured data,” in Proc. Int. Conf. Mach. Learn., PMLR, 2018, pp. 1115–1124.
[16] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, “Fast gradient attack on network embedding,” 2018, arXiv:1809.02797.
J. Chen et al., “MGA: Momentum gradient attack on network,” IEEE Trans. Comput. Social Syst., vol. 8, no. 1, pp. 99–109, Feb. 2021.

H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, “Geon-GCN: Geometric graph convolutional networks,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 7793–7804.

Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, “Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach,” in Proc. WWW, 2020, pp. 673–683.

J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng, “Scalable attack on graph data by injecting vicious nodes,” Data Min. Knowl. Discovery, vol. 34, no. 5, pp. 1363–1389, 2020.

J. H. Holland, “Genetic algorithms,” Sci. Amer., vol. 267, no. 1, pp. 66–73, 1992.

A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Automating the construction of internet portals with machine learning,” Inf. Retr., vol. 3, no. 2, pp. 127–163, 2000.

J. Chen et al., “GA-based Q-attack on community detection,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 1, pp. 82–95, Jan. 2023, doi: 10.1109/TKDE.2021.3078755.

J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, “Beyond homophily in graph neural networks: Current limitations and effective designs,” in Proc. Conf. Neural Inf. Process. Syst., 2020.

Junyuan Fang (Student Member, IEEE) received the B.Eng. degree in software engineering from Guangdong University of Technology, Guangzhou, China, in 2018, and the M.Eng. degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2020. He is currently working toward the Ph.D. degree in electrical engineering with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong SAR, China. His current research interests include robustness analysis and optimization, applications of network science, and graph mining.

Haizian Wen received the B.S. degree in information and computing science from Jinan University, Guangzhou, China, in 2020, and the M.Eng. degree in computer technology from Sun Yat-sen University, Guangzhou, China, in 2022. He is a Software Engineer at ByteDance, Shenzhen, China. His research interests include blockchain, robustness analysis, and applications of network science.

Jiaying Wu (Senior Member, IEEE) received the B.Eng. degree in communication engineering from Beijing Jiaotong University, Beijing, China, in 2010, and the Ph.D. degree in electrical and information engineering from The Hong Kong Polytechnic University, Hong Kong, China, in 2014. She is currently an Associate Professor with the School of Software Engineering, Sun Yat-sen University, Zhuhai, China. Her research interests include blockchain, graph mining, and network science.

Dr. Wu was awarded the Hong Kong Ph.D. Fellowship Scheme during her Ph.D. study in Hong Kong (2010–2014).
Qi Xuan (Senior Member, IEEE) received the B.S. and Ph.D. degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. He was a Postdoctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, from 2008 to 2010, respectively, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China, in 2010 and 2017. From 2012 to 2014, he was a Postdoctoral Fellow with the Department of Computer Science, University of California at Davis, CA, USA. He is currently a Professor with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. His current research interests include network science, graph data mining, and deep learning.

Zhibin Zheng (Fellow, IEEE) received his B.Eng. and M.Phil. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2005 and 2007, respectively and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, China, in 2010. He is currently a Professor and the Deputy Dean of the School of Software Engineering, Sun Yat-sen University, Guangzhou, China. He authored or coauthored more than 200 international journal and conference papers, including one ESI hot paper and ten ESI highly cited papers. According to Google Scholar, his papers have more than 34,000 citations. His research interests include blockchain, software engineering, and services computing.

Chi K. Tse (Fellow, IEEE) received the B.Eng. (Hons.) degree with first class honors in electrical engineering and the Ph.D. degree in electrical engineering from the University of Melbourne, Melbourne, Australia, in 1987 and 1991, respectively. He is currently an Associate Vice President (Innovation) and the Chair Professor in electrical engineering with the City University of Hong Kong, Hong Kong SAR, China. His research interests include power electronics, nonlinear systems, and complex network applications.

Dr. Tse was the recipient of a number of research and invention prizes, including the IEEE CASS Charles A. Desoer Technical Achievement Award in 2022 and a few best paper prizes from IEEE and other journals, as well as a Grand Prize and Gold Medal in Silicon Valley International Invention Festival 2019. He has been appointed to honorary professorship and distinguished fellowship by a few Australian, Canadian, and Chinese universities, including the Chang Jiang Scholar Chair Professor with Huazhong University of Science and Technology, the Honorary Professor with Melbourne University, and the Distinguished Professor-at-Large with the University of Western Australia. He also served as a Panel Member of Hong Kong Research Grants Council, and a Member of several professional and government committees. In 2005, 2010, and 2018, he was selected as an IEEE Distinguished Lecturer. In 2006, he chaired the IEEE CAS Technical Committee on Nonlinear Circuits and Systems. He serves and has served as the Editor-in-Chief of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2016–2019), IEEE Circuits and Systems Magazine (2013–2016), and IEICE Nonlinear Theory and Applications (since 2013); as an Associate Editor of a few other IEEE journals; and on the Editorial Board of Proceedings of the IEEE (2021–2022). He has served on a number of IEEE committees, including the IEEE Fellows Committee and the IEEE Awards Committee, and chaired the Steering Committee for IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.