Multi-view Heterogeneous Temporal Graph Neural Network for "Click Farming" Detection

Zequan Xu[1], Qihang Sun[2], Shaofeng Hu[2], Jiguang Qiu[3], Chen Lin[1], and Hui Li[1](✉)

[1] School of Informatics, Xiamen University, Xiamen, China
xuzequan@stu.xmu.edu.cn, {chenlin,hui}@xmu.edu.cn
[2] Tencent, Guangzhou, China
{aaronqhsun,hugohu}@tencent.com
[3] Xiamen Meiya Pico Information Co., Ltd., Xiamen, China
qiujg@300188.cn

Abstract. Multi-purpose Messaging Mobile App (MMMA) combines several functionalities in a single APP to provide integrated service that brings tremendous convenience to users. Therefore, MMMAs become more and more popular. However, the prevalence of MMMAs also makes them a hotbed for cybercrime. Among them, "Click Farming" fraud requires special attention, as it causes substantial pecuniary losses and is challenging to detect. In this paper, we describe Multi-view Heterogeneous Temporal Graph Neural Network (MHT-GNN), a framework for detecting "Click Farming" fraudsters in a popular MMMA called WeChat. We first adopt a Heterogeneous Temporal Graph (HTG) to model spatial, heterogeneous and temporal information contained in MM-MA data. We then extract two different types of user history sequences as two "views" of user behavior patterns from HTG. MHT-GNN contains a pretraining phase and a detection phase. The main components in MHT-GNN include Inductive Heterogeneous GNN Encoder, Temporal Snapshot Sequence Encoder, and User Relation Sequence Encoder. The first encoder aims to capture spatial information and the heterogeneity in each snapshot of HTG. The later two encoders are designed to incorporate temporal information to better reveal user's behavior patterns and MHT-GNN leverages them to capture the two different views of user history behavior data. We conduct experiments on a real-world, million-scale dynamic graph extracted from WeChat. Experimental results demonstrate the effectiveness of MHT-GNN: it significantly exceeds existing detection methods, and it is able to block "Click Farming" fraud activities.

Keywords: Click farming detection · Multi-purpose messaging mobile app · Graph based anomaly detection · Graph neural network

1 Introduction

Smart phones have become an essential tool in our everyday life and they provide different functionalities (e.g., socializing, online games, online shopping and mobile payment) via installed apps. Nowadays, one single app is no longer limited to one application scenario. Several functionalities can be assembled in one app to provide integrated service that helps users handle their daily demands. Such apps are often referred to as Multi-purpose Messaging Mobile Apps (MMMAs). WeChat, with over a billion users, is a representative app in this category. Users can easily chat with their friends using texts, voice messages or voice/video calls provided by WeChat. Moreover, the digital payment service of WeChat has revolutionized people's daily life: we can simply use QR code to replace wallet and transfer money, which is more convenient and safer.

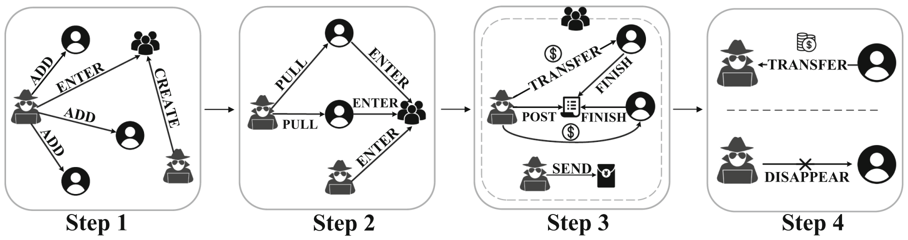

Fig. 1. An illustration of "Click Farming" in WeChat.

The great convenience brought by MMMAs attracts more and more users. On the other hand, the prevalence of MMMAs like WeChat makes them a hotbed for cybercrime [19]. This paper studies the detection task of "Click Farming" in WeChat: a type of deception that recently emerges. As depicted in Fig. 1, in "Click Farming" frauds, fraudsters first use illegally acquired personal information (e.g., phone number) and send "add friend" requests to victims (i.e., `ADD` in Fig. 1 - Step 1). Additionally, certain chat groups are created where cybercriminals enter and disguise as normal users (i.e., `ENTER` and `CREATE` in Fig. 1 - Step 1). Then, fraudsters will invite victims (i.e., `PULL` in Fig. 1 - Step 2) to join these groups (i.e., `ENTER` in Fig. 1 - Step 2) by using high reward as bait. After that, group members are encouraged to complete some tasks (i.e., `FINISH` in Fig. 1 - Step 3) posted in the group (i.e., `POST` in Fig. 1 - Step 3). Typical tasks include buying a number of products or topping up online shopping cards. The first a few tasks are easy. Victims do not need to pay too much (i.e., `TRANSFER` in Fig. 1 - Step 3) and fraudsters pay the commission as promised to gain the trust of victims. With victims' guard down, fraudsters raise the request of new tasks and ask victims to pay much more money. Victims may see that other group members (they are conspirators) complete new tasks and get reward. Hence, they decide to pay the money to complete new tasks. However, after victims transfer money, fraudsters disappear and do not response anymore (i.e., `DISAPPEAR` in Fig. 1- Step 4).

Due to the pecuniary losses that "Click Farming" frauds cause, the "Click Farming" fraudsters detection task (the CFD task) requests our attention. The social nature of WeChat makes it a natural choice to model WeChat user relationships as a user-user interaction graph. This way, the CFD task is closely related to Graph-based Anomaly Detection (GBAD) [1]. However, the CFD task in WeChat has unique properties and therefore is more challenging. The data of WeChat is *dynamic* and *diverse*. A fraudster may appear to be normal in each individual snapshot. But he/she becomes suspicious when considering all his/her different behaviors at each snapshot together.

In the literature, only few works [17,21] study the GBAD task in the dynamic setting. But they cannot handle both dynamics and diversity of WeChat data well. Particularly, all previous works leverage *only one* view of dynamic data (e.g., viewing states of a node in different snapshots of the dynamic graph as a sequence [21]), which is not sufficient to model the dynamic and diverse user behaviors in the CFD task. We propose a framework Multi-View Heterogeneous Temporal Graph Neural Network (MHT-GNN) for the CFD task in WeChat. We extract two types of user history sequences from our designed Heterogeneous Temporal Graph (HTG) as two "views". Then, MHT-GNN captures both temporal dependencies (dynamics) and behavior patterns (diversity) from user history sequences and the HTG through multi-view learning and graph representation learning. To our best knowledge, we are the first to study the "Click Farming" detection problem in MMMAs. The contributions of this work can be summarized as follows:

– We analyze and design features used in the CFD task of WeChat. We further propose a Heterogeneous Temporal Graph to model diverse MMMA data.
– We adopt an Inductive Heterogeneous graph encoder to capture spatial dependencies and heterogeneity in WeChat. It provides better representation learning for the WeChat graph compared to other GNN-based methods and it can generalize to unseen nodes.
– We construct two types of user history sequences for each node as two "views" of the dynamic data. We further design two encoders to encode two views to capture the temporal dependencies and behavior patterns, which helps generate better node representations in the CFD task.
– We conduct evaluations on a million-scale real-world graph extracted from the CFD task in WeChat. Results show that MHT-GNN exceeds existing methods by a large margin.

2 Related Work

2.1 Graph-Based Anomaly Detection (GBAD)

Anomaly detection identifies the abnormal patterns that deviate from the majorities [8]. Graph-based Anomaly Detection (GBAD) extends it to the graph data. Earlier methods for GBAD are mainly based on handcrafted feature engineering [6]. Recent works are mostly inspired by the deep learning techniques. DOMINANT [4] leverages the graph embeddings from GCN to reconstruct the original adjacent matrix for anomaly detection. ALARM [12] further employs multiple attributed views to describe different perspectives of the objects for anomalies detection. Different from previous methods that jointly learn the node representation and the classifier, DCI [18], inspired by the recent advances of self-supervised learning, decouples these two phases for node representation learning.

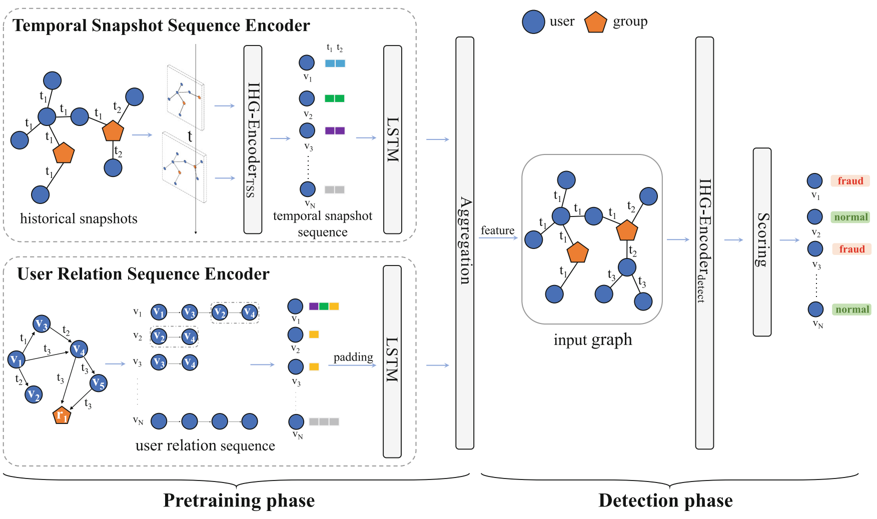

Fig. 2. Overview of MHT-GNN.

2.2 Anomaly Detection in Dynamic Graphs

Anomaly detection in dynamic graphs attracts increasing interest since many real-world networks can be generally represented in the form of dynamic graphs. Earlier methods such as CAD [15] detect node relationships responsible for abnormal changes in graph structure by tacking a measure that combines information regarding changes in both graph structure and edge weights. StreamSpot [10] is a clustering based approach that introduces a new similarity function for heterogeneous graph comparison. Another branch of approaches employs deep learning. [21] first utilizes temporal GCN and attention mechanism to model short-term and long-term patterns. Then a GRU network is introduced to process such patterns and encode temporal features. NetWalk [20] adopts a random walk based encoder to learn the network representations and employs a clustering-based anomaly detector to score the abnormality of each edge. StrGNN [2] extracts the h-hop enclosing subgraph of edges and labels each node to identify its corresponding role in the subgraph. Then it leverages GCN and GRU to capture the spatial and temporal information for anomaly detection.

3 Our Framework MHT-GNN

3.1 Overview

Figure 2 provides an overview of MHT-GNN. It consists of two phases: pretraining and detection. We first extract node features (Sect. 3.2) and construct a Heterogeneous Temporal Graph (Sect. 3.3) for WeChat data. Then, in the pretraining phase (Sect. 3.4), we construct two types of history sequence for each node in the constructed graph, namely temporal snapshot sequence and user relation sequence. These two types of sequences can be regarded as two "views" of user history sequence for each user. MHT-GNN generates embeddings for each node in each snapshot using an Inductive Heterogeneous GNN Encoder (IHG-Encoder).

Based on the graph representations generated by IHG-Encoder, temporal snapshot sequences and user relation sequences are passed through our designed Temporal Snapshot Sequence Encoder (TSS-Encoder) and User Relation Sequence Encoder (URS-Encoder) to generate more informative representations, respectively.

In the detection phase (Sect. 3.5), MHT-GNN uses pretrained encoders to generate sequence representations for predicting the suspicious score of a user.

3.2 Feature Extraction

For each user, we pre-extract six features from WeChat data based on the knowledge of human experts. Note that the detection of "Click Farming" fraudsters should not violate users' privacy. Hence, private information like chat content (text, video or speech) in WeChat is unaccessible. The data used for the CFD task is chosen through a strict investigation process in order to protect users' privacy.

3.3 Graph Construction

To capture the diverse behavior patterns in the WeChat graph, we construct a heterogeneous graph [14] capable of modeling heterogeneous spatial dependencies among different types of node entities and relations. Among different nodes and behaviors in WeChat, we consider two key node types (i.e., users and chat groups) and three important relation types: "join a group" (`ENTER`), "invite someone to join a group" (`PULL`) and "become WeChat friends" (`ADD`). The heterogeneous graph not only depicts the graph structure of WeChat graph, but also provides a higher-level abstraction of the user association. For example, a pattern of $\textit{fraudulent user} \xrightarrow{\texttt{PULL}} \textit{normal user} \xrightarrow{\texttt{ENTER}} \textit{group} \xleftarrow{\texttt{ENTER}} \textit{fraudulent user}$ in the heterogeneous graph can characterize a "Click Farming" fraud case: a fraudster invites a victim to join a chat group and another fraudster is also a member of this group.

Based on the above designed heterogeneous graph, we propose to further consider temporal dependencies (i.g., evolving user states and behaviors) and build a *Heterogeneous Temporal Graph* (HTG) for the CFD task:

Definition 1. *Heterogeneous Temporal Graph (HTG). We model a HTG as a graph stream consists of discrete snapshots. Let the latest timestamp be T. A graph stream can be denoted as $\mathbb{G} = \{\mathcal{G}^t\}_{t=1}^T$, where each $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ is a heterogeneous graph at timestamp t. We use $n^t = |\mathcal{V}^t|$ and $m^t = |\mathcal{E}^t|$ to denote the number of nodes and edges at timestamp t, respectively.*

HTG is the combination of several basic heterogeneous graphs from different time points. And each basic heterogeneous graph of the HTG is a snapshot of the HTG at the corresponding time point.

3.4 Pretraining Phase

Inductive Heterogeneous GNN Encoder (IHG-Encoder). We adopt an IHG-Encoder as the backbone of MHT-GNN for encoding graph data.

In the following, we only consider one snapshot of the WeChat graph to illustrate IHG-Encoder. We first project the raw user features $\mathbf{p}_v \in \mathbb{R}^6$ of a user u to a feature space and utilize projected features as the initial user node embedding for u. For the initial embeddings of a group node g, we aggregate all its members' initial embeddings:

$$\mathbf{h}_v^{(0)} = \mathbf{W}_\mathbf{h}\mathbf{p}_v, \quad \mathbf{h}_g^{(0)} = \mathrm{mean}(\{\mathbf{h}_{v'}, \forall v' \in N_g\}) \tag{1}$$

where $\mathbf{h}_v^{(0)}$ and $\mathbf{h}_g^{(0)}$ are initial embeddings for the user node v and the group node g, respectively. N_g denotes users in g, and $\mathbf{W}_\mathbf{h}$ is a learnable matrix.

The messaging passing mechanism in IHG-Encoder is *relation-wise*. Representations of neighboring nodes connected to a user u by the same relation r are aggregated by three different pooling methods. Results are concatenated and passed to a single-layer feedforward neural network:

$$\begin{aligned}
\mathbf{x}_{N_i^r}^{(k+1)} &= \mathrm{mean}(\mathbf{h}_{r,j_1}^{(k)}, \ldots, \mathbf{h}_{r,j_*}^{(k)}) \qquad & \mathbf{h}_{N_i^r}^{(k+1)} &= \mathbf{x}_{N_i^r}^{(k+1)} \oplus \mathbf{y}_{N_i^r}^{(k+1)} \oplus \mathbf{z}_{N_i^r}^{(k+1)} \\
\mathbf{y}_{N_i^r}^{(k+1)} &= \max(\mathbf{h}_{r,j_1}^{(k)}, \ldots, \mathbf{h}_{r,j_*}^{(k)}) & \mathbf{m}_{N_i^r}^{(k+1)} &= \mathbf{W}_r \mathbf{h}_{N_i^r}^{(k+1)} + \mathbf{b}_r \\
\mathbf{z}_{N_i^r}^{(k+1)} &= \mathrm{sum}(\mathbf{h}_{r,j_1}^{(k)}, \ldots, \mathbf{h}_{r,j_*}^{(k)}) & &
\end{aligned} \tag{2}$$

where the superscript (k) indicates the k-th iteration, $\oplus$ is the concatenation operation, N_i^r denotes the relation-r-based neighbors of node i and $j_* \in N_i^r$. $\mathbf{h}_{r,j_*}^{(k)}$ is the representation of node j_* for relation r, and $\mathbf{h}_{r,j_*}^{(0)}$ is equivalent to $\mathbf{h}_{j_*}^{(0)}$. $\mathrm{mean}(\cdot), \max(\cdot)$ and $\mathrm{sum}(\cdot)$ are average pooling, max pooling and sum pooling, respectively. W_r and b_r are learnable weights for relation type r.

IHG-Encoder adds a self-connection to each node so that the original node attributes extracted based on human knowledge can be retained in message passing:

$$\mathbf{s}_{r,i}^{(k+1)} = \mathbf{W}_{r,s}\mathbf{h}_{r,i}^{(0)} + \mathbf{b}_{r,s}, \qquad \mathbf{g}_{r,i}^{(k+1)} = \mathrm{RELU}(\mathbf{m}_{N_i^r}^{(k+1)} \oplus \mathbf{s}_{r,i}^{(k+1)}) \tag{3}$$

where $\mathbf{W}_{r,s}$ and $\mathbf{b}_{r,s}$ are learnable parameters and RELU$(\cdot)$ is the Rectified Linear Unit. The acquired $\mathbf{g}_{r,i}$ is then passed to a feedforward neural network with an L_2 normalization:

$$\mathbf{q}_{r,i}^{(k+1)} = \text{RELU}(\mathbf{W}_{r,q}\mathbf{g}_{r,i}^{(k+1)} + \mathbf{b}_{r,q}), \qquad \mathbf{h}_{r,i}^{(k+1)} = \mathbf{q}_{r,i}^{(k+1)} \Big/ \left\|\mathbf{q}_{r,i}^{(k+1)}\right\| \tag{4}$$

where $\mathbf{W}_{r,q}$ and $\mathbf{b}_{r,q}$ are learnable weights, and $\mathbf{h}_{r,i}$ indicates the final generated representation of i w.r.t. the relation r.

The output representations of node i for all relations will go through an inter-relation aggregation module and the result is the representation for node i:

$$\mathbf{h}_{i}^{(k+1)} = \text{AGG}(\{\mathbf{h}_{r,i}^{(k+1)}, \forall r \in R\}) \tag{5}$$

where AGG is the aggregation function and we adopt mean pooling. IHG-Encoder stacks two of the above GNN layers (i.e., Eqs. 2, 3, 4 and 5) to generate the final representation of node i. Note that some user nodes may only exist in certain view of the constructed heterogeneous graph, e.g., a user only has `ADD` actions in the considered time period. For other views the users are absent, their corresponding passing messages will be set to zero.

IHG-Encoder can be optimized with a standard binary cross entropy loss over labeled nodes. IHG-Encoder does not maintain node embeddings which are bounded by specific nodes. Instead, learnable weights $\mathbf{W}$ and $\mathbf{b}$ are updated during optimization. In detection, the trained model can be used to produce representations for *new* nodes based on their structural and raw attribute information. Hence, IHG-Encoder is indeed inductive (i.e., the trained model can be used over unseen nodes), which is essential for representation learning in WeChat as new users emerge every day. MHT-GNN, which uses IHG-Encoder as its backbone, is therefore also able to generate representations for unseen nodes.

Temporal Snapshot Sequence Encoder (TSS-Encoder). We observed that, in "Click Farming", a fraudster's fraud actions may spread across multiple timestamps. A fraudster Alice adds many potential victims as friends at time t_1. Alice spends a few days using high reward as bait to convince them to join a "Click Farming" group. Then, at time t_2, Alice will invite baited users to the group. Alice may continue to be active in the group performing actions like sending bonus packages for encouragement at time t_3. On the contrary, a normal user Bob typically does not have so many behaviors within a relatedly short time window. Hence, we concatenate the presentations of a user in different snapshots as its *temporal snapshot sequence* to capture temporal patterns in the HTG.

Given a series of historical snapshots $\{\mathcal{G}^t\}_{t=1}^{T}$ as inputs, we apply IHG-Encoder over each snapshot to obtain the representations for all the nodes in each snapshot. By doing so, we collect a sequence of representations for each user u at different time steps. Specifically, for each node u, we define its temporal snapshot sequence as $\text{seq}_u^{temp} = [\mathbf{h}_u^1, \mathbf{h}_u^2, \ldots, \mathbf{h}_u^T]$. Note that the WeChat graph can easily scale to millions or even tens of millions due to its massive users. We can utilize an efficient database (e.g., a key-value pair database) to store previous feature representations generated by IHG-Encoder for each user. When checking a user's anomalousness in current timestamp t, we can easily retrieve his/her historical representations from the database and construct the temporal snapshot sequence in a blink. Only the current representations requires the generation of the IHG-Encoder.

Next, we aggregate the retrieved temporal snapshot sequence to a representation that captures user behavior patterns. We adopt the Long Short-Term Memory (LSTM) as TSS-Encoder to model the input sequence seq_u^{temp} and capture the dynamic of user activities. LSTM fits perfectly in this scenario for the reason that it recognizes temporal dependencies. Each layer of the LSTM computes the following transformations:

$$\begin{aligned} \mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{e}_t] + \mathbf{b}_f), & \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}_t} \\ \mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{e}_t] + \mathbf{b}_i), & \mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{e}_t] + \mathbf{b}_o) \\ \tilde{\mathbf{c}_t} &= \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{e}_t] + \mathbf{b}_c), & \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \end{aligned} \tag{6}$$

where t is the time step, $\mathbf{h}_t$, $\mathbf{c}_t$, $\mathbf{e}_t$ are hidden state, cell state and previous layer hidden state at time t, respectively. $\mathbf{f}_t$, $\mathbf{i}_t$, $\mathbf{o}_t$ are respectively the forget gate, input gate and output gate, and $\odot$ indicates the Hadamard product.

The last hidden state output by TSS-Encoder is used as the representation $\mathbf{h}_{\text{seq}_u^{temp}}$ of temporal snapshot sequences for a user u.

User Relation Sequence Encoder (URS-Encoder). In social networks, a user's *direct* actions explicitly reveal his/her characteristics. In the HTG, a user's direct actions manifest in edges between itself and its 1-hop out-neighbors. Observed from the "Click Farming" fraud example we discussed in Fig. 1, we can conclude that this type of fraud typically involves several direct actions (e.g., `ADD` and `PULL`) of fraudsters appearing in different stages of "Click Farming" frauds (i.e., searching, gain trust and deceive). Hence, we believe it is beneficial to consider a special type of sequence called *user relation sequence*. Such sequences are composed of edges in all the 1-hop neighboring subgraphs of a user node from different snapshot and these edges are sorted in chronological order.

Given a node v and its 1-hop out-degree neighboring node set $N_v = [u_1, \ldots, u_m]$ associated with corresponding edge set $E_v = \{(v, u_1, t_1), \ldots, (v, u_m, t_m)\}$ where $|N_v| = m$ and a tuple (v, u_i, t_i) indicates that there is an edge from v to u_i at time step t_i, we sort N_v in a chronological manner and sample nodes from each time step to form a user relation sequence $\{u_1, u_2, \ldots, u_T\}_v$ for user v. For any two nodes u_i and u_j in the sequence with $i < j$, their associated edges (v, u_i, t_i) and (v, u_j, t_j) satisfy that $t_i < t_j$. An example is provided in the bottom left of Fig. 2.

When constructing user relation sequences, some issues require extra attention and we handle them as follows:

- The number of 1-hop neighbors is uneven across the graph, meaning that some users are relatively active in the recorded time period while some are not. Active users have much more user relation sequences than inactive users,

which may lead to model bias. Therefore, we sample up to a predefined maximum number of sequences for each user to avoid model bias as well as speed up model training.
- For users with few or no out-degree neighbors during the recorded time period, we take the sub-sequence from other users as sequences. For instance, node v_2 in the bottom left of Fig. 2 has no out-degree neighbors. But v_2 exists in the out-degree neighborhood of v_1. Hence, we extract the sub-sequence from v_1 that starts with v_2 as the user relation sequence for v_2.

User relation sequence describes a user's behavior over time, which remedies the limitation of temporal snapshot sequence that solely contains the hidden state of the same user over time. Given the user relation sequence of node u: $\text{seq}_u^{rel} = [v_1^{(u)}, v_2^{(u)}, \ldots, v_T^{(u)}]$, where $v_t^{(u)}$ $(1 \leq t \leq T)$ denotes the user/group node with which u interacts at t-th time step, we first generate initial embeddings for nodes in the sequence using the same projection shown in Eq. 1. Then, similar to TSS-Encoder, the embeddings sequence $[\mathbf{h}_{v_1^{(u)}}, \mathbf{h}_{v_2^{(u)}}, \ldots, \mathbf{h}_{v_T^{(u)}}]$ is fed into URS-Encoder composed of a LSTM to produce the representation $\mathbf{h}_{\text{seq}_u^{rel}}$ of the user relation sequences for the user u.

Optimization of Pretraining Phase. We train TSS-Encoder, URS-Encoder and IHG-Encoder$_{\text{detect}}$ independently on the training data with limited labels using binary cross-entropy loss and Adam optimizer. IHG-Encoder$_{\text{detect}}$ adopts the same encoder design as IHG-Encoder. As shown in Fig. 2, IHG-Encoder$_{\text{detect}}$ is later used in the detection phase. During training, each of TSS-Encoder, URS-Encoder and IHG-Encoder$_{\text{detect}}$ is connected to its own score module, which is a linear mapping layer followed by a sigmoid function, for predicting suspicious scores. If the predicted score of an input node is larger than 0.5, it is labeled as a "Click Farming" fraudster.

3.5 Detection Phase

During the detection phase, for a user node v, we generated temporal snapshot sequence representation $\mathbf{h}_{\text{seq}_v^{temp}}$ and user relation sequence representation $\mathbf{h}_{\text{seq}_v^{rel}}$ using pretrained TSS-Encoder and URS-Encoder. Then, $\mathbf{h}_{\text{seq}_v^{temp}}$, $\mathbf{h}_{\text{seq}_v^{rel}}$ and initial representation $\mathbf{h}_v^{(0)}$ are concatenated and the result is fed into IHG-Encoder$_{\text{detect}}$ followed by its scoring module to estimate the suspicious score of v. If the output value for v is larger than 0.5, v will be predicted as a "Click Farming" fraudster.

4 Experiments

4.1 Experiment Setting

Data. We extract a 14 day-period dataset from WeChat and construct a million-scale HTG as defined in Sect. 3.3. The graph contains nearly 4.6 million user nodes and 190 thousand WeChat chat group nodes. The number of edges are approximately 15 million covering three relations: `ADD`, `PULL` and `ENTER`. We use one day as the interval between two timestamps. Thus, for a 14 days observation period we derive 14 separate graph snapshots. We set the maximum number of sampled user relation sequence for each user to be 10. 85,000 user nodes are manually labeled by human experts: 25,000 are fraudsters and 60,000 are normal users. The labels for other 4.5 million user nodes are unknown. We randomly divide the labeled users by a ratio of 8:1:1 for training, validation and testing.

Baseline. We compare MHT-GNN with several competitive baselines:

- **Non-GNN classification methods.** XGBoost [3] and MLP. XGBoost is a gradient boosting algorithm that shows promising results in numerous prediction tasks and MLP is a feedfoward neural network with three hidden layers to predict the suspicious score of a node. The two methods only relies on data attributes for prediction.
- **Homogeneous graph based methods.** Graph Convolutional Network (GCN) [5] averages neighbor's embeddings with a linear projection, and Graph Attention Network (GAT) [16] utilizes attention mechanism to aggregate information of neighbors.
- **Heterogeneous graph based methods.** Relational Graph Convolutional Network (RGCN) [13] designs different linear projections for different types of relations for information aggregation, and Simple Heterogeneous Graph Neural Network (Simple-HGN) [7] enhances GAT with the redesign of three techniques: learnable edge-type embedding, residual connections, and L_2 normalization on the output embeddings.
- **Temporal graph anomaly detection method.** AddGraph [21] is an dynamic graph anomaly detection method. It leverages a GCN module to capture spatial information, and employs a GRU-attention module to extract short- and long- term dynamic evolving patterns. Furthermore, we modify the base graph encoder of AddGraph from GCN to RGCN in order to model the heterogeneous information and name this variant as AddGraph-H.

To verify the contribution of each component in MHT-GNN, we design several versions of MHT-GNN as follows:

- IHG-Encoder: It only contains the inductive heterogeneous GNN encoder.
- MHT-GNN-T: It is a variant of MHT-GNN that removes URS Encoder.
- MHT-GNN-R: It is a variant of MHT-GNN that removes TSS Encoder.

We adopt the same score module design (i.e., a linear mapping layer followed by a sigmoid function) as MHT-GNN for baselines without a score module. All methods adopt Adam optimizer if possible. We set initial learning rate to be 0.001 and use 128 as the dimension of representations. We use a batch size of 256. All methods will terminate optimization when they converge.

Evaluation Metrics. We use five widely adopted evaluation metrics:

- **AUC**: It signifies the probability that the positive sample's score is higher than the negative sample's score.
- **KS**: It is a measure of the degree of separation between the positive and negative distributions [11].
- **Precision, Recall and F1-score**: Precision is a measure of how many positive predictions are correct while Recall measures how many positive cases the classifier correctly predicted over all the positive cases in the data. F1-score is the harmonic mean of Precision and Recall.

Table 1. Overall detection performance.

Method	AUC	KS	Precision	Recall	F1-score
XGBoost	0.7452	0.3385	0.5783	0.3224	0.4140
MLP	0.7248	0.3375	0.5809	0.3287	0.4193
GCN	0.7946	0.4560	0.6488	0.4670	0.5431
GAT	0.8060	0.4801	0.6443	0.4869	0.5547
RGCN	0.8483	0.5308	0.7063	0.5373	0.6097
Simple-HGN	0.8498	0.5452	0.6972	0.5556	0.6183
IHG-Encoder	0.8623	0.5642	0.6907	0.5990	0.6415
AddGraph	0.8239	0.4949	0.6505	0.5716	0.6085
AddGraph-H	0.8416	0.5499	0.6467	0.6078	0.6251
MHT-GNN	**0.8969**	**0.6397**	**0.7297**	**0.6943**	**0.7115**

Table 2. Results of ablation study.

Method	AUC	KS	Precision	Recall	F1-score
IHG-Encoder	0.8623	0.5642	0.6907	0.5990	0.6415
MHT-GNN-R	0.8853	0.6169	0.7222	0.6853	0.7031
MHT-GNN-T	0.8856	0.6239	0.7300	0.6622	0.6944
MHT-GNN	**0.8969**	**0.6397**	**0.7297**	**0.6943**	**0.7115**

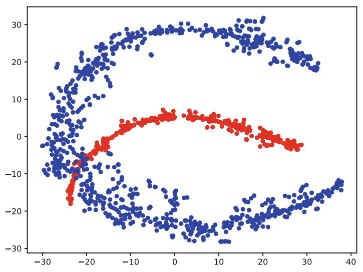

Table 3. t-SNE Projection of User Node Representations Generated by MHT-GNN: (1) Red Nodes: Fraudsters. (2) Blue Nodes: Normal users. (Color figure online)

Overall Detection Results. Table 1 presents the overall results of each method. The best performance are denoted in bold. From the results, we can see:

1. GNN-based approaches GCN and GAT generally exceed non-GNN methods XGBoost and MLP, indicating the spatial dependencies depicted by graph structure in WeChat contain rich information that can improve model performance in the CFD task.
2. Both dynamic and heterogeneous graph based models could achieve satisfactory results, and heterogeneous graph based methods generally outperform homogeneous graph based approaches. This observation shows that temporal dependencies and multi-relation information help model a user's behavior pattern better and boost the detection accuracy.

3. IHG-Encoder significantly outperforms other GNN-based methods. This observation has supported our decision of using IHG-Encoder as the backbone of MHT-GNN.
4. MHT-GNN achieves much better performance than other baselines including state-of-the-art dynamic graph anomaly detection methods AddGraph and its multi-relation version AddGraph-H. MHT-GNN consistently outperforms all baselines on all measures. The results demonstrate the superiority of MHT-GNN over existing methods for the CFD task.

Ablation Study. Table 2 lists the results of different variations of MHT-GNN. From Table 2, we can observe that:

1. The incorporation of either TSS Encoder or URS Encoder brings performance gain, as both MHT-GNN-T and MHT-GNN-R outperform IHG-Encoder.
2. The complete MHT-GNN shows the best result, indicating that modeling two views of historical data in HTG together can remedy the limitation of capturing only one view.

Overall, we can conclude that each module in MHT-GNN indeed contributes to the superior performance of MHT-GNN over existing detection methods in the CFD task of WeChat.

Visualization of Representation. To investigate the qualities of node representations generated by MHT-GNN, we adopt t-SNE [9] to project representations of nodes in the test set into a 2-dimensional space. The projection result is visualized in Fig. 3. From the result, we can see that representations of fraudsters and normal users have a clear distinction, showing that MHT-GNN is able to produce high-quality representations for the CFD task of WeChat.

5 Conclusion

In this paper, we illustrate MHT-GNN for the CFD task in WeChat. MHT-GNN can capture dynamics and diversity of MMMA data through multi-view learning and graph representation learning. Experiments on a real-world graph extracted from the CFD task in WeChat demonstrate the effectiveness of MHT-GNN. In the future, we will introduce attention mechanism for both intra-relation and inter-relation aggregation to adaptively assign weights for modeling the importance of information and further improve detection results. We also plan to enhance the interpretability of detection results so that fewer normal users will be wrongly labeled as fraudsters.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (No. 62002303, 42171456), the Natural Science Foundation of Fujian Province of China (No. 2020J05001), the China Fundamental Research Funds for the Central Universities (No. 20720210098), and 2021 Tencent WeChat Rhino-Bird Focused Research Program.

References

1. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl. Discov. **29**(3), 626–688 (2015)
2. Cai, L., et al.: Structural temporal graph neural networks for anomaly detection in dynamic graphs. In: CIKM, pp. 3747–3756 (2021)
3. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: KDD, pp. 785–794 (2016)
4. Ding, K., Li, J., Bhanushali, R., Liu, H.: Deep anomaly detection on attributed networks. In: SDM, pp. 594–602 (2019)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
6. Li, N., Sun, H., Chipman, K.C., George, J., Yan, X.: A probabilistic approach to uncovering attributed graph anomalies. In: SDM, pp. 82–90 (2014)
7. Lv, Q., et al.: Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In: KDD, pp. 1150–1160 (2021)
8. Ma, X., et al.: A comprehensive survey on graph anomaly detection with deep learning. IEEE Transactions on Knowledge and Data Engineering (2021)
9. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
10. Manzoor, E.A., Milajerdi, S.M., Akoglu, L.: Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In: KDD, pp. 1035–1044 (2016)
11. Massey, F.J.: The kolmogorov-smirnov test for goodness of fit. J. Am. Stat. Assoc. **46**(253), 68–78 (1951)
12. Peng, Z., Luo, M., Li, J., Xue, L., Zheng, Q.: A deep multi-view framework for anomaly detection on attributed networks. IEEE Transactions on Knowledge and Data Engineering (2020)
13. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC, pp. 593–607 (2018)
14. Shi, C., Li, Y., Zhang, J., Sun, Y., Yu, P.S.: A survey of heterogeneous information network analysis. IEEE Trans. Knowl. Data Eng. **29**(1), 17–37 (2017)
15. Sricharan, K., Das, K.: Localizing anomalous changes in time-evolving graphs. In: SIGMOD, pp. 1347–1358. ACM (2014)
16. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
17. Wang, L., et al.: TCL: transformer-based dynamic graph modelling via contrastive learning. arXiv Preprint (2021). https://arxiv.org/abs/2105.07944
18. Wang, Y., Zhang, J., Guo, S., Yin, H., Li, C., Chen, H.: Decoupling representation learning and classification for GNN-based anomaly detection. In: SIGIR, pp. 1239–1248 (2021)
19. Xu, Z., et al.: Efficiently answering k-hop reachability queries in large dynamic graphs for fraud feature extraction. In: MDM, pp. 238–245 (2022)
20. Yu, W., Cheng, W., Aggarwal, C.C., Zhang, K., Chen, H., Wang, W.: NetWalk: a flexible deep embedding approach for anomaly detection in dynamic networks. In: KDD, pp. 2672–2681 (2018)
21. Zheng, L., Li, Z., Li, J., Li, Z., Gao, J.: AddGraph: anomaly detection in dynamic graph using attention-based temporal GCN. In: IJCAI, pp. 4419–4425 (2019)