Cross-Stage Reward-Aware Node Injection Attacks against Graph Neural Networks for Fraud Detection

1st Weiqi Kang
Software Engineering Institute
East China Normal University
Shanghai, China
kangweiqi@stu.ecnu.edu.cn

2nd Linghao Ying
Software Engineering Institute
East China Normal University
Shanghai, China
51265902007@stu.ecnu.edu.cn

3rd Li Han*
Software Engineering Institute
East China Normal University
Shanghai, China
hanli@sei.ecnu.edu.cn

***Abstract*—Graph Neural Networks (GNNs) have become an effective tool for financial fraud detection. However, beyond continuously evolving fraud patterns in real transaction systems, GNN-based fraud detectors are increasingly exposed to adversarial attacks, where attackers deliberately manipulate transaction graphs to evade detection. In realistic financial scenarios, existing GNN attack methods are often ineffective, due to strict constraints such as limited access to model parameters and training data, dynamic transaction networks, extreme sparsity of fraud behaviors and severe class imbalance. To address these challenges in financial scenarios, we propose Cross-stage Reward-aware Node Injection (CRNI), a black-box node injection attack for realistic financial scenarios. Under the condition that only partial graph features, limited local structure information, and model outputs can be accessed, CRNI formulates node injection as a cross-stage optimization problem with stage-specific learning signals. To enhance the concealment and effectiveness of the attack, CRNI jointly optimizes feature generation and structural connection strategies through reward modeling and topological consistency constraints, while incorporating stable training and sample reuse mechanisms to alleviate the problems of strategy fluctuations and data imbalance. Experimental results show that CRNI achieves better performance than the state-of-the-art methods on multiple real datasets and GNN models.**

***Index Terms*—Graph neural network, adversarial attack, fraud detection, reinforcement learning**

I. Introduction

In recent years, graph neural networks (GNNs) have demonstrated remarkable capabilities in financial fraud detection [1]. By leveraging transaction records, user relationships, and behavioral patterns, GNNs can capture potential fraud features and help identify disguised fraudsters. However, GNN-based fraud detectors suffer from inherent vulnerabilities, such as their heavy reliance on graph structure and node features, which can be deliberately manipulated by fraudsters to evade detection [2], [3], thereby posing threats to real-world financial systems and causing significant economic losses. Therefore, it is necessary to study the attack methods against GNN-based fraud detectors to reveal their potential risks.

To expose these vulnerabilities, researchers have developed various methods. Early research focuses on directly modifying features or structures [4], [5]. However, such methods usually require the fraudsters to have access to the complete data, which is difficult to achieve in financial scenarios. To improve feasibility, the node injection attack [6]–[8] is proposed, which reduces the reliance on system permissions and enhances the practical feasibility. Despite this advantage, the existing methods usually rely on the structural information of the target model or require training proxy models. In financial scenarios, detection models are usually deployed as core systems and their structural details and parameter settings are confidential, making it difficult for attackers to obtain them. Besides, there are often differences between the real detection model and the proxy model, which can lead to a decrease in the effectiveness. Therefore, previous studies [9]–[11] have attempted to introduce advantage actor-critic to enhance the adaptability of the attack, utilize graph contrastive learning to enhance the efficiency of initial queries or optimize the characteristics of the injection nodes. However, due to the high dynamism of the transaction network, the sparsity of fraud behaviors and the imbalance in their categories, they either suffer from unstable strategy updates, poor adaptability to evolving environments, and ineffective utilization of limited training samples, or adopt the edge connection strategy that neglects the roles of other nodes in the subgraph, which may weaken the attack effectiveness.

To overcome the limitations of existing research in our scenario, we propose CRNI, a black-box node injection attack framework. Under realistic black-box constraints where the attacker can only access partial node features and limited local structural information, together with model outputs, we formulate the node injection process as a cross-stage optimization problem with stage-specific learning signals, rather than relying on sparse and delayed feedback induced solely by structural perturbations. This optimization is implemented within a Markov decision process to support sequential injection actions. To enhance attack effectiveness, we propose a cross-stage reward modeling mechanism, which takes statistical consistency and concealment as the optimization objectives, and applies them simultaneously to the decision stages. In the node generation stage, the reward provides attackers with dense and learnable feedback, enabling them to directly optimize the rationality of adversarial nodes in the feature distribution. In the edge connection stage, the reward works in combination with structural strategy, allowing the connec-

*Corresponding author.

tion decisions to improve the attack effect while maintaining the fit to real transaction network patterns. Furthermore, we introduce topological consistency constraints to characterize the similarity of candidate and injected nodes in the local neighborhood structure, thereby guiding the learning to more concealed and realistic connection strategies. In response to the characteristics of scarce samples and strong imbalance in financial graphs, we propose a training mechanism centered on sample reuse and stable strategy updates to alleviate strategy fluctuations under black-box interaction and improve the utilization efficiency of limited data samples. Our main contributions include:

- We formulate black-box node injection attacks against financial fraud detection systems as a cross-stage sequential decision-making problem, and propose CRNI, a reward-aware, policy-driven attack framework operating under realistic black-box constraints.
- We design a feature–structure collaborative attack mechanism, enhanced by cross-stage reward modeling and topological consistency constraints, which jointly optimize adversarial node generation and edge connection strategies to balance attack effectiveness and concealment.
- We introduce a training mechanism based on sample reuse and stable strategy updates, enabling CRNI to effectively handle severe data sparsity and class imbalance in financial graphs while reducing strategy fluctuations under black-box interactions.
- Extensive experimental results show that on multiple real-world datasets and GNN models, CRNI demonstrates superior performance compared to the existing state-of-the-art methods.

II. Methodology

Our proposed method, CRNI, is illustrated in Figure 1. CRNI employs a node generation module to generate deceptive and aggressive node features and employs an edge connection module to select the optimal connection edges which maximizes the misleading effect on the target model.

A. MDP for Node Injection

State. At timestamp t, $s_t \in S$ contains the modified graph $G'_t = (V'_t, E'_t, X'_t)$. To better simulate real-world attack scenarios, node generation module and adversarial edge connection module focus on the k-hop subgraph $G'_t(v)$ of the target node v.

Action. The node injection involves two key steps: node generation and edge connection. The attack process begins with generating fraudulent nodes, and then iteratively performs edge connection operations within the graph topology. The attack will terminate when any of the following conditions is met: the fraud detection model fails to identify the injected nodes, or the attack budget ω is exhausted. Formally, at timestamp t, we denote the fraudulent node generation, edge connection and action reward as a_t^n, a_t^e, r_t, respectively. The whole pipeline of proposed MDP is $(s_0, a_0^n, r_0, s_1, a_1^e, r_1, s_2, .., s_T)$, where s_T is the terminal state.

Reward. In our framework, we have designed reward mechanisms separately for the node generation and the edge connection operation, in order to effectively guide the optimization of the attack strategy. The reward for the node generation a_t^n can be denoted as:

$$r_n = \exp(-\beta \cdot \mathcal{L}_f(x_a)), \tag{1}$$

where β is the sensitivity coefficient, $\mathcal{L}_f(x_a)$ is the loss function for the feature space. At timestamp t, the reward for the adversarial edge wiring a_t^e can be denoted as:

$$r_e = \mathcal{L}(v_t, G'_{t+1}) - \mathcal{L}(v_t, G'_t), \tag{2}$$

where G'_{t+1} is the attacked graph after wiring a_t^e to G'_t. The reward function uses the classification loss to motivate CRNI to make decisions that have a greater impact on the victim model's classification. Meanwhile, if the target node is successfully misclassified after the attack, the agent will receive additional reward r_{add} to motivate it to avoid detection. The final reward is $r_t = r_n + r_e + r_{add}$.

B. Model Framework

Node Generation Module. This module aims to inject the adversarial node v_a embedded with malicious feature vector x_a into the k-hop subgraph $G'(v)$ of the target node v, thus inducing the fraud detection model to misclassify v. Simultaneously, we must ensure that the generated x_a preserve statistical consistency with the original distribution X. In this module, firstly, in order to better utilize the graph information in the current state s_t, we first extract the topology and feature information of s_t through stacked graph convolutional layers (GCLs). We define the convolution layer of node generation module as follows:

$$\mathbf{H}_n^{(k+1)} = f_{GCL}(\widetilde{\mathbf{A}}_v, \mathbf{H}_n^{(k)}, \mathbf{W}^{(k)}), \tag{3}$$

where $\mathbf{H}_n^{(0)} = X'$, $\widetilde{\mathbf{A}}_v$ is the normalized adjacency matrix of $G'(v)$, $\mathbf{W}^{(k)}$ is the weight of k-th convolution layer, $\mathbf{H}_n^{(k+1)}$ represents the embedding of state s_t after the k-th layer of the GCLs. Then $\mathbf{H}_n^{(k+1)}$ is applied to summarize the feature distribution z_n through a readout operation. For discrete feature space, the subsequent process is followed by the use of the Gumbel-Softmax technique to generate feature x_a. The discrete x_a can be calculated as follows:

$$x_a = \text{round}(\sigma((\log z_n + G_b)/\tau)), \tag{4}$$

where round$(\cdot)$ enables the process to be trainable, $\sigma(\cdot)$ denotes the Sigmoid activation function, z_n represents the feature distribution of $G'(v)$, $G_b \sim Gumbel(0, 1)$ denotes a random variable and τ denotes a temperature coefficient. Furthermore, we propose a loss function for the feature space, which can be denoted as follows:

$$\mathcal{L}_f(x_a) = (\|x_a - \widetilde{X}\|/\|X\|)^2, \tag{5}$$

where $\widetilde{X}$ denotes the mean value of node features in the subgraph. For continuous feature space, first, the variable z_n is mapped to the mean and standard deviation parameter μ_n, σ_n of a normal distribution through two learnable weight matrices

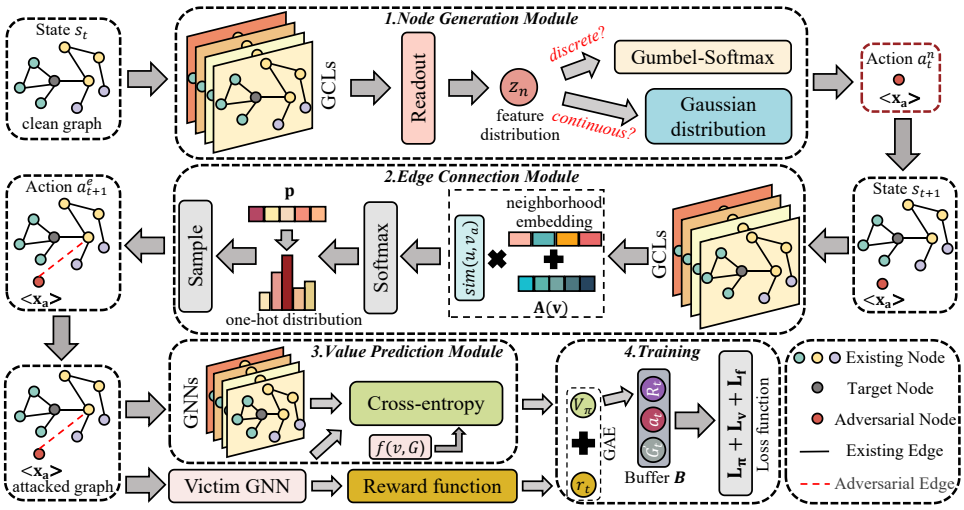


Fig. 1: Overview of CRNI. It models the attack as a Markov Decision Process, where the agent performs three-step actions: node generation, edge connection and value prediction.

$\mathbf{W}_\mu, \mathbf{W}_\sigma$. Then, features are generated based on the parametric Gaussian distribution, denoted as:

$$x_a \sim \mathcal{N}(\mu_n, \sigma_n^2), \tag{6}$$

Also, we propose a loss function for the feature space, denoted as:

$$\mathcal{L}_f(x_a) = \mathbb{E}_{x_a \sim \mathcal{N}(\mu_n, \sigma_n^2)}[\log \frac{\mathcal{N}(x_a; \mu_n, \sigma_n^2)}{\mathcal{N}(x_a; \mu_x, \sigma_x^2)}] + \lambda \|x_a - \mu_x\|_2^2, \tag{7}$$

where $\mathbb{E}_{x_a \sim \mathcal{N}(\mu_n, \sigma_n^2)}[\cdot]$ indicates the expected value on the generated distribution, $\mathcal{N}(x_a; \mu_n, \sigma_n^2)$ and $\mathcal{N}(x_a; \mu_x, \sigma_x^2)$ individually represent the probability density of the feature distributions of the generated nodes and the real nodes, λ is the L2 regularization coefficient.

Edge Connection Module. This module determines which nodes in the current graph will be connected to the adversarial node v_a, it also adopts stacked GCLs denoted as $\mathbf{H}_e$, the calculation method is similar to that of $\mathbf{H}_n$. The base score is denoted as:

$$g_{v_a} = (\mathbf{H}_e^{(k+1)} \oplus x_a) \cdot \mathbf{W} + \alpha \cdot \mathbf{A}(v), \tag{8}$$

where $\mathbf{W}$ is the learnable parameter matrix, $\oplus$ denotes the concatenation operation, α is a learnable parameter, $\mathbf{A}(v)$ denotes the adjacency information of the target node in the graph. To prevent the generation process from resulting in connection patterns that do not conform to the actual network structure, such as nodes being overly connected to irrelevant isolated nodes in the structure, thereby reducing the overall recognizability and concealment, we introduce topological consistency constraints to guide the connection to better conform to the patterns of real transaction networks, denoted as:

$$sim(u, v_a) = \frac{|\mathcal{V}(u) \cap \mathcal{V}(v_a)|}{|\mathcal{V}(u) \cup \mathcal{V}(v_a)|}, \tag{9}$$

where u is the candidate node and $\mathcal{V}(\cdot)$ is the set of neighbors. The connection probability between v_a and the candidate nodes in the current graph is calculated using the following formula:

$$\mathbf{p} = \text{Softmax}(g_{v_a} \cdot (1 + \rho sim(u, v_a))/\tau), \tag{10}$$

where ρ is topological consistency weight and τ is temperature hyperparameter. Based on the one-hot category distribution parameterized by $\mathbf{p}$, we sample one edge and incorporate it into the current graph to obtain the updated graph G'_{t+1}.

Value Prediction Module. This module is responsible for evaluating the performance of the current strategy and providing value estimates. First, we employ a GNN model with k-layers stacking to capture the node topological information, which is represented as $\mathbf{H}_v$. Then, we extract the embedding of v and connect it with the output of the victim model f to predict the value score:

$$V_\pi(s_t) = \mathcal{L}_{CE}((\mathbf{H}_v^{(k+1)}(v) \oplus f(v, G'_t)) \cdot \mathbf{W}', f(v, G)), \tag{11}$$

where $\mathcal{L}_{CE}(\cdot, \cdot)$ denotes the cross-entropy loss function, and $\mathbf{W}'$ is the learnable parameter matrix.

Training Algorithm. For training CRNI, we employ the experience buffer $\mathcal{B}$, which consists of tuples in the format (G'_t, a_t, R_t). We use the Generalized Advantage Estimation (GAE) to evaluate the advantage of taking action a_t in state s_t compared to the average behavior of the policy, denoted as:

$$A_t = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_{t+l}, \tag{12}$$

where γ is the discount factor, and δ_t denotes the TD residual form of the advantage function, formulated as: $\delta_t = r_t + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)$. Then, the expected total return R_t can be calculated as: $R_t = A_t + V_\pi(s_t)$. During the strategy update process, we use the importance sampling ratio $r_t(\theta)$ to measure the difference between the old strategy $\pi_{\theta_{old}}(a_t|s_t)$ and new strategy $\pi_\theta(a_t|s_t)$, formulated as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}. \tag{13}$$

Subsequently, we employ the clip function [12] to limit the range of the importance sampling ratio, in order to prevent

TABLE I: Statistics of the four datasets: S-FFSD, YelpChi, Cora and Citeseer

Dataset	Type	Node	Edge	Fraud
S-FFSD	Fraud detection network	130,840	3,492,226	2,950
YelpChi	Fraud detection network	45,954	7,739,912	6,677
Cora	Citation network	2,708	5,429	-
Citeseer	Citation network	3,327	4,432	-

the excessive update amount of the strategy. Then, the strategy loss function can be denoted as:

$$\mathcal{L}_\pi(t) = -\mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t], \quad (14)$$

where ϵ is the clipping coefficient. Furthermore, the value loss function evaluates and optimizes the prediction error of the value prediction module, formulated as:

$$\mathcal{L}_v(t) = (V_\pi(t) - R_t)^2 \quad (15)$$

The overall loss function is denoted as follows:

$$\mathcal{L} = \sum_{\mathcal{B}}(\mathcal{L}_\pi(t) + \mathcal{L}_v(t)) + \sum_{x_a \in X'_t} \mathcal{L}_f(x_a). \quad (16)$$

III. Experiments

A. Experiment Settings

Datasets. We evaluate the node injection attack method on four real datasets: S-FFSD [13], YelpChi [14], Cora and Citeseer [15]. Among them, S-FFSD and YelpChi are typical datasets for fraud detection. Cora and Citeseer are standard citation network datasets without fraud semantics, included to evaluate whether CRNI generalizes beyond financial fraud graphs under black-box settings. The statistical information of the four datasets is shown in Table I.

Baseline methods. In our experimental framework, we compare our method with various graph neural networks and adversarial attack methods. The victim models include four types of GNNs: GCN [16], GAT [17], GraphSAGE [18], SGC [19], as well as four fraud detection models: CARE-GNN [20], PC-GNN [21], BWGNN [22], and GTAN [13]. In terms of node injection attacks, we select five of the latest methods: G-NIA [6], TDGIA [23], G^{2}A2C [9], GCIA [10], TEANI [11].

Experimental Configuration. In CRNI, we set the sensitivity coefficient to 0.4, the number of GCLs layers and k to 3, the topological consistency weight to 0.3, the temperature hyperparameter of the Gumbel-Softmax to 1.0, the discount factor to 0.95, the clipping coefficient to 0.1, the hidden dimension to 256, and λ in GAE to 0.99. The AdamW optimizer is employed with a learning rate of 10^{-4}. Additionally, an early stopping mechanism is incorporated during the training process, with a patience period of 5 epochs. Both the baseline method and the victim model use their open-source default configurations. For papers that do not provide the public code implementation or detailed parameter settings, this study strictly replicates the baseline methods based on their description and sets them according to the common practices in the relevant field to ensure the stability and fairness of the experimental results.

B. Performance of Fraud Attacks

Tables II report the comparison between our proposed CRNI and multiple baselines across different datasets and victim models. On the financial fraud detection datasets, CRNI achieves the highest misclassification rate in most cases, maintaining a high attack effect on complex fraud detectors. In contrast, existing baselines exhibit various limitations: G-NIA relies on mean feature representation and ignores node interactions; TDGIA overlooks edges among injected nodes, limiting coordinated subgraph construction; The reward of G^{2}A2C only applies to the edge connection actions, lacking the return from the node generation stage. By combining A2C and experience replay, the strategy learning signal becomes sparse and the sample efficiency is low, making training unstable. GCIA's optimization target is not fully aligned with the final classification outcome, and its modeling of structural relationships among injected nodes is insufficient. Moreover, it frequently encounters OOM issues on large graphs, revealing limited scalability. TEANI restricts learnable connection patterns by defaulting injected–target node connections. In contrast, CRNI jointly models node generation and edge connection, and separately designs effective reward mechanisms, enabling the coordinated optimization of feature consistency and structural misleadingness. Moreover, topological consistency constraints enhance the concealment and structural rationality of the injected subgraph while ensuring the effectiveness of the attack. Additionally, by introducing policy stability constraints and sample reuse mechanisms, CRNI improves learning stability and sample efficiency, enabling it to maintain a higher misclassification rate. To verify whether CRNI can be generalized beyond fraud-related graphs, we conduct further experiments on two non-fraud citation network datasets using standard GNNs. The results in III show that CRNI also achieves the best performance across most models, further verifying its universality and robustness in non-fraud tasks.

C. Ablation Study

To evaluate the contribution of each component in CRNI, we perform ablation experiments by separately removing or randomizing the node generation and edge connection modules, yielding four variants: random_a (random node generation), random_e (random edge connection), w/o stab.& reuse and CRNI. Table IV presents the classification error rates of four datasets under three GNN models. It can be observed that the complete CRNI achieves the highest classification error rate across all datasets and models, significantly outperforming the two variants. Meanwhile, the performance of random_e is generally better than random_a, indicating that in the node injection attack, realistic feature generation is more critical than precise edge patterns. However, both variants are far inferior to the complete model, confirming that the joint optimization of node generation and edge connection is essential for enhancing stealthiness and maximizing attack strength. Moreover, the sample reuse & strategy update mechanism helps maximize the utilization of the samples and enhances stability.

TABLE II: Misclassification rate (%) of injection attack across various GNNs on S-FFSD and YelpChi (attack budget: one adversarial node and one edge). Rate in bold indicates the best and rate in underline is the second best. OOM: Out of Memory.

Dataset	Method	Surrogate / Victim Model							
		GCN	GAT	GraphSAGE	SGC	CARE-GNN	PC-GNN	BWGNN	GTAN
S-FFSD	Clean	37.86	38.07	34.04	37.54	18.68	31.39	35.90	15.12
	G-NIA	43.11	78.68	72.86	43.95	19.09	37.10	38.19	18.36
	TDGIA	44.92	74.26	58.15	42.44	20.15	37.23	36.22	16.89
	G^2A2C	50.90	94.72	95.84	48.47	23.14	41.53	42.51	23.25
	GCIA	38.35	93.40	97.20	41.15	OOM	OOM	43.78	24.62
	TEANI	48.55	90.16	88.60	46.90	25.10	40.84	38.96	20.50
	CRNI	**54.04**	**97.36**	**98.42**	**52.00**	**28.01**	**46.43**	**45.97**	**28.10**
YelpChi	Clean	43.35	47.31	34.55	14.18	29.52	28.40	17.09	11.79
	G-NIA	65.48	69.51	53.88	14.35	32.11	33.80	20.31	14.99
	TDGIA	59.65	59.05	42.10	14.25	30.50	30.55	22.12	11.87
	G^2A2C	73.72	91.49	64.78	14.53	33.92	35.72	25.90	17.98
	GCIA	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TEANI	70.49	86.75	60.71	14.40	35.66	36.98	26.00	16.00
	CRNI	**77.00**	**93.98**	**68.50**	**14.55**	**36.64**	**40.11**	**29.07**	**20.68**

TABLE III: Misclassification rate (%) of further attack experiments on Cora and Citeseer. Rate in bold indicates the best and rate in underline is the second best.

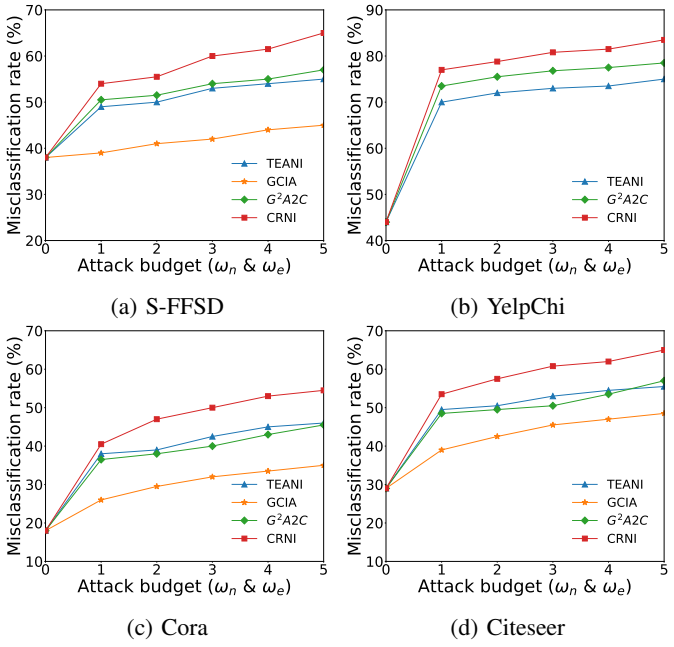
Dataset	Attack Method	Surrogate / Victim Model			
		GCN	GAT	GraphSAGE	SGC
Cora	Clean	18.32	16.81	17.15	15.84
	G-NIA	24.98	23.06	25.46	26.44
	TDGIA	29.54	28.37	24.77	35.20
	G^2A2C	36.36	33.48	26.36	41.63
	GCIA	26.10	23.20	**32.10**	17.20
	TEANI	38.32	35.17	27.79	40.20
	CRNI	**41.12**	**37.45**	30.65	**43.90**
Citeseer	Clean	28.70	27.50	23.45	21.50
	G-NIA	36.50	34.78	43.71	48.95
	TDGIA	44.22	52.10	41.46	51.24
	G^2A2C	48.44	55.71	46.86	53.81
	GCIA	39.04	39.40	43.79	29.60
	TEANI	49.42	58.00	47.00	51.34
	CRNI	**53.50**	**59.77**	**50.03**	**56.80**

TABLE IV: Misclassification rate (%) of ablation studies of CRNI on four datasets. Rate in bold indicates the best.

Dataset	Method	Surrogate / Victim Model		
		GCN	GAT	GraphSAGE
S-FFSD	Clean	37.86	38.07	34.04
	random_a	39.30	44.32	45.96
	random_e	41.18	49.74	50.50
	w/o stab.& reuse	51.68	94.44	95.97
	CRNI	**54.04**	**97.36**	**98.42**
YelpChi	Clean	43.35	47.31	34.55
	random_a	47.38	49.02	38.79
	random_e	51.42	51.49	40.10
	w/o stab.& reuse	75.40	92.24	66.77
	CRNI	**77.00**	**93.98**	**68.50**
Cora	Clean	18.32	16.81	17.15
	random_a	20.30	19.11	19.84
	random_e	23.65	21.52	22.24
	w/o stab.& reuse	38.02	34.95	28.80
	CRNI	**41.12**	**37.45**	**30.65**
Citeseer	Clean	28.70	27.50	23.45
	random_a	32.75	30.30	25.39
	random_e	35.03	33.44	28.96
	w/o stab.& reuse	50.65	57.43	48.11
	CRNI	**53.50**	**59.77**	**50.03**

D. Budget Analysis

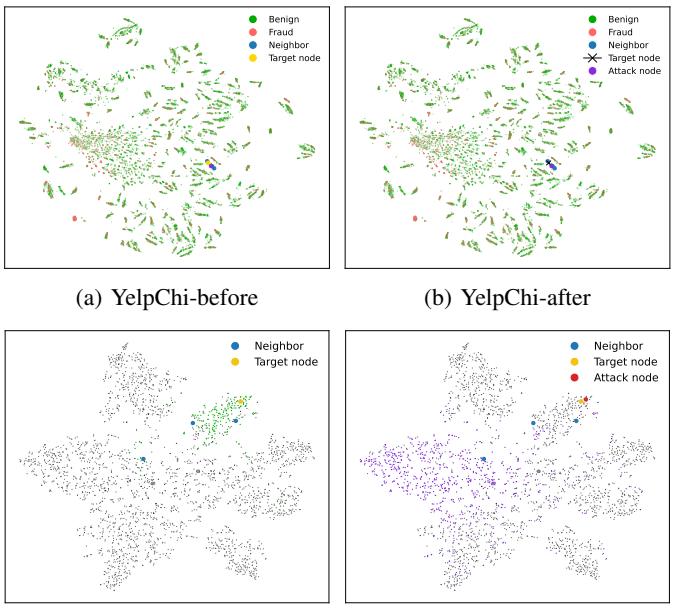
To investigate how CRNI preforms under different budget constraints, we evaluate its attack effectiveness across varying budgets (node budget ω_n and edge budget ω_e). The experimental results are shown in the Figure 2. The vertical axis represents the misclassification rate (%), the horizontal axis represents the attack budget. In the experiment, the node budget ω_n and the edge budget ω_e remain consistent. From the overall trend, the misclassification rates increase for all methods as the budget grows, indicating that higher budgets provides attackers with greater operational space, thereby enhancing the attack effectiveness. Specifically, CRNI demonstrates the best attack performance on all four datasets, indicating that CRNI can effectively utilize the limited attack resources to achieve more attack effects. Moreover, under low budget settings, CRNI still maintains strong attack capabilities, demonstrating its robustness and practicality in resource-constrained scenarios.



(a) S-FFSD (b) YelpChi

(c) Cora (d) Citeseer

Fig. 2: Attack performance and comparison under different budgets on four datasets, with GCN as the target model.



(a) YelpChi-before (b) YelpChi-after

(c) Cora-before (d) Cora-after

Fig. 3: The t-SNE visualization of potential node representations on YelpChi and Cora before and after the attack, computed by GCN.

E. Case Study

To visually illustrate the attack mechanism of CRNI, we conduct a case study on YelpChi and Cora using GCN as the surrogate model. Figure 3 shows the t-SNE [24] visualization of node representations before and after the attack. On YelpChi, green and red nodes represent benign and fraudulent users, with the target node marked by yellow circle before the attack and by black cross after the attack, the neighbors of the target node highlighted in blue, and the injected node shown in purple. On Cora, the target node before and after the attack is indicated by yellow circle, with its neighbors highlighted in blue. In the left subfigure, green nodes denote nodes from the original class of the target, while in the right subfigure, the injected node is marked in red and purple nodes represent the new class. As shown in Figure 3, CRNI affects the embedding of the target node by jointly perturbing its local topological structure and feature representation, causing it to migrate to the incorrect category region. Moreover, the topological consistency constraint ensures that the injected nodes have a connection pattern that more closely resembles the structural characteristics of the real network. This avoids abnormal connections with irrelevant isolated nodes, while maintaining strong attack effectiveness and concealment.

IV. Conclusion

This study explored the evasion attack achieved by injecting fraudulent nodes in a black-box environment. We proposed CRNI, which formalized the process of fraud injection as a MDP and implemented it using graph reinforcement learning, without relying on prior knowledge of the victim model. The model effectively achieved evasion by generating undetectable but aggressive node features and embedding them into the graph structure. Experiments on multiple datasets showed that CRNI outperforms existing methods.

Acknowledgement

The work was supported by the National Key R&D Program of China (2022YFC3302603), the National Natural Science Foundation of China 62202168.

References

[1] Fu, Kang, et al. "Credit card fraud detection using convolutional neural networks." International conference on neural information processing. Cham: Springer International Publishing, 2016.
[2] Bojchevski, Aleksandar, and Stephan Günnemann. "Adversarial attacks on node embeddings via graph poisoning." International conference on machine learning. PMLR, 2019.
[3] Dai, Hanjun, et al. "Adversarial attack on graph structured data." International conference on machine learning. PMLR, 2018.
[4] Zügner, Daniel, Amir Akbarnejad, and Stephan Günnemann. "Adversarial attacks on neural networks for graph data." Proceedings of the 24th ACM SIGKDD international conference. 2018.
[5] Zhao, Kaifa, et al. "Structural attack against graph based android malware detection." Proceedings of the 2021 ACM SIGSAC conference on computer and communications security. 2021.
[6] Tao, Shuchang, et al. "Single node injection attack against graph neural networks." Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021.
[7] Wang, Jihong, et al. "Scalable attack on graph data by injecting vicious nodes." Data Mining and Knowledge Discovery 34.5 (2020): 1363-1389.
[8] Sun, Yiwei, et al. "Node injection attacks on graphs via reinforcement learning." arXiv preprint arXiv:1909.06543 (2019).
[9] Ju, Mingxuan, et al. "Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning." Proceedings of the AAAI conference on artificial intelligence.
[10] Liu, Xiao, Jun-Jie Huang, and Wentao Zhao. "Gcia: A black-box graph injection attack method via graph contrastive learning." International Conference on Acoustics, Speech and Signal Processing. 2024.
[11] Zhao, Maochang, and Jing Zhang. "Highly Imperceptible Black-Box Graph Injection Attacks with Reinforcement Learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 39. No. 12. 2025.
[12] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).
[13] Xiang, Sheng, et al. "Semi-supervised credit card fraud detection via attribute-driven graph representation." Proceedings of the AAAI conference on artificial intelligence. Vol. 37. No. 12. 2023.
[14] Rayana, Shebuti, and Leman Akoglu. "Collective opinion spam detection: Bridging review networks and metadata." Proceedings of the 21th acm sigkdd international conference. 2015.
[15] Sen, Prithviraj, et al. "Collective classification in network data." AI magazine 29.3 (2008): 93-93.
[16] Kipf, T. N. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
[17] Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).
[18] Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017).
[19] Wu, Felix, et al. "Simplifying graph convolutional networks." International conference on machine learning. Pmlr, 2019.
[20] Dou, Yingtong, et al. "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters." Proceedings of the 29th ACM international conference. 2020.
[21] Liu, Yang, et al. "Pick and choose: a GNN-based imbalanced learning approach for fraud detection." Proceedings of the web conference. 2021.
[22] Tang, Jianheng, et al. "Rethinking graph neural networks for anomaly detection." International conference on machine learning. PMLR, 2022.
[23] Zou, Xu, et al. "Tdgia: Effective injection attacks on graph neural networks." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021.
[24] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9.Nov (2008): 2579-2605.