AUC-oriented Graph Neural Network for Fraud Detection

Mengda Huang[†]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
huangmengda19s@ict.ac.cn

Yang Liu[†]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
liuyang17z@ict.ac.cn

Xiang Ao[*†]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
aoxiang@ict.ac.cn

Kuan Li[†]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
likuan20s@ict.ac.cn

Jianfeng Chi
Alibaba Group
Hangzhou, China
bianfu.cjf@alibaba-inc.com

Jinghua Feng
Alibaba Group
Hangzhou, China
jinghua.fengjh@alibaba-inc.com

Hao Yang
Alibaba Group
Hangzhou, China
youhiroshi.yangh@alibaba-inc.com

Qing He[*†]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
heqing@ict.ac.cn

ABSTRACT

Though Graph Neural Networks (GNNs) have been successful for fraud detection tasks, they suffer from imbalanced labels due to limited fraud compared to the overall userbase. This paper attempts to resolve this label-imbalance problem for GNNs by maximizing the AUC (Area Under ROC Curve) metric since it is unbiased with label distribution. However, maximizing AUC on GNN for fraud detection tasks is intractable due to the potential polluted topological structure caused by intentional noisy edges generated by fraudsters. To alleviate this problem, we propose to decouple the AUC maximization process on GNN into a classifier parameter searching and an edge pruning policy searching, respectively. We propose a model named *AO-GNN* (Short for AUC-oriented GNN), to achieve AUC maximization on GNN under the aforementioned framework. In the proposed model, an AUC-oriented stochastic gradient is applied for classifier parameter searching, and an AUC-oriented reinforcement learning module supervised by a surrogate reward of AUC is devised for edge pruning policy searching. Experiments on three real-world datasets demonstrate that the proposed AO-GNN patently outperforms state-of-the-art baselines in not only AUC but also other general metrics, e.g. F1-macro, G-means.

[*]Corresponding author.
[†]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS). Also at University of Chinese Academy of Science. Xiang Ao is also at Institute of Intelligent Computing Technology, Suzhou, China.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512178

CCS CONCEPTS

• **Computing methodologies Neural networks**;

KEYWORDS

Graph Neural Networks, AUC Maximization, Fraud Detection

ACM Reference Format:
Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. AUC-oriented Graph Neural Network for Fraud Detection. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512178

1 INTRODUCTION

Benefited by the capability of mining profound dependence and correlations between entities, Graph Neural Networks (GNNs) [12, 33] have been broadly implemented in fraud detection tasks [7, 16, 21, 48, 50, 52]. In particular, due to its message passing scheme, which aggregates and transforms the representation of neighbors for each node recursively, GNN is competent to discover subtle and hidden signals over graphs, and categorize some ambiguous cases.

Despite achieving remarkable success on fraud detection tasks, GNN-based models for fraud detection yet suffer from the imbalanced label distribution problem, i.e. the fraudsters are far less than benign users [20]. The label imbalance problem is fairly ordinary to be observed in most fraud-related scenarios. For example, in the real-world dataset, *Amazon* [25], only 9.5% of accounts post spam reviews. Another example, the dataset in [50] contains only 0.5% financial defaulter who failed to repay the debt on time.

Under such an adverse condition, models are prone to overfitting in one class as a consequence of skewed data distribution [13]. Hence some works attempt to alleviate the impacts of imbalanced label distribution from the perspective of embellishing training procedures. To name some, re-sampling methods construct a balanced training set by oversampling the minority [41, 44] or undersampling

the majority [23, 30]. Re-weighting methods weigh each training case in a meta-learning [31, 51] or cost-sensitive way [14, 21].

In this paper, unlike existing works, we attempt to resolve the label imbalance problem by maximizing the AUC (Area Under ROC Curve) metric. Since AUC is unbiased with label distribution [24], AUC-oriented training tends to obtain a model with the competitive ability for classifying both majority (benign users) and minority (fraud users) samples. Though there are pioneer works designed for Euclidean space data [9, 28, 42, 43], specific solutions for non-Euclidean graph structure data are still under-explored [34].

However, maximizing AUC on GNN for fraud detection tasks is intractable due to the potential polluted topological structure caused by intentional noisy edges generated by fraudsters. Recall that a central node collects feature vectors from all its neighbor nodes during the message passing in GNN models. As a result, some fraudsters may intentionally form noisy edges, e.g. connect with the benign users as much as possible, to mislead the classifier returning incorrect output. For example, for each fraud node in the *Amazon* [25], 80.47% of its neighbors are benign on average. Hence, purifying the graph structure and filtering neighbor set appropriately by emerging graph structure learning [4, 47] are helpful for adapting AUC maximization to GNN-based fraud detection.

Applying graph structure learning to AUC maximization is also unmanageable for hinders from two aspects. On the one hand, the lack of supervised information highly increases the hardship of training a neighbor filter with generalizability over all nodes. On the other hand, learning to choose neighbors for each node adaptively is of high time complexity in dense or large-scale graphs.

To address the above challenges, we propose *AO-GNN*, an AUC-oriented GNN model for fraud detection, which consists of a node classifier and an edge pruner. Then we can decouple AUC maximization on graphs into a classifier parameter searching (classifier training) and an edge pruning policy searching (pruner training), respectively. The former mitigates the problem of label imbalance, and the latter policy searching is designed for graph noise removal.

The edge pruner chooses neighbor set(s) for each node particularly and further advances the message passing process, which is omitted in existing AUC maximization approaches. Specifically, we formulate the process of edge pruning (a.k.a choosing neighbor) for a central node as a Markov Decision Process (MDP). Conducted by Policy Gradient [36], our policy is capable of boosting AUC. Importing reinforcement learning into our framework is natural and plausible, for not only does reinforcement learning require no noise-related supervised data, but also keeps a consistent optimizing target if we set AUC variation as the reward. Furthermore, we devise surrogate reward, parameter sharing mechanism and halting mechanism to decrease time consumption, which accelerates the training by two orders of magnitude.

The contribution of this paper could be summarized as follows:

- We propose a novel GNN-based model for fraud detection from the standpoint of AUC maximization to resolve the problem of imbalanced label distribution and noisy topological structure. To the best of our knowledge, this is the first work to maximize AUC on graph data.
- We formulate neighbors choosing problem as an MDP with a theoretical guarantee of maximizing AUC and solve it by Deep Reinforcement Learning. Enhanced by our accelerating mechanisms, our edge pruning policy searching module is effective and efficient.
- Experiments on three public datasets demonstrate that *AO-GNN* clearly outperforms the state-of-the-art baselines that we are aware of.

2 PRELIMINARIES

2.1 Problem statement

2.1.1 Multi-relational imbalanced graph. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y})$ be an undirected graph. Specifically, vector set $\mathcal{V}$ consists of n nodes $\mathbf{v}_i \in \mathcal{V}$; $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_r\}$ is a set of edge sets from r distinct relations; $\mathcal{X}$ is a matrix of node feature, in which the matching row vector $X_{\mathbf{v}}$ is the corresponding feature vector of $\mathbf{v}$; $\mathcal{Y}$ is the label set in which $y_{\mathbf{v}} \in \{0, 1\}$ is the label of $\mathbf{v}$. In the context of fraud detection, $y_{\mathbf{v}} = 1$ represents fraud, otherwise $\mathbf{v}$ is a benign. Informally, if samples from one class appear much more frequently than another, then we call $\mathcal{G}$ as a multi-relational imbalanced graph.

2.1.2 AUC. Let $\mathcal{M}_{\mathcal{G}} : \mathcal{V} \rightarrow \mathbb{R}$ be a GNN-based model over multi-relational imbalanced graph $\mathcal{G}$. Parameterized by $\boldsymbol{\omega} \in \mathbb{R}^d$, $\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega}; \mathbf{v})$ returns the probability that $\mathbf{v}$ is a fraud by processing and aggregating information from $\mathbf{v}$'s neighbor set $N(\mathbf{v})$. As a result of message passing scheme in GNN, trimming on $N(\mathbf{v})$ would influence the prediction output of $\mathcal{M}_{\mathcal{G}}$. Therefore, given a pruning policy $\boldsymbol{\Pi} : 2^{\mathcal{V}} \rightarrow 2^{\mathcal{V}}$ which returning trimmed neighbor set $\boldsymbol{\Pi}(N(\mathbf{v})) \subseteq N(\mathbf{v})$, we can have a new prediction $\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega}; \mathbf{v}|\boldsymbol{\Pi})$.

Then we can calculate the AUC metric of $\mathcal{M}_{\mathcal{G}}$ under pruning policy $\boldsymbol{\Pi}$ according to the definition of AUC on a population level:

$$AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\boldsymbol{\Pi}) = \mathbb{P}(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega}; \mathbf{v}|\boldsymbol{\Pi}) \geq \mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega}; \mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}} = 1, y_{\mathbf{v}'} = 0). \quad (1)$$

In Eq (1), $\mathbf{v}, \mathbf{v}'$ are drawn from all positive/negative labeled samples independently. Eq (1) explains AUC as the probability that a random fraud case is more likely to be classified as a fraudster than a random benign case. The pairwise focus on benign users and fraudsters is helpful to explain the label distribution non-bias feature of AUC.

2.1.3 Problem formulation. Given a multi-relational imbalanced graph $\mathcal{G}$ and a GNN-based model $\mathcal{M}_{\mathcal{G}}$ with random parameters, the AUC maximization problem is to find optimal parameters $\boldsymbol{\omega}^*$ for $\mathcal{M}_{\mathcal{G}}$ and optimal $\boldsymbol{\Pi}^*$ simultaneously by solving

$$\max_{\boldsymbol{\omega}, \boldsymbol{\Pi}} \mathbb{E}_{\mathbf{v}, \mathbf{v}'}(\mathbb{I}(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega}; \mathbf{v}|\boldsymbol{\Pi}) \geq \mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega}; \mathbf{v}'|\boldsymbol{\Pi}))|y_{\mathbf{v}} = 1, y_{\mathbf{v}'} = 0), \quad (2)$$

where $\mathbb{I}$ is indicator function and $\mathbf{v}, \mathbf{v}'$ are drawn from $\mathcal{V}$ independently. Note that Formula (2) only transforms the right side of Eq (1) from probability to expectation for easier obtaining estimation.

2.2 Graph Neural Network

The core component of GNN is the message passing scheme, which aggregates and transforms the representation vectors of neighbors for each node recursively. Formally, let $\mathbf{H}^{(l)}_{\mathbf{v}}$ be the representation of $\mathbf{v}$ in the l-th layer of GNN, then we can update the representation in the $(l+1)$-th layer as

$$\mathbf{H}^{(l+1)}_{\mathbf{v}} = \sigma(W^{(l)} \underset{u \in N(\mathbf{v})}{\mathbf{AGGR}}(\mathbf{MSG}(\mathbf{H}^{(l)}_{\mathbf{v}}; \mathbf{H}^{(l)}_{u}))), \quad (3)$$

where $\mathbf{H}_{v_i}^{(0)} = X_i$; σ is an activating function; $W^{(l)}$ is the trainable parameter in the l-th layer; **AGGR** denotes aggregator function and **MSG** denotes the message passing function [45]. Following the ground work in Eq (3), many GNN architectures have been put forward, among which the most representative are GCN[12], GraphSAGE [10] and GAT[37], etc. In this paper, we choose GraphSAGE with relation-aware concatenation as our base architecture for the purpose of inductive learning and efficiency. The updating procedure can be written as

$$\mathbf{H}_{\mathbf{v},k}^{(l+1)} = \frac{1}{|N_k(\mathbf{v})|} \sum_{u \in N_k(\mathbf{v})} \mathbf{H}_u^{(l)}, \tag{4}$$

$$\mathbf{H}_{\mathbf{v}}^{(l+1)} = \sigma(W^{(l)}(\mathbf{H}_{\mathbf{v}}^{(l)} \oplus \mathbf{H}_{\mathbf{v},1}^{(l+1)} \oplus \cdots \oplus \mathbf{H}_{\mathbf{v},r}^{(l+1)})). \tag{5}$$

In Eq (4) and Eq (5), neighbor set $N(\mathbf{v})$ is divided into r neighbor set $N_1(\mathbf{v}), N_2(\mathbf{v}), ..., N_r(\mathbf{v})$ according to different relations in the edge set $\mathcal{E}$ of $\mathcal{G}$, and $\oplus$ stands for vector concatenation.

2.3 Reinforcement Learning

2.3.1 Markov Decision Process. An RL task could be regarded as a Markov Decision Process (MDP) and solved by maximizing the expected reward [35]. An MDP $\mathcal{P}$ can be described as a quadruple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. Precisely, $\mathcal{S}$ is the state set; $\mathcal{A}$ is the action set; $\mathcal{P}$ is a function returning the probability of transiting to s' given s, a, denoted as $\mathcal{P}(s'|s,a)$; $\mathcal{R}: \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ is reward function, which delivers reward r after the transition $s \to s'$ finished. In an MDP, we make a decision to take an action a at current state s, and transit to a new state s', recursively.

An agent for solving MDP could be represented as a policy $\pi: \mathcal{S} \to \mathcal{A}$. In state s, $\pi(a|s)$ returns the probability of choosing a as action. Given a policy π, we are able to evaluate the goodness of a state or action by predicting *return* G_t from timestep t, defined as $\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ where $\gamma \in (0, 1]$ is the discount factor. The discount factor weighs the future reward and instant reward. To be specific, the goodness of s, a under policy π is calculated by state-value function $\mathbf{V}^{\pi}(s) = \mathbb{E}_{\pi}(G_t|S_t = s)$ and $\mathbf{Q}^{\pi}(s,a) = \mathbb{E}_{\pi}(G_t|S_t = s, A_t = a)$. The goal of reinforcement learning is to discover an optimal policy π^* by solving:

$$\arg\max_{\pi} \sum_{s \in \mathcal{S}} d^{\pi}(s)\mathbf{V}^{\pi}(s)\,, \tag{6}$$

where $d^{\pi}(s)$ is the stationary distribution of s given a policy π.

2.3.2 Policy Gradient. Considering the MDP we are about to construct, where $\mathcal{S}$ is infinite and $\mathcal{A}$ only contains deleting or not deleting a neighbor, we can build a policy network π_θ parameterized by θ to model a policy, which takes representation vector of s as input and output $\pi_\theta(a|s)$ for each possible a. Instead of backward gradient from a loss function, gradient for policy network is differentiated from Formula (6), which is the goal of RL. The detailed gradient expression is given by Policy Gradient [36]:

$$\nabla_\theta = \mathbb{E}(\mathbf{Q}^{\pi}(s,a)\nabla \ln \pi_\theta(a|s)). \tag{7}$$

To perform Eq (7), we could produce trajectories, shaped like "$S_0, A_0, S_1, R_1, A_1, S_2, R_2, A_2, ..., S_n, R_n$", by sampling action from $\pi_\theta(A_t|S_t)$ recursively given an initial state S_0.

3 METHODOLOGY

In this section, we first introduce the framework of *AO-GNN*, which decouples AUC maximization on GNN into classifier parameter searching and edge pruning policy searching. Then we provide more detailed descriptions of both modules and our efforts on efficiency.

3.1 Framework of Model

Recall the AUC maximization problem formulated in Formula (2). Actually $\boldsymbol{\omega}$ can determine a unique node classifier GNN (hereafter called classifier) and $\boldsymbol{\Pi}$ represents an edge pruner (hereafter called pruner). Therefore, the AUC maximization problem is searching a classifier and a pruner to achieve the goal proposed in Formula (2). However, it is rather difficult to optimize both $\boldsymbol{\omega}$ and $\boldsymbol{\Pi}$ at once, because training them needs distinct materials. As the classifier should be trained with nodes in graph and their labels; while the edge pruner should be trained with trajectories we simulated. As shown in Alg. 1, to solve this problem, we decide to train the classifier and pruner asynchronized by decoupled optimization, which transforms the original problem into two subproblems (Line 4,5 of Alg. 1). In Line 3 of Alg. 1, the break condition could be set by either a certain number of iterative rounds or a threshold of AUC variation indicating convergence.

Algorithm 1: AUC maximization with GNN

input : A multi-relation graph $\mathcal{G}$, a GNN architecture $\mathcal{M}_{\mathcal{G}}$

1 Initialize $\mathcal{M}_{\mathcal{G}} \leftarrow \mathcal{M}_{\mathcal{G}}^{\omega}$ with random parameters;
2 Initialize $\boldsymbol{\Pi}$ as a random function;
3 **while** *Break Condition is False* **do**
4 $\quad \boldsymbol{\omega} \leftarrow \arg\max_{\boldsymbol{\omega}} AUC(\mathcal{M}_{\mathcal{G}}^{\omega}|\boldsymbol{\Pi})$; // Parameter searching
5 $\quad \boldsymbol{\Pi} \leftarrow \arg\max_{\boldsymbol{\Pi}} AUC(\mathcal{M}_{\mathcal{G}}^{\omega}|\boldsymbol{\Pi})$; // Policy searching
6 **end**

return : A GNN $\mathcal{M}_{\mathcal{G}}^{\omega}$, Pruning policy $\boldsymbol{\Pi}$

As a greedy algorithm, our framework is able to maintain a monotonous growth on AUC because we fix one part of *AO-GNN* when optimizing another part. To avoid falling into local optima, we make efforts for both modules, which will be further explained.

In the following subsections, we will show that this training schema is synergistic: classifier provides a preciser reward for pruner training and pruner provides cleaner topological structure as the training material for the classifier.

3.2 Classifier Parameter Searching

In this subsection, we aim at solving Line 4 in Alg. 1, i.e. $\arg\max_{\boldsymbol{\omega}} AUC(\mathcal{M}_{\mathcal{G}}^{\omega}|\boldsymbol{\Pi})$ via searching $\boldsymbol{\omega}$, which represents parameters of the classifier. The architecture of our classifier is illustrated in the left part of Fig. 1. Assume we use existing edge pruning policy $\boldsymbol{\Pi}$ to prune central node $\mathbf{v}$'s neighbor sets $N_1(\mathbf{v}), ..., N_r(\mathbf{v})$ for r diverse relations. Then we use pruned neighbor sets $\boldsymbol{\Pi}(N_1(\mathbf{v})), ..., \boldsymbol{\Pi}(N_r(\mathbf{v}))$ to generate layer-wise hidden embeddings $\mathbf{H}_{\mathbf{v}}^{(\cdot)}$ recursively by Eq (4). After inputting hidden embedding of last layer $\mathbf{H}_{\mathbf{v}}^{(l)}$ into an MLP, we can access the predictive result and loss function $\mathcal{L}_{AUC}$ specified by us.

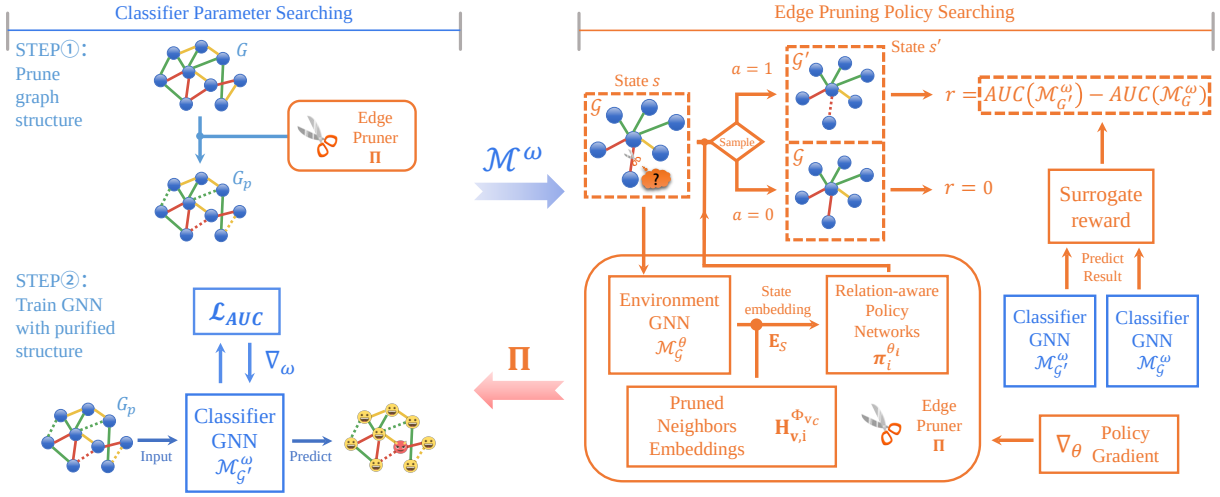


Figure 1: Model training schematic of *AO-GNN*. Dashed lines denote pruned edges.

The core of traditional parameter searching is optimizing a loss function following the philosophy of empirical risk minimization (ERM). To maximize a metric in parameter searching, the most straightforward idea is designing a loss function derived from the unbiased estimation of the metric to be optimized. For instance, cross-entropy loss is designed to minimize Kullback–Leibler divergence; Dice loss [26] is evolved from the unbiased estimation of F1-score to maximize F1-score directly. Similarly, we can devise a loss function to maximize AUC based on Formula (2). As the indicator function in Formula (2) is not consistent [46], we can replace it as l_2 convex surrogates. We eliminate l_1 loss for its statistically inconsistency with AUC [9]. Then optimizing problem can be written as a loss function by minimization:

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^d} \mathbb{E}_{\mathbf{v},\mathbf{v}'}[(1-(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi}))^2|y_{\mathbf{v}}=1,\, y_{\mathbf{v}'}=0]. \quad (8)$$

However, Formula (8) requires coupled samples with different labels as input which is not convenient to manufacture mini-batch gradient. To make the optimization batch-friendly so that it is applicable to graph with large scale, we deconstruct and convert it into a saddle point optimization problem [2].

Theorem 1. *The optimizing problem in Formula (8) is equivalent to*

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^d,\,\{a,b\}\in\mathbb{R}^2} \max_{\alpha\in\mathbb{R}} \mathbf{E}_{\mathbf{v}}[\mathcal{L}_{AUC}(\boldsymbol{\omega},a,b,\alpha,\mathbf{v}|\boldsymbol{\Pi}|p)], \quad (9)$$

where p is the ratio of fraud nodes, and

$$\begin{aligned}\mathcal{L}_{AUC}(\boldsymbol{\omega},a,b,\alpha,\mathbf{v}|\boldsymbol{\Pi},p) &= \mathbb{I}(y=1)[(1-p)(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-a)^2\\ &\quad + 2(p-1)(1+\alpha)\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})]\\ &\quad + \mathbb{I}(y=0)[p(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-b)^2\\ &\quad + 2p(1+\alpha)\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})] + p(1-p)\alpha^2.\end{aligned}$$

Remark. *This theorem is similar to Theorem 1 in [46], and we present an understandable proof and the reason why we import learnable aid parameters a, b, α in the appendix.*

For simplicity, we define $\mathbf{m} = [\boldsymbol{\omega}, a, b] \in \mathbb{R}^{d+2}$. Intuitively, $\mathbf{m}$ is updated by stochastic gradient descend, while α is updated by stochastic gradient ascend. Unlike traditional loss function only calling for minimization, optimization problem described in Formula (9) is allocated as a saddle point optimization problem, which is well-studied by academia. Therefore, as shown in Alg. 2, we employ proximal primal-dual stochastic gradient (PPD-SG) [18] to solve Formula (9) which achieves the best time complexity at present.

Algorithm 2: Solving $\arg\max_{\boldsymbol{\omega}} AUC(\mathcal{M}_{\mathcal{G}}^{\boldsymbol{\omega}}|\boldsymbol{\Pi})$

input : A multi-relation graph $\mathcal{G}$, train set $\mathcal{V}_t$, validation set $\mathcal{V}_v$ drawn from node set $\mathcal{V}$ of $\mathcal{G}$, learning rate η_0, number of inner iteration T_0, a GNN architecture $\mathcal{M}_{\mathcal{G}}$, an edge pruning policy $\boldsymbol{\Pi}$

1 Initialize $\overline{\mathbf{m}}_0 \in \mathbb{R}^{d+2}$ randomly with constraint $||\mathbf{m}||_2 < \epsilon$, $\overline{\alpha}_0 \leftarrow 0$;
2 **for** $k = 1, \ldots, K$ **do**
3 $\mathbf{m}_0^k \leftarrow \overline{\mathbf{m}}_{k-1}$, $\alpha_0^k \leftarrow \overline{\alpha}_{k-1}$, $\eta_k \leftarrow 3^{1-k}\eta_0$, $T_k = 3^{k-1}T_0$;
4 **for** $t = 1, \ldots, T_k$ **do**
5 Draw a batch of nodes $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_b\}$ from $\mathcal{V}_t$;
6 Prune for each $\mathbf{v}_i$'s neighbor in each relation $N_1(\mathbf{v}_i) \leftarrow \boldsymbol{\Pi}(N_1(\mathbf{v}_i)), \ldots, N_r(\mathbf{v}_i) \leftarrow \boldsymbol{\Pi}(N_r(\mathbf{v}_i))$;
7 Calculate $\mathcal{M}_{\mathcal{G}}(\overline{\boldsymbol{\omega}}_{t-1}^k;\mathbf{v}_i|\boldsymbol{\Pi})$ for each $\mathbf{v}_i$;
8 Calculate $\mathcal{L}_{AUC}(\boldsymbol{\omega}, a, b, \alpha, \mathbf{v}|\boldsymbol{\Pi}, p)$ in this batch by (9);
9 $\mathbf{m}_t^k \leftarrow \mathbf{m}_{t-1}^k - \eta_k[\nabla_m \mathcal{L}_{AUC} + \lambda(\mathbf{m}_{t-1}^k - \mathbf{m}_0^k)]$;
10 $\alpha_t^k \leftarrow \alpha_{t-1}^k - \eta_k[\frac{\partial \mathcal{L}_{AUC}}{\partial \alpha} + \lambda(\alpha_{t-1}^k - \alpha_0^k)]$;
11 **end**
12 $\overline{\mathbf{m}}_k \leftarrow \frac{1}{T_k}\sum_{i=1}^{T_k}\mathbf{m}_i^k$; // $\overline{\boldsymbol{\omega}}_k$ is part of $\overline{\mathbf{m}}_k$
13 $\overline{\alpha}_k \leftarrow \frac{\sum_{\mathbf{v}\in\mathcal{V}_v}\mathcal{M}_{\mathcal{G}}(\overline{\boldsymbol{\omega}}_k;\mathbf{v}|\boldsymbol{\Pi})\mathbb{I}(y=1)}{\sum_{\mathbf{v}\in\mathcal{V}_v}\mathbb{I}(y=1)} - \frac{\sum_{\mathbf{v}\in\mathcal{V}_v}\mathcal{M}_{\mathcal{G}}(\overline{\boldsymbol{\omega}}_k;\mathbf{v}|\boldsymbol{\Pi})\mathbb{I}(y=0)}{\sum_{\mathbf{v}\in\mathcal{V}_v}\mathbb{I}(y=0)}$;
14 **end**

return : GNN $\mathcal{M}_{\mathcal{G}}^{\overline{\boldsymbol{\omega}}_K}$

In Alg. 2, $K, T_0, \eta_0, \epsilon, \lambda$ are hyper-parameters. The effectiveness and converging property of Alg. 2 can be proved by Theorem 2 in [18], which is omitted in this paper as it is out of the scope of this paper. In Line 12 and 13, $\mathbf{m}$ and α jump from the current value to the geometric center of the last T_k traces, which prevents falling into local optima.

3.3 Edge Pruning Policy Searching

In this subsection, we focus on solving Line 5 in Alg. 1, i.e. $\arg\max_{\mathbf{\Pi}} AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\mathbf{\Pi})$ via training a pruner that is competent to purify graph topology and further boost AUC. We first demonstrate how we model edge pruning as an MDP, then prove the consistency between solving this MDP and maximizing AUC mathematically, and finally we introduce the architecture of our pruner and how we train it by policy gradient.

3.3.1 Edge pruning MDP. As aforementioned, $\mathbf{\Pi} : 2^{\mathcal{V}} \rightarrow 2^{\mathcal{V}}$ returns the pruned neighbor set for each node $\mathbf{v}$. Searching for an optimized $\mathbf{\Pi}$ is not trivial due to the discreteness of $\mathbf{\Pi}$ and the intricacy in $2^{\mathcal{V}}$ space. Considering this challenge, we split the whole pruning process conducted by $\mathbf{\Pi}$ into separate prune-or-not decision making processes controlled by $\boldsymbol{\pi}$, which return a deleting probability for each neighbor. In other words, we obtain each pruned neighbor sets $\mathbf{\Pi}(N_{()}(\mathbf{v}))$ by deleting neighbors one by one instead of screening all at once. Then the whole deleting process can be formulated as an MDP as follows.

- State space $\mathcal{S}$: The state s in $\mathcal{S}$ determined by the center node $\mathbf{v}_c$, the prune-to-be neighbor node $\mathbf{v}_p$ and the relation-aware pruned neighbor sets $\Phi_{\mathbf{v}_c} = \{\phi_1^{\mathbf{v}_c}, \phi_2^{\mathbf{v}_c}, ..., \phi_r^{\mathbf{v}_c}\}$, denoted as a triplet $(\mathbf{v}_c, \mathbf{v}_p, \Phi_{\mathbf{v}_c})$.
- Action space $\mathcal{A}$: $\mathcal{A} = \{0, 1\}$. In state $s = (\mathbf{v}_c, \mathbf{v}_p, \Phi_{\mathbf{v}_c})$, $a = 1$ represents deleting $\mathbf{v}_p$ from its relation neighbor set $N_{(\cdot)}(\mathbf{v}_c)$ and $a = 0$ represents keeping $\mathbf{v}_p$ as a neighbor.
- State transition probability space $\mathcal{P}$: Next state is determined in our problem, thus $\mathcal{P}$ is a constant for each a in state s.
- Reward function $\mathcal{R}$: The reward of a transition, $s, a \rightarrow s'$, is the variation of AUC, i.e. $AUC(\mathcal{M}^{\omega}_{\mathcal{G}'}|\mathbf{\Pi}) - AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\mathbf{\Pi})$, aroused by the action a. The new AUC is calculated by existing GNN model $\mathcal{M}_{\mathcal{G}}$. Obviously, when $a = 0$, $\mathcal{R}(s, a, s') = 0$.

Concretely, in a single decision making action, we randomly select a neighbor $\mathbf{v}_p$ from one of the centre nodes $\mathbf{v}_c$'s relation-aware neighbor set $N_{(\cdot)}(\mathbf{v}_c)$, then we delete it from neighbor set with probability of $\boldsymbol{\pi}(a = 1|s = (\mathbf{v}_c, \mathbf{v}_p, \Phi_{\mathbf{v}_c}))$. If we do prune $\mathbf{v}_p$, it would be added into the corresponding $\phi^{\mathbf{v}_c}_{(\cdot)}$. By executing single pruning action recursively on non-repeatable neighbor node, we can generate a series of new relation-aware neighbor sets $\mathbf{\Pi}(N_{(\cdot)}(\mathbf{v}_c)) \subseteq N_{(\cdot)}(\mathbf{v}_c)$.

In this case, we can reproduce the edge pruning policy $\mathbf{\Pi}$ by a policy $\boldsymbol{\pi}$ for MDP we formulated. In view of we have r specific relations in $\mathcal{G}$, we can further appoint r isolated policies $\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_r$ to generate deleting probability for edges in each relation. After traversing each neighbor by corresponding relation-aware policy $\boldsymbol{\pi}_{(\cdot)}$ to inquire pruning or not, we can determine all relation neighbor sets of the central nodes. Formally, as the i-th relation neighbor set of $\mathbf{v}_c$, $\mathbf{\Pi}_{\boldsymbol{\pi}_1,...,\boldsymbol{\pi}_r}(N_i(\mathbf{v}_c))$ could be determined by $\boldsymbol{\pi}_i$.

After explaining how $\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_r$ define a $\mathbf{\Pi}$, we must present the equivalence between solving above MDP and AUC-oriented edge pruning policy searching:

$$\begin{aligned} &\arg\max_{\boldsymbol{\pi}_1,...,\boldsymbol{\pi}_r} \sum_{s\in\mathcal{S}} d^{\pi}(s)\mathbf{V}^{\pi}(s = (\mathbf{v}_c, \mathbf{v}_p, \Phi_{\mathbf{v}_c})) \\ =& \arg\max_{\boldsymbol{\pi}_1,...,\boldsymbol{\pi}_r} AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\mathbf{\Pi}_{\boldsymbol{\pi}_1,...,\boldsymbol{\pi}_r}). \end{aligned} \tag{10}$$

Theorem 2. *Under the MDP we constructed above, Eq (10) holds when the discount factor $\gamma = 1$.*

Remark. *We provide a proof to Theorem 2 in appendix. In Eq (10), the left part is derived from Eq (6), where $\boldsymbol{\pi}$ is chosen according to relation between $\mathbf{v}_c$ and $\mathbf{v}_p$. Theorem 2 indicates that policies $\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_r$ which maximize the expectation of reward in aforementioned MDP also maximize AUC.*

3.3.2 Architecture and training of the pruner. To solve this MDP, we design a pruner consists of multiple relation-aware policy networks illustrated in the lower right part of Fig. 1. In the pruner, the decision making is a two-stage process: 1) generate state embedding of s, i.e. $\mathbf{E}_s = (\mathbf{v}_c, \mathbf{v}_p, \Phi_{\mathbf{v}_c})$; 2) calculate the probability of deleting $\mathbf{v}_p$, i.e. $\boldsymbol{\pi}_{(\cdot)}(a = 1|s)$. In first stage, we use an environment GNN $\mathcal{M}^{\theta}_{\mathcal{G}}$ to create embeddings of states, motivated by [27] which use CNNs to model the state space. Based on the assumption that some pruned neighbors are similar, we add the original feature, $\mathcal{X}_{\mathbf{v}_p}$, and embeddings of pruned neighbors, $\mathbf{H}^{\Phi_{\mathbf{v}_c}}_{v,1} \oplus \cdots \oplus \mathbf{H}^{\Phi_{\mathbf{v}_c}}_{v,r}$, to help to describe the state:

$$\mathbf{H}^{\Phi_{\mathbf{v}_c}}_{v,i} = \frac{1}{|\phi_i^{\mathbf{v}_c}|} \sum_{u\in|\phi_i^{\mathbf{v}_c}|} \mathcal{X}_u, \tag{11}$$

$$\mathbf{E}_s = \mathcal{X}_{\mathbf{v}_p} \oplus \mathbf{H}^{(l)}_{\mathbf{v}_c} \oplus \mathbf{H}^{(l)}_{\mathbf{v}_p} \oplus \mathbf{H}^{\Phi_{\mathbf{v}_c}}_{\mathbf{v},1} \oplus \cdots \oplus \mathbf{H}^{\Phi_{\mathbf{v}_c}}_{\mathbf{v},r}. \tag{12}$$

In Eq (12), $\oplus$ means concatenation, and $\mathbf{H}^{(l)}_{\mathbf{v}_c}, \mathbf{H}^{(l)}_{\mathbf{v}_p}$ are calculated by Eq (4) and Eq (5) without pruning. Then we input $\mathbf{E}_s$ to matching $\boldsymbol{\pi}_i$ according to the category of edge connecting $\mathbf{v}_c$ and $\mathbf{v}_p$:

$$\boldsymbol{\pi}_i(a = 1|s) = \text{Sigmoid}(\text{MLP}(\mathbf{E}_s)). \tag{13}$$

Because $\mathcal{A} = \{0, 1\}$, $\boldsymbol{\pi}_i(a = 0|s) = 1 - \boldsymbol{\pi}_i(a = 1|s)$. And the MLP in each π_i is parameterized by θ_i.

In the training phase, we apply a classic policy gradient, REINFORCE [36] with simulated trajectories. We train all policy networks simultaneously by ascending gradients generated from Eq (7).

For each edge pruning policy searching round, we train all policy networks starting from random parameters, which avoids being trapped in local optima.

3.4 Accelerating Edge Pruning Searching

Parameter sharing. As demonstrated in Fig. 1, each policy network has its own parameters while sharing the same environment GNN, i.e. $\mathcal{M}^{\theta}_{\mathcal{G}}$. As our state embedding collects almost all possible information, it is acceptable for all policy networks to share a GNN. Also, parameter sharing significantly reduces space complexity.

Halting mechanism. We restrict the maximum number of deleting in a single relation-aware neighbor set to limit length of trajectories. When enough neighbors are pruned, the mechanism would be triggered and stop the pruning process, thus a higher halting parameter usually leads to longer trajectories. Considering that noisy edges may not account for the majority in the neighborhood of each node, the lack of halting mechanism or a high halting parameter would bring sparse reward issue [1] to the RL training. Short trajectories not only save time in producing training material but also leverage the trajectories length, which prevents the pruning policy from overfitting on nodes with large degrees.

Surrogate reward. As other metric-oriented RLs, we regard metric variation caused by action as rewards. Since our optimization is maximizing AUC on the training set, we only examine AUC variation on the training set. When an edge is cut, the graph structure is distinct. Then AUC variation is $AUC(\mathcal{M}^{\omega}_{\mathcal{G}'}|\mathbf{\Pi}) - AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\mathbf{\Pi})$, in which $AUC()$ is defined in Eq (2). Influenced by the expectation for coupled nodes, to figure out this value need $O(n \log n)$ time (n is the number of nodes in training set). Note that only one node's neighborhood is changed in one pruning action, we can simplify the AUC variation into a surrogate loss with computing complexity of $O(n)$. Say we have predictive fraudster probability set $\{P_{\mathbf{v}}|\mathbf{v} \in \mathcal{V}_{TrainingSet}\}$ for each node $\mathbf{v}$ in training set. For a node $\mathbf{v}_c$, if we cut one of its neighbor, then its predictive result evolve from $p_{\mathbf{v}_c}$ to $p'_{\mathbf{v}_c}$. The surrogate reward of this action is:

$$\mathcal{R}(s, a, s') = F(y_{\mathbf{v}_c}) \cdot \begin{cases} |\{p_{\mathbf{v}}|p_{\mathbf{v}} \in (p_{\mathbf{v}_c}, p'_{\mathbf{v}_c}], y_{\mathbf{v}} \neq y_{\mathbf{v}_c}\}|, & p'_{\mathbf{v}_c} \geq p_{\mathbf{v}_c} \\ -|\{p_{\mathbf{v}}|p_{\mathbf{v}} \in [p'_{\mathbf{v}_c}, p_{\mathbf{v}_c}), y_{\mathbf{v}} \neq y_{\mathbf{v}_c}\}|, & p'_{\mathbf{v}_c} < p_{\mathbf{v}_c} \end{cases}. \tag{14}$$

Where $F(x)$ is a sign function, defined as $F(1) = 1, F(0) = -1$.

Theorem 3. *Eq (14) is strictly proportional to AUC variation, i.e. $AUC(\mathcal{M}^{\omega}_{\mathcal{G}'}|\mathbf{\Pi}) - AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\mathbf{\Pi})$, and computing it takes $O(n)$ time.*

Remark. *We provide a proof for Theorem 3 in the appendix. This theorem implies that this surrogate reward can accelerate reward returning since it reduces the complexity from $O(nlogn)$ to $O(n)$, which is the main source of the pruner's training time.*

4 EXPERIMENTS

In this section, we implement *AO-GNN* on three public fraud detection datasets, with the intention to answer the following research questions:

- **RQ1** Does *AO-GNN* outperform state-of-the-art GNN-based fraud detection models?
- **RQ2** How significant are the classifier parameter searching and the edge pruning policy searching in boosting AUC?
- **RQ3** How does AUC evolve under the AUC-oriented parameter searching?
- **RQ4** What kind of edges are more likely to be pruned?
- **RQ5** How effective are RL accelerating mechanisms?

4.1 Experimental Setup

4.1.1 Datasets. In experiment sessions, we use three public benchmark datasets in fraud detection, *YelpChi* [7], *Amazon* [25] and *Books* [32]. The statistics of datasets are listed in Table 1, and the practical significance of our datasets is explained in Appendix B.

Table 1: Statistics of datasets.

Dataset	#Node	#Edge	Relations	Relation#Edges
YelpChi	45,954; 14.5% Fraud	3,846,979	R-U-R	49,315
			R-T-R	573,616
			R-S-R	3,402,743
Amazon	11,944; 9.5% Fraud	4,398,392	U-P-U	165,608
			U-S-U	3,566,479
			U-V-U	1,036,737
Books	1,418; 1.9%	3,695	Co-purchase	3,695

Our datasets vary in many dimensions. In terms of fraud ratio, *YelpChi* is relatively mild (14.5%) whereas *Books* is extremely imbalance (1.9%). For the average degree, we have both dense (*Amazon*, 368.25) and sparse graphs (*Books*, 2.61). Besides, we have both single relation (*Books*) and multi-relation graphs (*YelpChi*, *Amazon*). The difference among datasets could assist us to illustrate the well generalizability of *AO-GNN* in real-life applications.

4.1.2 Baselines. Besides regular GNN models (i.e. GCN [12], GraphSAGE [10] and GAT [37]), we also compare *AO-GNN* with several SOTA GNN-based fraud detection models to examine the effectiveness on fraud detection tasks. Among baselines, *DR-GCN* [34] is designed for imbalanced graph data, and *GraphConsis* [22], *CARE-GNN* [7], *PC-GNN* [20] are designed for fraud detection on graphs.

Aside from our *AO-GNN*, we also stage three variants of our model to analyze the performance of each module:

- **AO-GNN**$_{woP}$: removing the edge pruning policy searching module from original *AO-GNN*;
- **AO-GNN**$_{CE}$: switching the AUC-oriented parameter searching to Cross-Entropy-oriented searching from original *AO-GNN*;
- **AO-GNN**$_{RP}$: switching our well-trained edge pruning policy into a random pruning policy;

4.1.3 Experimental Settings. For all baselines and *AO-GNN* related models, we make a 10-run experiment to evaluate them by mean and standard deviation. The training, validation and testing set ratios are 40%, 20%, 40% with consistent fraud ratios, respectively. We set hyper-parameters in *AO-GNN* as: (1) framework (Alg. 1): we perform 3 iterations and break; (2) classifier parameter searching (Alg.2): $K = 3, T_0 = 600, \eta_0 = 0.2, \epsilon = 1e-3, b = 128, \lambda = 5e-4$; (3) edge pruning policy searching: learning rate is 0.01, and we collect 20 trajectories for each node in the training set. For each relation-aware neighbor set, we prune 10 or a half of the total members at most to trigger the halting mechanism.

For reproducibility, we present more details about implementation of *AO-GNN* in Appendix C.

4.1.4 Metrics. As the focus point of *AO-GNN*, AUC is one of our metrics in experiments. To prove that our model not only outperforms on AUC, we also use other two common metrics. One of them is the F1-macro, which is the unweighted mean of the F1-score of each class. And the other one, GMean, is defined as $\sqrt{\frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP}}$. It is necessary to mention that we do not adopt accuracy, precision, recall as measures to weigh the quality of models, since they are not applicable to imbalanced datasets.

For original GCN, GraphSAGE, CARE-GNN and DR-GCN that are suffering from label imbalance, we adjust the threshold of classification by threshold moving [6].

4.2 Performance Comparison (RQ1)

We evaluate all baselines mentioned before to answer RQ1. From the experimental results of AUC, F1-macro and GMean shown in Table 2, we have the following three observations:

First, our method outperforms not only in AUC but also in F1-macro and GMean. Meaningful increases on F1-macro and GMean

Table 2: Performance comparison for baselines, AO-GNN and its variants

Method	Dataset	YelpChi			Amazon			Books		
	Metric	AUC	F1-macro	GMean	AUC	F1-macro	GMean	AUC	F1-macro	GMean
Baselines	GCN	0.5983±0.0049	0.5620±0.0067	0.4365±0.0262	0.8369±0.0125	0.6408±0.0694	0.5718±0.1951	0.4538±0.1977	0.2374±0.2065	0.0000±0.0000
	GAT	0.5715±0.0029	0.4879±0.0230	0.1659±0.0789	0.8102±0.0179	0.6464±0.0387	0.6675±0.1345	0.4006±0.2023	0.2058±0.1623	0.0000±0.0000
	GraphSAGE	0.5439±0.0025	0.4405±0.1066	0.2589±0.1864	0.7589±0.0046	0.6416±0.0079	0.5949±0.0349	0.4761±0.1508	0.2464±0.2004	0.0000±0.0000
	DR-GCN	0.5921±0.0195	0.5523±0.0231	0.4038±0.0742	0.8295±0.0079	0.6488±0.0364	0.7963±0.0091	0.5131±0.1579	0.3048±0.2454	0.0000±0.0000
	GraphConsis	0.6983±0.0302	0.5870±0.0200	0.5857±0.0385	0.8741±0.0334	0.7512±0.0325	0.7677±0.0486	0.5647±0.1281	0.2912±0.1325	0.0000±0.0000
	CARE-GNN	0.7619±0.0292	0.6332±0.0094	0.6791±0.0359	0.9067±0.0112	**0.8990±0.0073**	0.8962±0.0018	0.6267±0.0462	0.4050±0.0996	0.4861±0.0811
	PC-GNN	0.8178±0.0014	0.6400±0.0230	0.7395±0.0130	0.9586±0.0014	0.8956±0.0077	0.9030±0.0044	0.6431±0.0189	0.4951±0.0037	0.5244±0.1012
Ablation	AO-GNN$_{woP}$	0.8680±0.0020	**0.7182±0.0177**	0.7484±0.0125	0.9588±0.0008	0.8956±0.0026	0.8740±0.0137	0.6720±0.0111	0.4131±0.0102	0.4829±0.0519
	AO-GNN$_{woC}$	0.8545±0.0177	0.7063±0.0129	0.7305±0.0241	0.9392±0.0166	0.8914±0.0041	0.8828±0.0267	0.5821±0.1397	0.2901±0.2102	0.3711±0.1919
	AO-GNN$_{R-P}$	0.8302±0.0286	0.6936±0.0351	0.7192±0.0586	0.9197±0.0238	0.8827±0.0135	0.8602±0.0164	0.5604±0.1733	0.2845±0.2329	0.3068±0.1240
Ours	AO-GNN	**0.8805±0.0008**	0.7042±0.0051	**0.8134±0.0232**	**0.9640±0.0020**	0.8921±0.0045	**0.9096±0.0105**	**0.7174±0.0158**	**0.5503±0.0141**	**0.6127±0.0252**

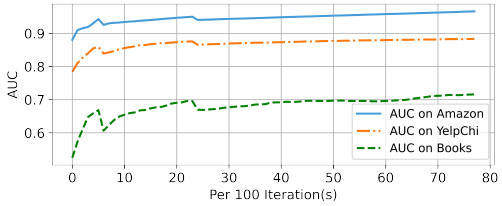


Figure 2: AUC evolving process

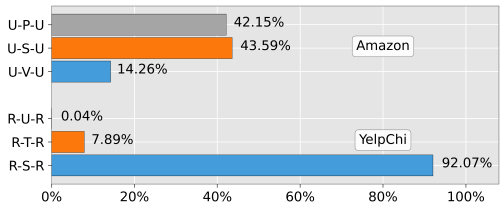


Figure 3: The relation category distribution of pruned edges

in 2 out of 3 datasets indicate that our method is practical and AUC-oriented training does empower the model comparable capability of classifying both classes. We argue that relatively low improvements in **Amazon** are because the performance of PC-GNN is high and the increasing room is limited. What's more, the feature of low AUC standard deviation suggests that *AO-GNN* is stable and reliable.

Second, the combination of AUC-oriented parameter and policy searching is the key to solving label imbalance and noisy edges problems. Compared to classic GNN models (i.e. GCN,GAT,GraphSAGE)), our model is far superior in fraud detection tasks. This result is reasonable because these models have no special design for such tasks. The fact that PC-GNN performs much better than classic GNNs proved the necessity of sampling neighbors from the side. Similarly, the improvement of GraphConsis shows that noise in the graph is also a negative factor for fraud detection. CARE-GNN also proposed an RL neighborhood sampling for camouflage issue, but value-based sampling sacrifices effectiveness for efficiency. Unlike the above methods, *AO-GNN* considers both challenges and solves them in a consistent framework, which ensures monotonic improvement on AUC mathematically offsets the negative influence of noise and label imbalance practically.

Third, *AO-GNN* is valid in extremely imbalanced datasets. Among the previous baselines, only CARE-GNN and PC-GNN can study and achieve relatively stable scores in *Books*. We found that other baselines failed to identify any fraudsters and tend to classify all nodes as benign with utmost probabilities (i.e. Gmean=0). However, *AO-GNN* holds a preferable performance in *Books* as similar as in the other two datasets but with a slightly higher standard deviation.

In general, *AO-GNN* improves AUC and GMean, which are metrics unbiased to label distributions[24], on 3 datasets compared to all baselines. Therefore, *AO-GNN* is competent to alleviate the node label imbalance problem.

4.3 Ablation Study (RQ2)

We conduct three kinds of ablation tests to evaluate how each module contributes to the final result. Two of them are obtained by switching the edge pruning policy to a random policy, while another one is generated by deleting one of the modules. Based on the data in Table 2, we have the following observations:

First, topology optimizing in *AO-GNN* does promote AUC performance compared to optimizing via only parameters. Aside of display metrics, we also make t-test on AUC metric between *AO-GNN*$_{woP}$ and *AO-GNN*, whose p-value $\ll 0.01$. Thus, the breakthrough of *AO-GNN* indicates the superiority of the comprehensive optimization via both parameter and topology.

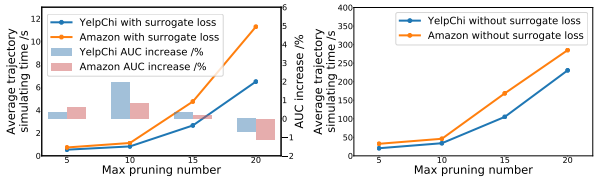
Second, by comparing *AO-GNN*$_{RP}$ and *AO-GNN*$_{woP}$, we can observe that random pruning is harmful to the AUC-oriented parameter searching, which further proves the rationality and necessity of our pruning policy.

Third, by comparing *AO-GNN*$_{woP}$ and *AO-GNN*$_{CE}$, we can find that model with only classifier parameter searching module has a better performance. We speculate it is the result of difference in RL reward quality. As our RL process needs an AUC-oriented GNN to return reward, GNN trained by cross entropy would influence the training effect of the edge pruning.

At last, we can notice that the F1-macro gets lower when the pruning policy is added. We speculate that the reason is the AUC-oriented training would push the predictive results away from the classification border in both directions, which may induce F1-macro compensation in late optimization process.

4.4 AUC Evolving Process Study (RQ3)

To answer RQ3, we display how AUC evolves in all datasets per 100 iterations in Fig. 2. The AUCs are recorded on the validation set and we randomly choose one of the records to present without loss



(a) (b)

Figure 4: Efficiency and effectiveness statistics. The bar charts in (a) is used to demonstrate the improvement of the halting mechanism, and the line charts in both (a)&(b) show the savings in time cost brought by the surrogate reward.

of generality. Recall that we set $T_k = 3^{k-1}T_0, T_0 = 600, K = 3$, thus the parameters of the classifier would jump to the center of past traces in the 600th, 2400th iterations. We can observe two apparent falls in the 6-th and the 24-th point as described in Alg. 2's Line 12-13. According to the Fig. 2, there are two decreases that happen when the increasing extent is gradually disappearing. After each falling, the AUC curves gain a longer-lasting growth and higher upper bound, which indicates that this trace jump mechanism avoids local optima.

4.5 Pruned Edges Study (RQ4)

To answer RQ4, we listed edge pruning-related statistics when applying our well-trained policy. We only present results on *YelpChi* and *Amazon* because multi-relation is easier to explain the pruning tendency. In the test sets of both datasets, we pruned 122,147 (12,527) edges out of 608,415 (54,618) edges we traversed in *YelpChi* (*Amazon*, respectively). Among those pruned edges, we classify them by relations as demonstrated in Fig.3.

From the histogram, we can discover that in *YelpChi* **R-U-R** edges are hardly pruned. Actually, only one **R-U-R** edge is pruned accidentally in practice. According to the connotation in Appendix B, the reviews connected by **R-U-R** are posted by the identical user and have the same label. This fact entails that our pruning policy is effective to learn a reasonable pruning tendency among relations even without prior knowledge. In *Amazon*, **U-P-U** edges account for 42.15% in the pruned edge with 3.77% frequency in total. This fact suggests that **U-P-U** edges are very noisy, because having a common review on the same product is effortless to realize.

4.6 RL Accelerating Mechanism Study (RQ5)

To answer RQ5, we compare the time consumption between whether using surrogate reward (Line charts in Fig. 4 (a) and (b)) and probe how the halting mechanism influences RL training efficiency (Bar charts in Fig. 4 (a)). We only test on two relatively large-scale graphs.

First, the halting mechanism is helpful in both efficiency and effectiveness. As demonstrated in Fig. 4 (a), 10 is the best max pruning number with the highest AUC increase (highest bars in both datasets) and relatively low time consumption (low position in lines of both datasets). Note that time consumption is not linear to max prune number. From the trajectory records, we find the reason is that the pruning policy seldom takes action after pruning enough edges, which leads to lengthy trajectories. And the negative impact when the max pruning number is 20 confirms that the issue of sparse reward is harmful to the effectiveness of the model.

Second, the surrogate reward reduces the time cost. Comparing Fig. 4 (a) and (b), when switching the surrogate loss to naive AUC variation, the average time for simulating one trajectory on *YelpChi* (*Amazon*) is 56 (30) times that of surrogate loss when max pruning number = 10. The facts above testify that our RL accelerating mechanism is effective and efficient.

5 RELATED WORK

Imbalanced Learning. The core idea of alleviating impacts of imbalanced label distribution is embellishing training sessions. In general, all imbalance learning methods could be divided into re-sampling approaches and re-weighting approaches. Re-sampling, i.e. oversampling and undersampling, is constructing a balanced training set essentially. Oversampling is usually accomplished by generating synthesis minorities. SMOTE [3] is the first work in this orientation based on interpolation. ADASYN [11] restudied this question from the learning difficulty of diverse minority cases. Besides, [41] generates samples with optimal transport theory. Compared to oversampling, undersampling is more intuitive since random oversampling is beneficial in some cases [44]. Trainable samplers is the mainstream in recent undersampling research, such as [23, 30]. Re-weighting can be summarized as meta-learning way [31, 51] and cost-sensitive way, both of which attempt to assign customized weight to different samples to leverage their influence throughout the whole training process. For cost-sensitive methods, loss function modification [14, 26] is the prevailing idea nowadays. Similarly, AUC maximization is also a cost-sensitive method that transforms AUC into a batch-friendly loss function.

Stochastic AUC Maximization. Traditional AUC maximization is unsuitable for the online setting because of its pairwise updating nature. [49] and [8] intend to solve this dilemma from the angle of reservoir buffer mechanism and optimal regret bound, respectively. [46] first converts AUC to a surrogate loss in a novel saddle point reformulation way. [19, 28] follow the framework of surrogate AUC loss and proposed optimization algorithms in better convergence rate and time complexity. However, all the above works need strong convex assumptions, which is hard to satisfy in DNNs. [18] solved this problem from proximal primal-dual optimization, which makes stochastic AUC maximization applicable in DNNs. To the best of our knowledge, our *AO-GNN* is the first work to solve the AUC maximization process on GNN, which takes topological structure into account in the whole optimization process.

GNN-based Fraud Detection. Following the groundwork in Eq (3), a lot of GNN architectures have been proposed. The most wide-applied GNNs are Graph Convolution Network (GCN) [12], GraphSAGE [10] and Graph Attention Network(GAT) [37]. Based on classic GNN architectures, various models are invented by reconstructing graph for fraud detection tasks, such as Fdgars [39], GraphConsis [22] and SemiGNN [38]. At present, GNN-based fraud detectors models are driven by challenges, which specific devise modules to comply with noise and label imbalance problems[5, 15, 17]. DR-GCN [34] is a pioneer work concerning label imbalance in graphs by implementing a dual-regularized GCN, which is composed of a class-conditioned adversarial regularizer and a latent distribution alignment regularizer, while it is not adaptable to graphs with large

scale. CARE-GNN [7] regards noise in graphs as camouflage, and enhances the aggregating process against it. PC-GNN [20] proposes to undersample a label balanced neighborhood for label balanced training set and achieves state-of-the-art in GNN-based fraud detection. In this paper, we attempt to render both challenges under a consistent AUC-oriented framework.

6 CONCLUSION

In this paper, we proposed *AO-GNN*, an AUC-oriented GNN-based model for graph fraud detection tasks. Unlike previous work, we attempted to alleviate label imbalance problem from the standpoint of maximizing AUC metric, which is unbiased to label distribution. Considering the noisy nature of fraud detection graph, we decoupled the AUC maximization process on graph into a classifier parameter searching and an edge pruning policy searching, respectively. The classifier parameter searching module is conducted by a surrogate loss of AUC. Meanwhile, we regard edge pruning policy searching as an MDP and optimize it with Policy Gradient. Experiments on three public fraud detection datasets demonstrate the effectiveness of *AO-GNN* on AUC, F1-macro, and GMean.

ACKNOWLEDGMENTS

The research work is supported by the National Natural Science Foundation of China under Grant (No.61976204, 92046003, U1811461). Xiang Ao is also supported by the Project of Youth Innovation Promotion Association CAS, Beijing Nova Program Z201100006820062. We would like to thank the anonymous reviewers for their valuable comments, and Guoxin Yu, Linfeng Dong, Zidi Qin for their insightful discussions.

REFERENCES

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *NeurIPS*. 5055–5065.
[2] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.
[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.
[4] Hao Chen, Yue Xu, Feiran Huang, Zengde Deng, Wenbing Huang, Senzhang Wang, Peng He, and Zhoujun Li. 2020. Label-aware graph convolutional networks. In *CIKM*. 1977–1980.
[5] Jianfeng Chi, Guanxiong Zeng, Qiwei Zhong, Ting Liang, Jinghua Feng, Xiang Ao, and Jiayu Tang. 2020. Learning to Undersampling for Class Imbalanced Credit Risk Forecasting. In *ICDM*. 72–81.
[6] Guillem Collell, Drazen Prelec, and Kaustubh R Patil. 2018. A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. *Neurocomputing* 275 (2018), 330–340.
[7] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM*. 315–324.
[8] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. 2013. One-pass AUC optimization. In *ICML*. 906–914.
[9] Wei Gao and Zhi-Hua Zhou. 2015. On the consistency of AUC pairwise optimization. In *IJCAI*. 939–945.
[10] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1025–1035.
[11] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IJCNN*. 1322–1328.
[12] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[13] Bartosz Krawczyk. 2016. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence* 5, 4 (2016), 221–232.
[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ICCV*. 2980–2988.
[15] Wangli Lin, Li Sun, Qiwei Zhong, Can Liu, Jinghua Feng, Xiang Ao, and Hao Yang. 2021. Online Credit Payment Fraud Detection via Structure-Aware Hierarchical Recurrent Neural Network. In *IJCAI*. 3670–3676.
[16] Can Liu, Li Sun, Xiang Ao, Jinghua Feng, Qing He, and Hao Yang. 2021. Intention-aware heterogeneous graph attention networks for fraud transactions detection. In *SIGKDD*. 3280–3288.
[17] Can Liu, Qiwei Zhong, Xiang Ao, Li Sun, Wangli Lin, Jinghua Feng, Qing He, and Jiayu Tang. 2020. Fraud transactions detection via behavior tree with local intention calibration. In *SIGKDD*. 3035–3043.
[18] Mingrui Liu, Zhuoning Yuan, Yiming Ying, and Tianbao Yang. 2019. Stochastic AUC Maximization with Deep Neural Networks. In *ICLR*.
[19] Mingrui Liu, Xiaoxuan Zhang, Zaiyi Chen, Xiaoyu Wang, and Tianbao Yang. 2018. Fast Stochastic AUC Maximization with $O(1/n)$-Convergence Rate. In *ICML*. 3189–3197.
[20] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection. In *WWW*. 3168–3177.
[21] Yang Liu, Xiang Ao, Qiwei Zhong, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Alike and Unlike: Resolving Class Imbalance Problem in Financial Credit Risk Assessment. In *CIKM*. 2125–2128.
[22] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *SIGIR*. 1569–1572.
[23] Zhining Liu, Pengfei Wei, Jing Jiang, Wei Cao, Jiang Bian, and Yi Chang. 2020. MESA: boost ensemble imbalanced learning with meta-sampler. In *NeurIPS*. 14463–14474.
[24] Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de las Heras. 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition* 91 (2019), 216–231.
[25] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*. 897–908.
[26] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*. 565–571.
[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. In *NeurIPS Workshop*.
[28] Michael Natole, Yiming Ying, and Siwei Lyu. 2018. Stochastic proximal algorithms for auc maximization. In *ICML*. 3710–3719.
[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: an imperative style, high-performance deep learning library. In *NeurIPS*. 8026–8037.
[30] Minlong Peng, Qi Zhang, Xiaoyu Xing, Tao Gui, Xuanjing Huang, Yu-Gang Jiang, Keyu Ding, and Zhigang Chen. 2019. Trainable undersampling for class-imbalance learning. In *AAAI*. 4707–4714.
[31] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to Reweight Examples for Robust Deep Learning. In *ICML*. 4334–4343.
[32] Patricia Iglesias Sánchez, Emmanuel Müller, Fabian Laforet, Fabian Keller, and Klemens Böhm. 2013. Statistical selection of congruent subspaces for mining attributed graphs. In *ICDM*. 647–656.
[33] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.
[34] Min Shi, Yufei Tang, Xingquan Zhu, David Wilson, and Jianxun Liu. 2021. Multi-class imbalanced graph convolutional network learning. In *IJCAI*. 2879–2885.
[35] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*.
[36] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*. 1057–1063.
[37] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attebtion Networks. In *ICLR*.
[38] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A Semi-supervised Graph Attentive Network for Financial Fraud Detection. In *ICDM*. 598–607.
[39] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. 2019. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *WWW*. 310–316.
[40] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).
[41] Yuguang Yan, Mingkui Tan, Yanwu Xu, Jiezhang Cao, Michael Ng, Huaqing Min, and Qingyao Wu. 2019. Oversampling for imbalanced data via optimal transport. In *AAAI*. 5605–5612.

[42] Zhiyong Yang, Qianqian Xu, Shilong Bao, Xiaochun Cao, and Qingming Huang. 2021. Learning with Multiclass AUC: Theory and Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
[43] Zhiyong Yang, Qianqian Xu, Shilong Bao, Yuan He, Xiaochun Cao, and Qingming Huang. 2021. When All We Need is a Piece of the Pie: A Generic Framework for Optimizing Two-way Partial AUC. In *ICML*. 11820–11829.
[44] Jian Yin, Chunjing Gan, Kaiqi Zhao, Xuan Lin, Zhe Quan, and Zhi-Jie Wang. 2020. A Novel Model for Imbalanced Data Classification. In *AAAI*. 6680–6687.
[45] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. In *NeurIPS*. 9240.
[46] Yiming Ying, Longyin Wen, and Siwei Lyu. 2016. Stochastic online AUC maximization. In *NeurIPS*. 451–459.
[47] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *ICLR*.
[48] Ge Zhang, Jia Wu, Jian Yang, Amin Beheshti, Shan Xue, Chuan Zhou, and Quan Z Sheng. 2021. FRAUDRE: Fraud Detection Dual-Resistant to Graph Inconsistency and Imbalance. In *ICDM*. 867–876.
[49] Peilin Zhao, Steven CH Hoi, Rong Jin, and Tianbao Yang. 2011. Online AUC maximization. In *ICML*. 233–240.
[50] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial Defaulter Detection on Online Credit Payment via Multi-view Attributed Heterogeneous Information Network. In *WWW*. 785–795.
[51] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. 2020. BBN: Bilateral-Branch Network with Cumulative Learning for Long-Tailed Visual Recognition. In *CVPR*. 9719–9728.
[52] Xiaoqian Zhu, Xiang Ao, Zidi Qin, Yanpeng Chang, Yang Liu, Qing He, and Jianping Li. 2021. Intelligent financial fraud detection practices in post-pandemic era. *The Innovation* 2, 4 (2021), 100176.

Appendices

In the appendix, we present proofs for 3 theorems mentioned in main text , and details about datasets and reproducibility.

A PROOFS

A.1 Proof for Theorem 1

Theorem 1. *The optimizing problem in Formula (8) is equivalent to*

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^d,\,\{a,b\}\in\mathbb{R}^2}\ \max_{\alpha\in\mathbb{R}}\ \mathbf{E}_{\mathbf{v}}[\mathcal{L}_{AUC}(\boldsymbol{\omega},a,b,\alpha,\mathbf{v}|\boldsymbol{\Pi}|p)],$$

where p is the ratio of fraud nodes, and

$$\begin{aligned}\mathcal{L}_{AUC}(\boldsymbol{\omega},a,b,\alpha,\mathbf{v}|\boldsymbol{\Pi},p) = {}& \mathbb{I}(y=1)[(1-p)(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-a)^2\\ &+2(p-1)(1+\alpha)\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})]\\ &+\mathbb{I}(y=0)[p(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-b)^2\\ &+2p(1+\alpha)\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})]+p(1-p)\alpha^2.\end{aligned}$$

Proof. The $\mathbb{E}_{\mathbf{v},\mathbf{v}'}$ part to be minimized in Formula (8) can be transformed as follows:

$$\begin{aligned}&\mathbb{E}_{\mathbf{v},\mathbf{v}'}[(1-(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi}))^2|y_{\mathbf{v}}=1,y_{\mathbf{v}'}=0]\\ =&\mathbb{E}_{\mathbf{v},\mathbf{v}'}\big[1+[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})]^2-2[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-\\ &-\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})]|y_{\mathbf{v}}=1,y_{\mathbf{v}'}=0\big]\\ =&1+\mathbb{E}_{\mathbf{v}}[{\mathcal{M}_{\mathcal{G}}}^2(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]+\mathbb{E}_{\mathbf{v}'}[{\mathcal{M}_{\mathcal{G}}}^2(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\\ &-2\big[\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]-\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big]\\ &-2\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\\ =&1+\underbrace{\{\mathbb{E}_{\mathbf{v}}[{\mathcal{M}_{\mathcal{G}}}^2(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]-(\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1])^2\}}_{A}\\ &+\underbrace{\{\mathbb{E}_{\mathbf{v}}[{\mathcal{M}_{\mathcal{G}}}^2(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]-(\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0])^2\}}_{B}\\ &-2\big[\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]-\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big]\\ &+\big(\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]-\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big)^2\end{aligned}$$

Note that for a random variable X, $\mathbb{E}(X^2)-[\mathbb{E}(X)]^2 = Var(x) = \mathbb{E}[(X-\mathbb{E}(X))^2]$, and $\min_k \mathbb{E}[(X-k)^2] = Var(x)$ i.i.f $k=\mathbb{E}(X)$, so:

$$\begin{aligned}&\mathbb{E}_{\mathbf{v},\mathbf{v}'}[(1-(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi}))^2|y_{\mathbf{v}}=1,y_{\mathbf{v}'}=0]\\ =&1+\min_{(a,b)\in\mathbb{R}^2}\big(\underbrace{\mathbb{E}_{\mathbf{v}}[(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-a)^2|y_{\mathbf{v}}=1]}_{A}+\underbrace{\mathbb{E}_{\mathbf{v}'}[(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'}\\ &\underbrace{-b)^2|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big)}_{B}\\ &-2\big[\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]-\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big]\\ &+\underbrace{\big(\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]-\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big)^2}_{C}\end{aligned}$$

Note that $\max_{\alpha\in\mathbb{R}}(2\alpha A-\alpha^2)=A^2$ i.i.f $\alpha=A$. If we denote $A=\big[\mathbb{E}_{\mathbf{v}}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})|y_{\mathbf{v}}=1]-\mathbb{E}_{\mathbf{v}'}[\mathcal{M}_{\mathcal{G}}\ (\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big]$, then:

$$\begin{aligned}&\mathbb{E}_{\mathbf{v},\mathbf{v}'}[(1-(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'|\boldsymbol{\Pi}))^2|y_{\mathbf{v}}=1,y_{\mathbf{v}'}=0]\\ =&1+\min_{(a,b)\in\mathbb{R}^2}\big(\mathbb{E}_{\mathbf{v}}[(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-a)^2|y_{\mathbf{v}}=1]+\mathbb{E}_{\mathbf{v}'}[(\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}'\\ &-b)^2|\boldsymbol{\Pi})|y_{\mathbf{v}'}=0]\big)+\underbrace{\max_{\alpha\in\mathbb{R}}(2\alpha A-\alpha^2)}_{C}-2A\\ =&1+\min_{(a,b)\in\mathbb{R}^2}\max_{\alpha\in\mathbb{R}}\mathbb{E}_{\mathbf{v}}\{\frac{\mathbb{I}(y_{\mathbf{v}}=1)}{p}[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-a]^2+\frac{\mathbb{I}(y_{\mathbf{v}}=0)}{1-p}\\ &[\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-b]^2+2(1+\alpha)\big(\frac{\mathbb{I}(y_{\mathbf{v}}=0)}{1-p}\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})-\\ &\frac{\mathbb{I}(y_{\mathbf{v}}=1)}{p}\mathcal{M}_{\mathcal{G}}(\boldsymbol{\omega};\mathbf{v}|\boldsymbol{\Pi})\big)-\alpha^2\}\\ =&1+\frac{\min_{\{a,b\}\in\mathbb{R}^2}\max_{\alpha\in\mathbb{R}}\ \mathbf{E}_{\mathbf{v}}[\mathcal{L}_{AUC}(\boldsymbol{\omega},a,b,\alpha,\mathbf{v}|\boldsymbol{\Pi},p)]}{p(1-p)}\end{aligned}$$

□

By importing aid parameters a,b,α, we eliminate all square related expectation terms, which simplifies following optimization. Be aware that the optimal α, i.e. $\alpha^*=A$, which is consistent to Line 13 in Alg. 2 because RHS in Line 13 is the unbiased estimation of A in the current batch.

A.2 Proof for Theorem 2

Theorem 2. *Under the MDP we constructed above, Eq (10) holds when discount factor $\gamma=1$.*

$$\begin{aligned}&\arg\max_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_r}\sum_{s\in\mathcal{S}}d^{\pi}(s)\mathbf{V}^{\pi}(s=(\mathbf{v}_c,\mathbf{v}_p,\Phi_{\mathbf{v}_c}))\\=&\arg\max_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_r}AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\boldsymbol{\Pi}_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_r})\end{aligned}$$

Proof. We denote the graph $\mathcal{G}$ in timestep t as $\mathcal{G}_t$, then the reward in timestep $t+1$ is $AUC(\mathcal{M}^{\omega}_{\mathcal{G}_{t+1}}) - AUC(\mathcal{M}^{\omega}_{\mathcal{G}_t})$ Then, if $\gamma = 1$, we have:

$$\begin{aligned}G_0&=\sum_{k=0}^{\infty}\gamma^k r_{k+1}=\sum_{k=0}^{\infty}[AUC(\mathcal{M}^{\omega}_{\mathcal{G}_{k+1}})-AUC(\mathcal{M}^{\omega}_{\mathcal{G}_k})]\\&=-AUC(\mathcal{M}^{\omega}_{\mathcal{G}_0})+\lim_{t\to\infty}AUC(\mathcal{M}^{\omega}_{\mathcal{G}_t})\end{aligned}$$

The G_0 is the return in timestep $t=0$ in MDP introduced in Section 2.3.1. Therefore, $\forall s, \mathbf{V}^{\pi}(s) = -AUC(\mathcal{M}^{\omega}_{\mathcal{G}_0}) + \lim_{t\to\infty} AUC(\mathcal{M}^{\omega}_{\mathcal{G}_t})$, then :

$$\begin{aligned}&\max_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_r}\sum_{s\in\mathcal{S}}d^{\pi}(s)\mathbf{V}^{\pi}(s=(\mathbf{v}_c,\mathbf{v}_p,\Phi_{\mathbf{v}_c}))\\=&-AUC(\mathcal{M}^{\omega}_{\mathcal{G}_0})+\max_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_r}\lim_{t\to\infty}AUC(\mathcal{M}^{\omega}_{\mathcal{G}_t})\\=&-AUC(\mathcal{M}^{\omega}_{\mathcal{G}_0})+\max_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_r}AUC(\mathcal{M}^{\omega}_{\mathcal{G}_t}|\boldsymbol{\Pi}_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_r})\end{aligned}$$

Since $\mathcal{G}_0$ and $\mathcal{M}^{\omega}$ are determined, $AUC(\mathcal{M}^{\omega}_{\mathcal{G}_0})$ is a constant. Thus the original equation Eq (10) holds. □

A.3 Proof for Theorem 3

Theorem 3. *Eq(16) is equivalent to AUC variation,* $AUC(\mathcal{M}^{\omega}_{\mathcal{G}'}|\boldsymbol{\Pi}) - AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\boldsymbol{\Pi})$*, and computing it only takes* $O(n)$ *time.*

Proof. If we prune one neighbor of $\mathbf{v}_c$ and the original graph $\mathcal{G}$ transform into $\mathcal{G}'$, only $P_{\mathbf{v}_c}$ would change to $P'_{\mathbf{v}_c}$. Say there are n^+ positive nodes ($\mathbf{v}^+$) and n^- negative nodes ($\mathbf{v}^-$), denoted as node sets $\mathcal{V}^+$ and $\mathcal{V}^-$ respectively. And node set $\mathcal{V}^*$ denotes the nodes ($\mathbf{v}^*$) whose label is not $y_{\mathbf{v}_c}$. According to Eq (2), we have:

$$\begin{aligned}&AUC(\mathcal{M}^{\omega}_{\mathcal{G}'}|\boldsymbol{\Pi})-AUC(\mathcal{M}^{\omega}_{\mathcal{G}}|\boldsymbol{\Pi})\\=&\frac{1}{n^+n^-}[\sum_{\mathcal{V}^+}\sum_{\mathcal{V}^-}\mathbb{I}(P'_{\mathbf{v}^+}\geq P'_{\mathbf{v}^-})-\sum_{\mathcal{V}^+}\sum_{\mathcal{V}^-}\mathbb{I}(P_{\mathbf{v}^+}\geq P_{\mathbf{v}^-})]\\\propto&F(y_{\mathbf{v}_c})F(P'_{\mathbf{v}_c}\geq P_{\mathbf{v}_c})\sum_{\mathcal{V}^*}\mathbb{I}[(P_{\mathbf{v}_c}<\mathbf{v}^*\leq P'_{\mathbf{v}_c})\vee(P'_{\mathbf{v}_c}<\mathbf{v}^*\leq P_{\mathbf{v}_c})]\\=&\textit{RHS in Eq (14)}\end{aligned}$$

After traversing the whole node set once, we can count how many nodes is eligible in Eq (14), which consumes $O(n)$ time. □

Theorem 3 transforms the whole AUC computing process into a local comparison, which boosts the efficiency.

B DATASETS INTRODUCTION

The *Yelpchi* collects reviews and ratings of all accommodation serving merchants in Chicago from Yelp.com, in which each review will be regarded as a node in the graph with a 100-dimension feature vector extracted from the metadata and review context, and labeled as fraud if spamming behaviors are detected. Meanwhile, nodes are connected in 3 relations with particular meanings: (1) **R-U-R**: posted by the same user; (2) **R-T-R**: posted at the approximately same time; (3) **R-S-R**: posted for the same merchant with identical rating scores. The *Amazon* dataset regards users from Amazon.com as nodes with 100-dimension feature, which are labeled as benign with more than 80% up-votes. There are 3 diverse relations in *Amazon*: (1) **U-P-U**: at least one common reviewing product; (2) **U-S-U**: at least one identical rating within a week; (3) **U-V-U**: top-5% mutual review TF-IDF similarities among all user pairs. The *Books* is an extremely imbalanced dataset with single relation, in which nodes represent books and edges imply co-purchasing. A node is marked as fraud if over 20 users labeled it for unmatch sales ranking.

C REPRODUCIBILITY

For our baselines, GCN, GraphSAGE and GAT are implemented based on DGL [40]. GraphConsis, CARE-GNN and PC-GNN are implemented using public source code provided by the authors. DR-GCN is implemented by ourselves as no source code is available.

The whole code for *AO-GNN* is implemented in PyTorch 1.6.0 [29]. The python version we choose is 3.8, and the launch Ubuntu 18.04 LTS server with 56 Intel E5-2680 cores and 512G memory. And for CUDA running, we employ 1x NVidia Tesla V100 (32GB) to train our model. Because we need to record gradient history in Alg. 2, it is recommended to use devices with high memory.

C.1 Classifier Parameter Searching Module

First, we present structure-related hyper-parameters in the classifier parameter searching module. For each parameter matrix W in each layer of GNN, it remains the same feature dimension so that the layer object is reusable. For the final MLP to produce predictive results, we devise a hidden layer for it with 20% dimension of the input vector, and the output activation function is Softmax. Besides, we use conventional tricks to get better results: 1) we add an affine layer before GNN to map the raw feature vectors into a better representation space; 2) we use PReLu as GNN layer activation function, which yields better results compared to ReLu; 3) we apply dropout in each MLP layer with a dropout of 50%. In data preprocessing, we use a standard scaler provided by scikit-learn to normalize all feature data by the mean and variation observed from the training set only. The optimizer for Alg. 2 is not aligned to any predefined optimizers and we implement it under Pytorch framework.

C.2 Edge Pruning Policy Searching Module

The policy network is an MLP with a hidden layer, of which the dimension is as 60% of input vectors. Training the whole RL model is quite skillful. Half of the trajectories are generated in an off-policy way to avoid the policy being too deterministic. Moreover, we conduct gradient ascend for every 100 trajectories. For tiny-scale datasets or datasets easy to be classified, training multiple policy networks may face a lack of training material. For instance, in Amazon whose AUC result is very high, the predictive probabilities hardly move after pruning. Under this circumstance, we can oversample the trajectories generated from poorly classified nodes.