2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00289

Turbo: Fraud Detection in Deposit-free Leasing Service via Real-Time Behavior Network Mining

Sihao Hu[1,3], Xuhong Zhang[1,*], Junfeng Zhou,[1] Shouling Ji[1,*], Jiaqi Yuan[1],
Zhao Li[3], Zhipeng Wang[2], Qi Chen[1], Qinming He[1], Liming Fang[4]

[1]*Zhejiang University, Hangzhou, China*
{husihao26,zhangxuhong,zhoujf620,sji,yjq,chenqi,hqm}@zju.edu.cn
[2]*Jimi Store, Hangzhou, China*
wangzhipeng@jimistore.com
[3]*Alibaba Group, Hangzhou, China*
{sihao.hsh,lizhao.lz}@alibaba-inc.com
[4]*Nanjing University of Aeronautics and Astronautics, and Astronautics, Nanjing, China*
fangliming@nuaa.edu.cn

***Abstract*—Online deposit-free leasing service has witnessed rapid growth in China and shows a promising market in the future. While eliminating the requirement of a deposit does attract more users to the service, it also lowers the cost for fraudsters. Since the emergence of this service is relatively new, there are few works in literature focusing on detecting fraud transactions in it. Existing efforts mainly fall into hard-coded solutions such as block-listing or scorecard methods, which can be impotent in the face of the diverse fraud tactics, e.g., identity theft, or even suffering concept drift problem as the tactics evolve.**

In this paper, we contribute Turbo, an efficient graph-based anti-fraud system, to fully exploit the abundant user behavior logs in a real-time manner. Turbo is able to additionally make use of the implicit user relationships beyond the user features in the logs. To capture the user relationships, we first propose a novel algorithm to construct a time-evolving user behavior network called BN. Empirical analysis demonstrates that fraudsters in BN exhibit unique temporal aggregation and homophilic patterns, which inspires us to develop a novel heterogeneous adaptive graph neural network algorithm called HAG. Specifically, in HAG two graph operators are presented to mitigate the over-smoothing problem and make better use of the heterogeneous behavior relations in BN. Extensive experiments on a real-world dataset show that our method outperforms state-of-the-art methods significantly and can give a response in seconds for each detection request.

I. Introduction

The sharing economy in China has witnessed rapid development in the past three years and has the potential to sustain this growth in the future. According to the statistics [24], in the year of 2018, around 2,942 billion RMB sharing-economy transactions were made with an annual growth rate of over 40%. The number of participants involved was 760 million, which accounts for 54% of China's total population. In this prosperous market, the sharing services with deposits as its core or even primary source of profit are gradually phased out, e.g., the shutdown of the famous bike-sharing company ofo [31] marks the death of deposit oriented sharing services. In contrast, deposit-free leasing service, which mainly operates based on the credits of users, is getting popular and gradually becoming an industry trend. While eliminating the deposit requirement does attract more users to the service, it also lowers the cost for fraudsters. In China, more than 100 million stolen accounts are controlled by the grey industries, which incurs over 10 billion RMB annual profit [25]. The grey industries, including identity credit packaging, false identity provision, and fraud tactic teaching, can incur a massive risk of asset loss for the credit-based leasing companies.

Although considerable efforts have been made to detect fraud in e-commerce [16], [20], [35], few works in literature are focusing on fraud detection in the deposit-free leasing services, which is partially due to the newly emergence of these services. Furthermore, existing e-commerce anti-fraud approaches cannot be directly applied to the online leasing services mainly due to the following three reasons:

- **Failure to exploit the abundant user behavior logs.** Many works [21], [26], [34] take handcrafted profile features as the primary data source while ignoring the user behavior logs, which make them impotent to deal with the diverse fraud tactics, like identity theft or credit packaging, and may even suffer from concept drift problem [13] as fraud tactics evolve.
- **Unable to exploit the implicit relations between users.** In online leasing, there are no explicit relations like the deals between buyers and sellers or the reviews from users to products, which prevents most existing dedicated graph-based approaches [3], [16], [35] from being applied directly to our scenario. Moreover, the implicit relation revealed by the behavior logs such as location co-occurrence contains too much uncertainty, e.g., the probability that two users appear in the same place by chance is relatively high, making the direct use of these relations to detect fraudsters problematic.
- **Not supporting real-time detection.** Fraud in online leasing application needs to be detected immediately, while many e-commerce fraud detection systems [20], [34], [35]

* Xuhong Zhang and Shouling Ji are the co-corresponding authors.

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00289

are designed to work offline. Moreover, their transductive nature and prohibitive computational costs hinder them from achieving real-time efficiency.

As a result, the hard-coded solutions like block-listing [8] or scorecard methods [1], [32] are still the leading risk management tools used by most deposit-free leasing platforms such as Jimi Store[1].

With the increasing economic losses caused by fraudsters, an effective anti-fraud system is urgently required by the deposit-free leasing industry. To address the above-mentioned drawbacks, we contribute **Turbo**, a real-**T**ime **u**se**r** **b**ehavi**or** network based anti-fraud system. The system is designed to additionally make use of the implicit user relations beyond the user features from the logs. These implicit relations mainly reveal the co-occurrences of user behaviors, e.g., appearing at the same location, using the same Wi-Fi, etc. As the users' frequency of using the online leasing app increases, the volume of the logs becomes tremendous and thus makes the implicit relations a potential strong signal in fraud detection.

First, to capture the implicit user relations accurately, we propose a novel algorithm to construct BN from the user behavior logs. Our algorithm adopts hierarchical time windows and fine-grained weight calculation rules to alleviate the uncertainty in the implicit user relations.Analytical experiments conducted on BN unearth the unique temporal aggregation and homophilic patterns of fraudsters (Section III-B), indicating that the fraud behaviors in online leasing services cluster both in time and topology.

Second, inspired by the analysis, we propose a novel Heterogeneous Adaptive Graph (HAG) neural network algorithm to detect fraud transactions in online leasing services in a real-time manner. Based on the unique properties of BN, HAG makes the following three improvements over the basic GNN paradigms [12], [36]: 1) The implicit user relations naturally exhibit the typical fully-connected subgraph structure called *clique* in BN. This clique structure can lead to a severe over-smoothing problem in existing GNNs which project any node in a *clique* to the same point in representation space. Self-aware Aggregation operator (SAO) is proposed to mitigate this effect, which adaptively aggregates the hidden representations from itself and its neighbors to keep a balance between merging the neighbor context information and maintaining one's original information; 2) Based on the observation that the certainty of edges varies by types, and the contribution of a specific type to the final representation also varies across nodes, Cross-type Fusion Operator (CFO) is presented to model the edge heterogeneity at both macro(graph)- and micro(node)-level; 3) To support the real-time response requirements and generalize to newly-coming nodes, Turbo is designed to work in an inductive context like GraphSAGE [7] by taking a computation subgraph of the target user as an input instead of the entire BN.

We conduct a case study on Jimi Store, the first and largest smart-device online leasing platform in China, and present the first quantitative exploration of fraud detection in the deposit-free leasing industry. By leveraging the user behavior logs generated by 67,072 users from Jan. 2017 to Jun. 2018, our work achieves superior performance comparing to existing methods. Hitherto, Turbo has been deployed for real-time fraud detection in Jimi Store since Jul. 1, 2019. The online evaluation further illustrates its effectiveness. To summarize, the key contributions of this paper are:

- We propose a time-evolving heterogeneous graph to extract the implicit user relations from real-time user behavior logs. Carefully designed rules are used in the graph construction process to alleviate the uncertainty of the implicit relations.
- Based on the analysis of BN, we propose a dedicated algorithm named HAG, which consists of two modified GNN operators to better model the topology of BN, and validate its effectiveness on a large real-world dataset.
- We deploy the entire **Turbo** system in a real-world scenario and achieve significant performance improvements.

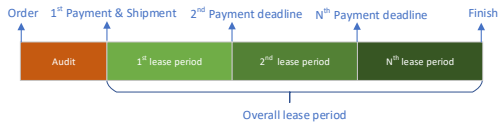


Fig. 1. Timeline of deposit-free leasing process

II. Preliminary

A. Leasing Procedure

After registration online, users could initiate applications for commodity leasing. Before their qualifications are confirmed, applicants need to complete personal profiles, including name, age, occupation, and phone number, etc., and upload their photo ID for identity verification. The profile information will first be verified against third party data. The verified profile information together with the credit information and historical transactions of a applicant will be audited by a risk management system within a business day. After passing through the audit process, the applicant should finish rent payment for the 1^{st} lease period within the required time before receiving the goods. At the end of each lease period, the applicant should pay rental for the next lease period until the entire lease expires. Figure 1 demonstrates the timeline of the overall leasing process. In this paper, a transaction refers to an application that has passed through the audit process.

B. Problem Formulation

Deposit-free leasing fraud. Based on the experts' domain knowledge, a fraudster is defined as a user who passes through the audit process but only pays rent for the first 1 or 2 lease period(s) after receiving the products, and then stops the payments and does not return the leased products. The fraudster's corresponding transaction is defined as a fraudulent transaction. In this paper, we focus on detecting fraudsters and their corresponding applications in the audit process.

Fraud detection task. Let $\mathcal{U}$ be a set of users, and $\mathcal{T}$ the set of corresponding transactions; each transaction $\tau \in \mathcal{T}$ belongs

[1]https://www.jimistore.com

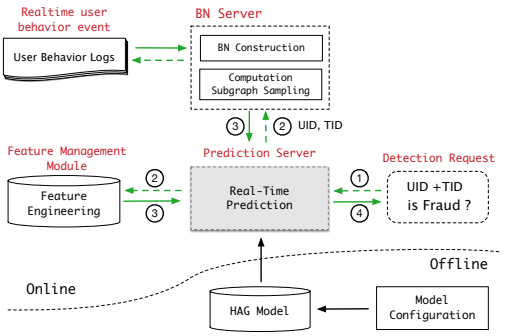


Fig. 2. Online anti-fraud procedure

to a specific user $u \in \mathcal{U}$. $b_u^t \in \mathcal{B}_u$ represents a behavior log of user u at time t in a form of $[u, r, s, t]$, where r is a type of b_u^t and s is a certain value of type r. Each user u's profile information and credit history are denoted as $\mathcal{X}_u$, and the features of a transaction τ are denoted as $\mathcal{X}_\tau$. In light of the above, we define the goal of the deposit-free leasing fraud detection as to estimate $\mathcal{P}(\mathcal{Y}|\mathcal{X}_\tau, \mathcal{X}_u, \mathcal{B}_u)$, the probability that user u swindles the commodity in an application τ ($\mathcal{Y} = 1$) or not ($\mathcal{Y} = 0$).

C. Online Inference

Figure 2 gives a brief illustration of the online inference process of Turbo. It consists of four key components: BN server, feature management module, real-time prediction server, and model management module. Specifically, when a client makes a prediction request, the prediction server asks BN server to sample a computation subgraph for the target user u, along with features $\mathcal{X}_u$ and $\mathcal{X}_\tau$ constructed by feature management module simultaneously. With the returned subgraph and its corresponding features, the prediction server makes a real-time prediction and returns the result to the client. The execution order is marked by the number in Figure 2. In Turbo, HAG is adopted as the classification model and retrained offline on a daily baisis. In the following section, we will elaborate on not only BN construction and HAG algorithm, but also the empirical studies and potential insights that underlie these algorithms.

It is worth noting that Turbo system works in a log-sufficient way. If there is a lack of logs for new-registered users, it is better to trigger Turbo a short period after the user submits the application to acquire sufficient logs.

III. Behavior Network

Existing graph-based anti-fraud methods seek to capture the explicit relations between entities [15], [20], [23], [35]. However, it is rather difficult to extend them to the online leasing scenario where no explicit relations exist. Furthermore, the only available implicit user relations come from the real time user logs, which make the underlying graph highly dynamic and the construction/update of this dynamic graph from a large scale streaming user logs very challenging [5], [10]. Further adding to the complexity of exploiting these implicit user relations is the the uncertainty contained in these relations. To address these challenges, we introduce our algorithm to build BN. The associated empirical findings are also presented in this section.

TABLE I
Descriptions of Behavior Types for BN Construction

Type	Description
Device Id	The unique identifier for a mobile device.
IMEI	International Mobile Equipment Identity number.
IMSI	International Mobile Subscriber Identity number.
IPv4	Internet Protocol version 4 address.
Wi-Fi MAC	The MAC address of a Wi-Fi router.
GPS	Precise GPS coordinates of user location.
GPS_{100}	100-meter square of user GPS location.
GPS_{Dev}	Precise GPS coordinates of delivery address.
$\text{GPS}_{\text{Dev100}}$	100-meter square of GPS_{Dev}.
Workplace	User workplace address.

The user behavior dataset used in this section is provided by Jimi Store, which contains user behavior logs from Jan. 2017 to Jun. 2018. Due to privacy and security concerns, throughout the paper we only report the average statistics. Table I lists the behavior types and their descriptions.

BN is a time-evolving heterogeneous graph consisting of user nodes and multiple types of edges, which is designed to precisely capture the implicit relations between users in real-time. The implicit relations are mainly the co-occurrence relations, e.g., two users showing up at the same location and the same time indicates that they are related. Thus, the main idea of the construction method is straightforward: edges will be built between user nodes whose behavior logs share the same behavior type r and value s within a specified period. Obviously, the users nodes connected by these edges form a *clique*, which reveals the nature of the implicit user relations. Besides, it can be seen that the edge type is the same as the behavior type. To more accurately capture the relations between users, a fine-grained edge weight calculation is required. For example, if the number of users sharing the same behavior within a period is larger, then the weights between these users should be smaller. Additionally, if two users share the same behavior within a shorter period, then the weight between them should be larger. Thus, two strategies, *inverse weight assignment* and *hierarchical time windows*, are adopted for edge construction. The whole construction procedure is described in Algorithm 1.

Inverse weight assignment. In BN construction, time is discretized by time windows $\mathbf{W}$ into epochs like $(t_{j-1}, t_j]$. Within each time epoch, edges are established between the user nodes whose behavior logs share the same type r and value s, e.g., the weight of edge $\mathcal{E}(u, v)$ built by sharing s within $(t_{j-1}, t_j]$ is set to the inverse of the number of users connected through s within the j-th time epoch, i.e., $1/N_{j,s}$ (line 6), and is accumulated to the edge weight $w_r^W(u, v)$ (line 8) by traverse all the possible value s shared between v and u. Figure 3 further gives a toy example of edge building within a 1-hour time epoch (red dotted box): a fully-connected subgraph circled by a red dotted line is generated, and the edges all receive a weight of 1/4. This inverse relationship

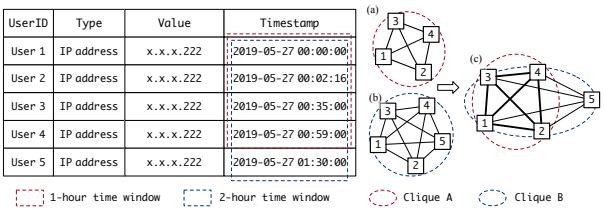


Fig. 3. A toy example of BN construction.

enables BN to measure the certainty of the implicit relations more accurately, e.g., a public Wi-Fi may lead to a large number of user nodes connected via the same IP address. Thus the weights of the edges between these users will be tiny according to the inverse weight assignment rule.

Hierarchical time windows. In Section III-B, we unearth some unique patterns of fraud behaviors, i.e., they tend to burst and be associated within a short period of time. Therefore the length of the time interval between associated behaviors would be a reliable indicator to separate two groups of users. To be aware of this difference, hierarchical time windows $\mathbf{W}$ are set to capture co-occurrence relations at different time granularities, where $\mathbf{W} = [W_1, W_2, \ldots, W_n]$ with $W_i < W_{i+1}$. In this setting, co-occurrences in a smaller time window will also be caught by all the larger time windows, and the weights of corresponding edges will be greater after summing up the weights obtained at different time granularities, i.e., $w^r_{u,v} = \sum_i^n w^{r,t_i}_{u,v}$ (line 10). As exemplified by Figure 3, time windows with different sizes generate two corresponding subgraphs (dotted circles). Based on the inverse rule, weights of edges built within the 1-hour time window are all 1/4, and the 2-hour time window 1/5. After summing up, the thickness of edges (value of weights) suggests that BN assigns higher weights to relations that appeared in a shorter interval. Typically, an empirical parameter $\mathbf{W}$ = [1 hour , 2 hours, ... , 12 hours, 1 day] is employed in our experimental setting.

Sampling & normalization. To make Turbo work in an inductive context and thus support real-time detection, we take a computation subgraph $\mathcal{G}_v$ as the input for HAG, where $\mathcal{G}_v$ denotes a subgraph that contains the complete information a GNN relies on for computing v's representation h_v. Note that $\mathcal{G}_v$ only includes the nodes having transactions. During online inference, $\mathcal{G}_v$ is sampled by the BN server after a detection request is made toward user v.

To account for the volume difference of different edge types, we further normalize the edge weight based on its edge type. The normalized edge weight is $w'_r(u,v) = w_r(u,v) \cdot (\deg'_r(u) \cdot \deg'_r(v))^{-1/2}$, where $w_r(u,v)$ is the original weight of the edge between node u and node v with edge type r, $\deg'_r(u) = \sum_{i \in \mathcal{N}_r(u)} w(u,i)$ is the weighted degree of node u on with edge type r.

A. Network Construction

B. Empirical Study

In this section, we conduct an empirical study to reveal some unusual patterns of the fraud behaviors in online leasing services, which provides the insights for the design of the following anti-fraud algorithm.

Algorithm 1: BN Construction

Input: Behavior logs in a form of [uid, r, s, timestamp], hierarchical time windows $\mathbf{W}$=[W_1, W_2, .., W_n]
Output: BN $\mathcal{G}$=($\mathcal{V}, \mathcal{E}, \mathcal{R}$)

1 **for** *each* r *in* $\mathcal{R}$, *each* W *in* $\mathbf{W}$ **do**
2 t_0 = initial time
3 **for** j=1, 2, ..., *max* **do**
4 $t_j = t_0 + j * W$
5 **for** *all pairs of* (u, v) *on* s *within* $(t_{j-1}, t_j]$ **do**
6 $w^W_{r,j}(u,v)$ += $1/N_{j,s}$;
7 **end**
8 $w^W_r(u,v)$ += $w^W_{r,j}(u,v)$;
9 **end**
10 $w_r(u,v)$ += $w^W_r(u,v)$
11 $\mathcal{E}_r(u,v)$ = (u, v, $w_r(u,v)$);
12 **end**

Time burst observation. Figure 4a and b show the distributions of behaviors for normal users and fraudsters, respectively. Each dot in the figures represents one behavior log of a user. The X-axis is time and the Y-axis is user ID. Thus, each horizontal line in the figure represents all the behavior logs of a particular user. Similar to [27], we find that fraudsters' behaviors tend to burst only within a short period and it is usually around the time of application, whereas the behavior logs of normal users uniformly scatter over the entire leasing period. This significant difference leads us to investigate the patterns of fraud behaviors. This observation also indicates that the logs available in the audit process should already include most of the logs of a potential fraudster and thus should be sufficient for fraud detection.

Temporal aggregation. Figure 4c is a violin plot of the temporal distribution of users having the same behaviors. Specifically, we first collect the pairwise absolute time intervals between the timestamps of the logs with the same behavior type r and value s. Then, we plot the time interval distributions of 7 behavior types for normal users and fraudsters, respectively. From the plot, we can observe that for each behavior type the time interval distribution of normal users decreases smoothly as the value of the interval increases, whereas that of fraudsters shows significant burst at the small intervals and then quickly decays. This suggests that fraudsters might be associated with each other and generate similar behaviors around the same time (usually 0∼3 days window). This temporal aggregation effect of fraud behaviors can be captured via our hierarchical time windows mechanism, which leads to a larger weight between the connected fraudster nodes.

Homophilic effect. Since fraudsters' behaviors show a temporal aggregation effect, we are also interested to know if they also cluster on topology, which is also the homophilic effect of fraudster nodes. In Figure 4d, we plot the ratios of the fraud nodes in the n-hop neighbors of normal user nodes and fraudster nodes, respectively. It is evident that: 1) the fraud ratio of the fraudster nodes' n-hop neighbors is much higher than that of the normal nodes' n-hop neighbors; 2) the

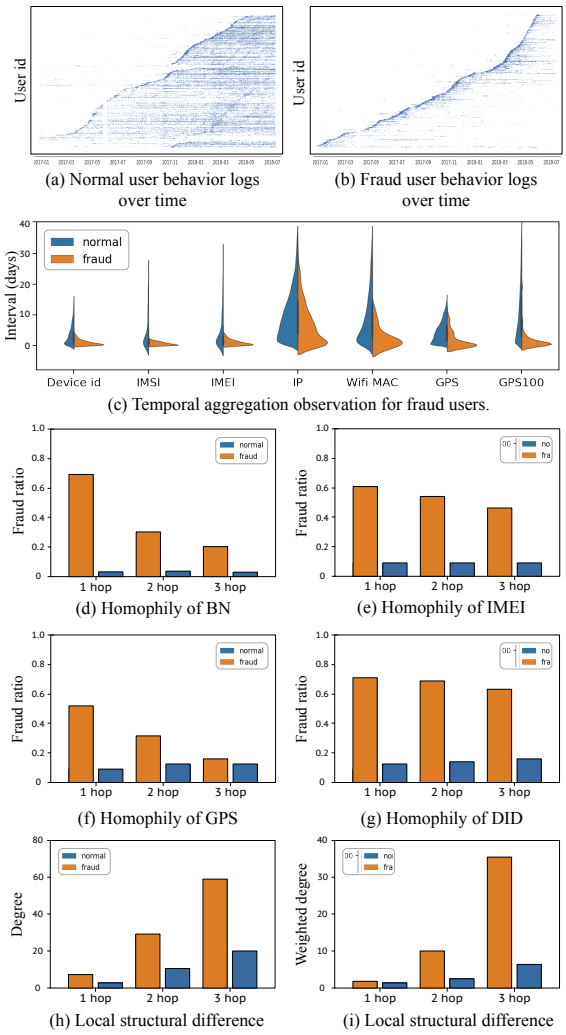


Fig. 4. The observational study of fraud behaviors. (a)-(b) are two distributions of behavior logs over time. (c) illustrates the temporal aggregation effect of fraud behaviors. (d-g) present the homophlic effect exhibited by BN. (h-i) illustrate the structural difference between the two groups of nodes.

fraud ratio of the fraudster nodes' n-hop neighbors decreases as the number of hops increases, while that of the normal nodes' n-hop neighbors is more stable with the increase of the number of hops. This indicates that fraudster nodes cluster on BN, which inspires us to leverage GNN-based algorithms to exploit the homophilic effect of local neighbors. Furthermore, the homophilic effects of the fraudster nodes with respect to different types of behavior edges are also shown in Figure 4e-g. It can be seen that the homophilic effects with respect to different behavior edges also differ, which suggests that our HAG algorithm should have the capability of modeling this heterogeneity.

Structural difference. We take a further look the local structure difference between the normal user nodes and fraudster nodes by examining their degrees. In Figure 4h, we observe that the average degree of the n-th hop neighbors of the fraudster nodes is much larger than that of the normal

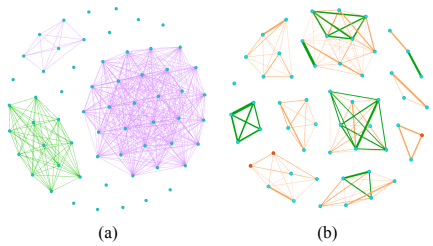


Fig. 5. Visualization of cliques in two subgraphs of BN: (a) is three separate *cliques*. (b) contains multiple overlapping *cliques*. Different colors of edges denote different edge types and the thickness of edges denotes the edge weight.

nodes. This phenomenon is further augmented when the edge weights are considered when calculating the degrees, which is shown in Figure 4i. These observations indicate that fraudster nodes tend to connect with more nodes and are more closely connected with them. Therefore, the use of a GNN based algorithm is a right choice, since GNN-based algorithms naturally have the ability to capture this structural difference.

IV. HAG ALGORITHM

In this section, we integrate the insights gained above into the design of our Heterogeneous Adaptive Graph (HAG) neural network. The core design of HAG is two new GNN operators: Self-aware Aggregation Operator (SAO) and Cross-type Fusion Operator (CFO). SAO is proposed to mitigate the over-smoothing problem caused by *cliques* that greatly degrade the expressive power of popular GNNs. Subsequently, CFO enables HAG to model the heterogeneity of BN at both macro(graph)- and micro(node)-level via assigning different importance to multi-type embeddings with node-wise adaptivity.

A. Self-aware Aggregation Operator (SAO)

Clique. As discussed in Section III-A, the nature of the implicit user relations reflected on the constructed BN is a superposition of multiple *cliques*. Figure 5 shows the visualization for two subgraphs of BN to demonstrate clique structures, where Figure 5a shows three separate *cliques* and Figure 5b shows an superposition of multiple *cliques*. The thickness of the edges denotes the edge weights, and different colors represent different edge types.

Obviously, in a single *clique* $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$, each node's neighborhood is all the same, which can decrease the performance of GNNs, since the expressive power of them lies in recursively aggregating feature vectors of neighboring nodes [36]. To prove the degraded expressiveness of GNNs, we discuss the two most typical aggregation schemes: GCN [12] and GraphSAGE [7] in Eq. 1 and Eq. 2, respectively.

$$\mathbf{h}_v^{(l)} = \mathrm{ReLU}(\mathbf{W}_l \cdot \sum_{u \in \tilde{\mathcal{N}}_v} (\deg(v) \cdot \deg(u))^{-1/2} \cdot \mathbf{h}_u^{(l-1)}) \quad (1)$$

$$\begin{aligned} \mathbf{h}_{\mathcal{N}_v}^{(l)} &= \tfrac{1}{\deg(v)} \cdot \textstyle\sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(l-1)}, \\ \mathbf{h}_v^{(l)} &= \mathrm{ReLU}\left(\mathbf{W}_l \cdot \left(\mathbf{h}_v^{(l-1)}; \mathbf{h}_{\mathcal{N}_v}^{(l)}\right)\right) \end{aligned} \quad (2)$$

where $\mathbf{W}_l$ is a layer-specific trainable weight matrix, $\mathcal{N}_v = \{u \in \mathcal{V} | (v,u) \in \mathcal{E}\}$ and $\tilde{\mathcal{N}}_v = \{v\} \cup \mathcal{N}_v$. $\mathbf{h}_v^{(l)}$ is v's hidden feature learned by l-th layer of GNN, and ; is the concatenation operator. GraphSAGE (Eq. 2) is also named skip-connection paradigm since it shares skip paths between different layers. Generally speaking, a bunch of GNNs [16], [20], [37] can be deemed as variants of these two aggregation schemes.

Over-smoothing problem. Over-smoothing is the most severe problem GNNs could encounter when dealing with *cliques*, i.e., embeddings of nodes in the same *clique* will be mapped into nearly identical locations in a hidden space after one-round calculation. It can lead to undesirable consequences in risk management, e.g., a fraudster and a normal user connected via a public Wi-Fi cannot be separated by the decision boundary, even if they are distinguishable in the original feature space. To facilitate further analysis, we adopt the influence score and distribution [14], [37] metrics to measure the influence effect between nodes:

Definition 1. ***Influence score and distribution.*** *The influence score $S_i(j)$ of node i by node j in i's computation subgraph is defined as the sum of the absolute values of the entries of the Jacobian matrix $\left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}}\right]$. The influence distribution D_i is defined by normalizing the influence scores: $D_i(j) = \mathrm{S}_i(j) / \sum_k \mathrm{S}_i(k)$.*

First, we investigate the extreme case of over-smoothing problem. For convenience, all the discussion below is for homogeneous *clique* $\mathcal{G}_c$, where types and weights are all the same and omitted during description in this section. Through analyzing the influence score of nodes in $\mathcal{G}_c$ to demonstrate the expressive power of GCN, we can derive the following theorem:

Theorem 1. *Given a l-layer GCN with aggregation as in Eq. 1, assume that all paths in the computation DAG (directed acyclic graph, i.e., the computation graph of a neural network.) are activated with the same probability ρ. Take $\mathcal{G}_c$ as the input computation subgraph for node u, the influence score $S_u(v)$ for any node $v \in \mathcal{G}_c$ is equivalent in expectation, and the k-th ($k > 0$) layer's hidden feature $\mathbf{h}_v^{(k)}$ is identical in expectation.*

Proof. We use $\mathbf{y}_i^{(n)}$ to denote the feature before activation at l-th layer. For any $n = 1, .., l$, $\mathbf{y}_i^{(n)} = \frac{1}{\tilde{\deg}(x)} \cdot \sum_{j \in \tilde{\mathcal{N}}_i} \mathbf{W}_n \mathbf{h}_j^{(n-1)}$. for any $n > 0$, we have

$$\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(n-1)}} = \frac{1}{\tilde{\deg}(i)} \cdot \mathrm{diag}(1_{\mathbf{y}_k^{(n)} > 0}) \cdot \mathbf{W}_n$$

Due to the property of fully-connected structure, $\tilde{\deg}(\cdot) \equiv m$, and there are m^n computation paths from node i to node j. We denote the x-th path as a_x^n, a_x^{n-1},.., a_x^1, a_x^0, where a_x^n is node i and a_x^0 is node j. All nodes are neighbors to each other. By applying the chain rule, we can get

$$\begin{aligned}\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}} &= \sum_{x=1}^{m^n} \left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}}\right]_x \\ &= \frac{1}{m^n} \sum_{x=1}^{m^n} \prod_{k=n}^{1} \mathrm{diag}\left(1_{\mathbf{y}_{a_x^k}^{(k)} > 0}\right) \cdot \mathbf{W}_k\end{aligned}$$

For each path x, the derivative $\left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}}\right]_x$ can be considered as a computation DAG (directed acyclic graph) and formalized as below

$$\left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}}\right]_x^{(u,v)} = \frac{1}{m^n} \sum_{y=1}^{\Psi} \xi_y \prod_{k=n}^{1} \mathbf{w}_x^{(k)} \tag{3}$$

Where Ψ is the total number of computation paths y from the input node u to the output node v in the computation graph of $\left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_y^{(0)}}\right]_j$. $\mathbf{w}_y^{(k)}$ is the weight entry of $\mathbf{W}_k$ used in the y-th path for each layer k. The random variables ξ_y denotes whether the y-th path is active or not and obeys Bernoulli distribution. If there any node in y-th path is deactivated by $\mathrm{ReLU}(\cdot)$, $\xi_y = 1$, otherwise $\xi_y = 0$. Based on the assumption, for each path y, the probability of success equals to ρ. Take the expectation of Equation 3, we can get

$$\mathbb{E}\left[\left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}}\right]_x^{(u,v)}\right] = \frac{\rho}{m^n} \cdot \prod_{k=n}^{1} \mathbf{w}_x^{(k)}$$

Add m^n paths from node i to node j, we have

$$\mathbb{E}\left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}}\right] = \rho \prod_{k=n}^{1} \mathbf{W}_k$$

Thus we can draw a conclusion that the $\mathbb{E}\left[\frac{\partial \mathbf{h}_i^{(n)}}{\partial \mathbf{h}_j^{(0)}}\right]$ is equivalent for each pairwise (i,j) in $\mathcal{G}_c$. Based on the definition of Influence Score, $\mathbb{E}[\mathrm{S}_i(j)] = ||\rho \prod_{k=n}^{1} \mathbf{W}_k||_1$ and $\mathbb{E}[\mathrm{D}_i(j)] = 1/m$, where m is the node number in *clique* $\mathcal{G}_c$. For each node i in $\mathcal{G}_c$, $\mathbf{h}_i^{(n)}$ share the same influence from any node in $\mathcal{G}_c$. Using the similar proof techniques above, we can easily prove that the n-th ($n > 0$) layer's hidden feature $\mathbf{h}_i^n$ in $\mathcal{G}_c$ is identical in expectation. □

Theorem 1 denotes that scheme like Eq. 1 fails to distinguish node's self-feature from neighborhood features with the same expected influence score $\mathbb{E}[S_u(*)]$ and distribution $\mathbb{E}[\mathrm{D}_u] = 1/\deg(u)$, which means GCN will project all nodes of $\mathcal{G}_c$ into the same point in hidden space after only once aggregation operation. Furthermore, since Xu et al. [37] have discussed the relationship between the original GCN (Eq.1) and a random walk-liked GCN, a similar conclusion on the latter can be derived by replacing $(\deg(v) \cdot \deg(u))^{-1/2}$ with $\deg(v)^{-1}$. A straightforward way to mitigate this one-round over-smoothing problem is to separate self-feature from neighborhood features

as skip-connection does (Eq.2). Through a simple matrix operation, we can derive the following formula from Eq.2:

$$\mathbf{h}_v^{(l)} = \mathrm{ReLU}(\mathbf{W}_{ls} \cdot \mathbf{h}_v^{(l-1)} + \frac{1}{\deg(v)} \cdot \mathbf{W}_{ln} \sum_{u \in \mathcal{N}_v} \cdot \mathbf{h}_u^{(l-1)}) \quad (4)$$

where $\mathbf{W}_l = (\mathbf{W}_{ls}; \mathbf{W}_{ln})$. Eq. 4 suggests that although GraphSAGE can make aware of difference between the target node and its neighbors, the affine transformation matrices $\mathbf{W}_{ls}$ and $\mathbf{W}_{ln}$ are optimized in a way that works best for the overall dataset, which is insufficient to overcome the impact of *clique* after several aggregation operations. Moreover, one may find it work well on a small graph that requires less adaptability, yet considering the ubiquitous *clique* structure and the scale of BN, an aggregation operator that can keep node-wise adaptability is preferred.

Method. Based on the above analysis, SAO is proposed to acquire node-level adaptivity during aggregation. The underlying intuition is straightforward: distinguishing self-feature and neighborhood features via learned attention scores. It is worth noting that SAO is operated independently on each homogeneous sub-graph $\mathcal{G}^r$=$(\mathcal{V}, \mathcal{E}, r \in \mathcal{R})$ which is formed by all the edges with the edge type r. For description convenience, we omit the type symbol for a specific edge type r and formulate the operator as follows:

$$\mathbf{h}_v^{(l)} = \mathrm{ReLU}(\alpha_{self} \cdot \mathbf{W}_{ls} \cdot \mathbf{h}_v^{(l-1)} + \alpha_{neigh} \cdot \mathbf{W}_{ln} \cdot \mathbf{h}_{\mathcal{N}_v}^{(l-1)}) \quad (5)$$

$$\mathbf{h}_{\mathcal{N}_v}^{(l-1)} = \frac{1}{\deg(v)} \sum_{u \in \mathcal{N}_v} w_{uv} \cdot \mathbf{h}_u^{(l)} \quad (6)$$

$$\alpha'_{self} = p^{\mathrm{T}} \cdot \tanh(\mathbf{W}_s \mathbf{h}_v^{(l-1)}; \mathbf{W}_s \mathbf{h}_v^{(l-1)}) \quad (7)$$

$$\alpha'_{neigh} = p^{\mathrm{T}} \cdot \tanh(\mathbf{W}_n \mathbf{h}_{\mathcal{N}_v}^{(l-1)}; \mathbf{W}_s \mathbf{h}_v^{(l-1)}) \quad (8)$$

$$\alpha_{self} = \frac{\exp(\alpha'_{self})}{\exp(\alpha'_{self}) + \exp(\alpha'_{neigh})}, \alpha_{neigh} = 1 - \alpha_{self} \quad (9)$$

where $\mathbf{h}_{\mathcal{N}_v}^{(l-1)}$ is a whole representation for the neighborhood $\mathcal{N}_v$; α'_{self} and α'_{neigh} are served as the importance coefficients for $\mathbf{h}_v^{(l-1)}$ and $\mathbf{h}_{\mathcal{N}_v}^{(l)}$ during aggregation. We parameterize the attention score with a self-attention layer in Eq. 7 and Eq. 8, where $\mathbf{W}_{ls}, \mathbf{W}_{ln} \in \mathbb{R}^{d_l, t}$, $p \in \mathbb{R}^t$ are trainable parameters. d_l denotes the l-th hidden layer size of SAO, and t denotes the hidden layer size of the attention layer. The attention scores are normalized through $\mathrm{softmax}(\cdot)$ function in Eq. 9.

SAO can also be view as one kind of gate mechanism to adaptively control the amount of information propagated from a node's neighbors. Similar application of gate mechanism is proposed in CLN-HWN [29] to tackle the difficulty of making previous states propagate through layers in the feed-forward nets with a high number of layers. However, in CLN-HWN, gate is applied on current layer's state $h_v^{(l)}$ and previous layer's state $h_v^{(l-1)}$, while the gate in SAO is applied on a node's self feature $h_v^{(l)}$ and its neighborhood features $h_{\mathcal{N}_v}^{(l)}$. Moreover, the gate in SAO is computed via attention, which is totally different from that in CLN-HWN.

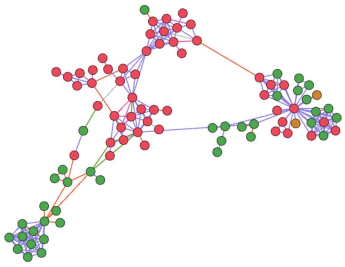


Fig. 6. Visualization of an example BN. A total of 91 vertices (fraud in red, normal in green, and under reviewing in yellow) and multiple types of edges: purple means GPS, green means IMEI, orange means IP, red means IMSI, gray means GPS of delivery address.

B. Cross-type Fusion Operator (CFO)

In the previous section, SAO is introduced to tackle the homogeneous *cliques* in each sub-graph $\mathcal{G}^r$, while BN $\mathcal{G}$ exhibits heterogeneity with multiple edge types and can be viewed as a superstition of $\mathcal{G}^r, r \in \mathcal{R}$ as visualized in Figure 6. It is intuitive to consider the fact that the certainty contained in edges varies by types, and the contribution of a specific type r to the final representation varies from the chosen node v. Motivated by this, we empower HAG to tackle BN's heterogeneity via CFO, an operator that can model the contribution of a certain type in both macro(graph)- and micro(node)- level. First, the l-th layer hidden embedding $\mathbf{h}_{v,r}^{(l)}$ on type r is generated by SAO as:

$$\mathbf{h}_{v,r}^{(l)} = \mathrm{SAO}(\{\mathbf{h}_{u,r}^{(l-1)}, \forall u \in \tilde{\mathcal{N}}_{v,r}\}) \quad (10)$$

where $\tilde{\mathcal{N}}_{v,r}$ is neighbor set of node v on type r. We denote the k-th (final) layer output embedding $\mathbf{h}_{v,r}^{(k)}$ as type embedding $\mathbf{h}_{v,r}$, and then concatenate all the type embeddings for node v as $\mathbf{H}_v$ with size $\mathbb{R}^{d_k \times |\mathcal{R}|}$, where d_k is dimension of $\mathbf{h}_{v,r}^{(k)}$:

$$\mathbf{H}_v = (\mathbf{h}_{v,1}, \mathbf{h}_{v,2}, \ldots, \mathbf{h}_{v,|\mathcal{R}|}) \quad (11)$$

Instead of merely assigning global attention scores to fuse the multi-type embeddings [20], we employ the self-attention mechanism to acquire node-wise attention coefficients as follows:

$$\alpha_{v,r} = \mathrm{softmax}_r(\mathbf{v}_r^{\mathrm{T}} \tanh(\mathbf{W}_r \mathbf{H}_v))^{\mathrm{T}} \quad (12)$$

where $\mathbf{v}_r \in \mathbb{R}^{d_a}$ and $\mathbf{W}_r \in \mathbb{R}^{d_a \times d_k}$ are trainable parameters for type r and will be estimated through back-propagation process. Each $\alpha_{v,r}$ is a $|\mathcal{R}|$-length vector representing the attention coefficients for $\mathbf{h}_{v,r}$. To further characterize the importance of type r in a macro-level, we assign a transformation matrix $\mathbf{M}_r \in \mathbb{R}^{d_k \times d_m}$ to type r, and the operation for type r can be formulated as $\mathcal{H}_{v,r} = \mathbf{M}_r^{\mathrm{T}} \cdot \mathbf{H}_v \cdot \alpha_{v,r}$. Generally, the whole CFO can be formulated as follows:

$$\mathcal{H}_v = \mathcal{M}^{\mathrm{T}} \cdot \mathbf{H}_v \cdot \boldsymbol{\alpha}_v \quad (13)$$

$$\boldsymbol{\alpha}_v = (\alpha_{v,1}, \alpha_{v,2}, .., \alpha_{v,|\mathcal{R}|}) \quad (14)$$

$$\mathcal{M} = (\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_{|\mathcal{R}|}) \quad (15)$$

where $\boldsymbol{\alpha}_v$ is a coefficient matrix with size $|\mathcal{R}| \times |\mathcal{R}|$, $\mathcal{M}$ is a transformation tensor with size $d_k \times d_m \times |\mathcal{R}|$.

It is worth noting that although attention mechanisms are both leveraged, HAG differs from GATs [33] mainly in the following aspects: 1) GATs calculate attention coefficients to make one-time aggregation, while SAO assigns attention scores to one's self-embedding and aggregated neighborhood embedding to address the over-smoothing problem; 2) CFO fuses multi-type information via attention mechanism to better tackle the heterogeneity of edges, whereas GAT do not consider edge/node heterogeneity and mainly use attention to aggregate neighbor information.

V. System Implementation

In this section, we introduce the details about the implementation of Turbo as shown in Figure 2. In the following, we mainly introduce the implementations of real-time BN construction and feature preparation. Then we show the strategies for reducing the latency of retrieving the computation subgraph and the corresponding node features.

In our implementation, BN sever receives user behavior logs in real-time and then store the logs into the local database. Then, BN server periodically schedules a job to construct edges and their weights based on the new logs within each time window introduced in Algorithm 1, e.g., for a 1-hour time window, BN server schedules a hourly job to construct edges based on the new logs within the passed hour. The new edges will added to a global edge list stored in the local database. Our implementation makes jobs with shorter time window run more frequently to help BN capture a user's most recent behaviors timely. The underlying reason is that our online deployment indicates that a user's most recent behaviors contribute more to the final predication. Besides, a max TTL (Time To Live) is set to 60 days for each edge to prevent the monotonous increase of the graph. It is noted that these operations are in parallel to the subgraph sampling request, and thus the latency of BN construction is not counted into the latency of online prediction. When an audit request is sent to the system, the prediction server asks the BN server to construct a computation subgraph of the target user node by querying its k-hop neighboring nodes from the corresponding local database.

The next step is to retrieve the features for the sampled nodes from the feature management module. The node features include user behavior statistical features $\mathcal{X}_s$, user profile features $\mathcal{X}_u$ and application-related features $\mathcal{X}_\tau$. Specifically, $\mathcal{X}_s$ is calculated based on the behavior logs of each user, including the frequency of logins, the number of associated devices in 1 hour, 6 hours, 1 days, etc. Ideally, $\mathcal{X}_s$ should be calculated via a streaming processing framework such as Apache Flink[2]. However, at the time of our implementation, Jimi Store did not have streaming processing infrastructure. Thus, we alternatively compute $\mathcal{X}_s$ in real time, which accounts for most of the predication latency. Other static features are retrieved via normal database queries.

[2]https://flink.apache.org

To reduce the latency of computing $\mathcal{X}_s$ and retrieving the computation subgraph and other node features, we adopt the Redis [4] caching technology by additionally storing the graph, user profile and application features, and behavior logs into an in-memory database. Any update will synchronized to both the local database and in-memory database. Furthermore, a MySQL cluster and Redis cluster with disaster backup, primary-and-replica switching capabilities is provided to deal with data loss and the slow recovery when the database services crash.

Through the above optimization approaches, the efficiency and stability of Turbo have been effectively improved. Specifically, the average latency of the entire prediction decreases from 6.8 seconds to 0.8 seconds. The 50-th latency percentile (p50) drops from 6.73 seconds to 0.8 seconds; P99 drops from 11.3 seconds to 0.99 seconds; P999 drops from 12.66 seconds to 1.33 seconds. In general, the online operation time is reduced by 88%.

VI. Experiments

In this section, we evaluate the proposed methods on a real-world dataset to answer the following research questions:

- **Q1**: How well does Turbo perform on fraud detection in deposit-free online leasing services, comparing to state-of-the-art baselines?
- **Q2**: How do SAO and CFO contribute to the expressive power of HAG? Does SAO mitigate the over-smoothing problem, and CFO better model the heterogeneity of BN?
- **Q3**: How is the efficiency of Turbo in a real-world application? Can it achieve real-time response on magnitudes of data to support online inference?

Next, we present the experimental settings, followed by answering the above research questions one by one.

A. Experimental Setup

Dataset. Jimi datasets are tailored as a benchmark for the deposit-free leasing fraud detection task, which consist of the user behavior data introduced in Section III and the complete user/transaction data from Jan. 2017 to Jun. 2018 provided by Jimi Store. Table II shows the statistics of Jimi datasets and the constructed BN. Note that node u's feature $\mathcal{X}_{u+\tau}$ is the concatenation of its user feature $\mathcal{X}_u$ and transaction feature $\mathcal{X}_\tau$. D_1 includes all the applications that passed through the original risk management system and got labeled after finishing the lease period. However, D_2 includes all the applications initiated in the same period as D_1. Most of the applications ($>$90%) in D_2 did not pass through Jimi's original risk management system due to the relatively strict threshold setting, thus there are no label for most applications to indicate whether they are fraud or not. Instead, we take users who *are rejected by the original risk management system or passed through but are labeled as fraud* as the positive samples and users who *passed through the original risk management system and are labeled as normal* as the negative samples.

TABLE II
STATICS OF TWO JIMI DATASETS

Dataset	# node	# positive	# edge	# type
D_1	67,072	918	207,890	8
D_2	1,072,205	989,728	2,787,733	8

Baselines. We compare the empirical results of HAG with three types of baselines. The first type is classification methods based on handcrafted features, including Logistic Regression (LR), Support vector machine (SVM), Gradient Boosting Decision Tree (GBDT), and Deep Neural Network (DNN), which have been widely used in fraud detection tasks. The second type is graph neural networks (GNNs), including:

- GCN [12]: one of the basic GNNs that works as in Eq. 1 and is reimplemented as a random walk-liked GCN in our case to support the inductive inference.
- GraphSAGE [7]: a variant of GCN that samples a fixed-size neighborhood of each node and performs an aggregation over the sampled neighborhood as illustrated in Eq. 2.
- GAT [33]: a GNN that uses multi-headed self-attention to calculate the coefficient of each neighbor node based on the assumption that the importance of different neighbor nodes varies.

As for the third type of baselines, we implement two graph-based fraud detection approaches: Behavior language processing (BLP) and DeepTrax (DTX).

- BLP [23]: a method that constructs an offline bipartite graph with a homophily test and then extracts graph-based features like cluster coefficient for subsequent classifiers to detect online-lending fraud.
- DTX [3]: an approach proposed by Capital One for credit fraud detection, which poses sequences of financial transactions as a bipartite graph and applies simplified two-hop DeepWalk [28] on it.

Implementation details. To meet the requirements of real-world application, we build BN, extract computation subgraph and generate feature for each applicant and its application twenty-four hours after its initial order time (to simulate the duration before the audit in Jimi Store). The training and testing sets are randomly divided into 80/20 based on UID.

For all GNNs, we set the layer number k to 2 and the number of hidden units to 128 and 64 cascaded by a MLP with 32 hidden units. For HAG, the number of units of attention layers is set to 64. Adam optimizer is adopted, and the learning rate of 5e-4 is set. Batch size of 256 among 128, 256, 512 gives the best result for all GNNs. For other baselines, the grid search strategy is applied to find the optimal hyper-parameters. DNN is a three-layer MLP with 128, 64 and 32 hidden units. LightGBM is adopted as the binary classifier for BLP. Classification results conducted by GBDT on embedding generated by DTX and concatenation of the embedding and original features are denoted as DTX_1 and DTX_2, respectively. In the offline evaluation, the classification threshold is set to 0.5.

TABLE III
PERFORMANCE COMPARISON ON $D_1(\%)$. G-SAGE REPRESENTS GRAPHSAGE.

Methods	Precision	Recall	F_1	F_2	AUC	Variance
LR	89.59	41.45	56.68	46.44	69.39	0.10
SVM	**100.0**	33.36	50.33	38.78	68.61	0.15
GBDT	83.33	65.45	73.32	68.38	77.86	0.27
NN	78.95	54.55	64.52	58.14	72.37	0.53
GCN	74.57	69.03	71.69	70.07	77.10	0.33
G-SAGE	79.02	72.78	75.77	73.95	81.77	0.42
GAT	79.17	69.09	73.79	70.90	79.36	1.26
BLP	84.62	67.82	75.29	70.62	78.59	0.37
DTX_1	36.91	47.21	41.43	44.71	37.30	0.69
DTX_2	83.77	68.00	75.07	70.66	78.92	0.69
HAG	81.27	**74.82**	**77.91**	**76.03**	**83.13**	0.61

TABLE IV
PERFORMANCE COMPARISON ON $D_2(\%)$

Methods	Precision	Recall	F_1	F_2	AUC
G-SAGE	93.17	96.09	94.61	96.66	97.31
HAG	95.88	97.46	95.50	97.14	98.28

B. Performance Comparison (Q1)

First, we benchmark HAG against state-of-the-art methods on dataset D_1, concentrating on the AUC of the task. Table III illustrates the average results of ten methods on the testing set, where AUC is the area under the ROC curve and Variance denotes the variance of the AUC in multiple rounds of the same experiment. Two F-measures (F_1, F_2) are included to measure the weighted harmonic mean of the precision and recall of the test, with F_2 assigning recall twice the weight of precision.

From the table, our first observation is that (1) comparing to GNNs, approaches using handcrafted features get higher precision (an average +13.4%), but much lower recall (an average -30.7%). We conjecture the reason is that GNNs potentially cluster nodes in the hidden space by aggregation operation, and thus discover more fraud nodes at the cost of losing part of the accuracy; (2) graph-based methods (BLP and DTX_2) score higher F_1 (+15.3% on average) than another two types of models with no loss of original information but an improvement in recall. The comparison between DTX_1 and DTX_2 further shows the importance of original information in classification; (3) The third observation is that our method consistently outperforms existing methods. Though GraphSAGE shows the best performance compared to other competitors in terms of AUC, HAG still exceeds it by 1.66%, which indicates that HAG can further improve the capability of GNN by the self-aware aggregation and cross-type fusion mechanisms.

Comparison on a larger dataset. Since D_1 already spans 1.5 years and covers most of Jimi Store's labeled data, it is difficult for us to collect another labeled dataset much larger than D_1, or to find a public dataset that fits our scenario. Therefore, we further test the efficiency of HAG on D_2. Though the data distribution of D_2 varies from the real environment (i.e., Turbo is deployed behind Jimi's original risk management system), we can still use it as a benchmark

TABLE V
EFFECT OF SAO AND CFO (%)

Operator	Precision	Recall	F_1	F_2	AUC
SAO(-)	80.11	72.64	76.19	74.02	82.37
CFO(-)	80.65	73.06	76.67	74.46	82.72
Both(-)	79.38	71.88	75.44	73.26	81.93
HAG	81.27	74.82	77.91	76.03	83.13

dataset to train a single anti-fraud model that integrates the function of Jimi's risk management system and Turbo. In offline evaluation, we select GraphSAGE [7] as the competitor, since it shows the best performance on D_1 among other methods. The experimental settings are kept the same and the results are presented in Table IV, where HAG shows superior performance compared to GraphSAGE (+1.00% AUC and +0.94% F_1), which is consistent with the conclusions drawn on D_1.

C. Model Effectiveness (Q2)

Secondly, we study how the proposed SAO and CFO contribute to the expressive power of HAG. To answer Q2, we remove these elements in turn and evaluate the performance of ablated models.

Results are presented in Table V, where we use (-) to denote the removed operator. Specifically, SAO(-) means removing α_{self} and α_{neigh} in Eq. 5 but keeping CFO; CFO(-) means only applying SAO on BN without distinguishing edge types. From the table, we can see that the performance drops when either SAO or CFO is removed (-1.90% of F_1, -2.35% of F_2 and -1.19% of AUC on average). Removing both of them impacts the performance more significantly, suggesting that the two operators work well together to generate more expressive node representations. To testify SAO's effectiveness, we observe that CFO(-) has an average 3.58% improvement of AUC over GNNs, especially 7.29% over GCN and 1.16% over GraphSAGE, meaning SAO not only mitigates the over-smoothing problem but also keeps a better balance between one's contextual information and original feature. SAO(-) also works well and gains a 1.63% and 3.36% improvement of F_1 over Both(-) and other homogeneous GNNs, demonstrating the effect of extra information brought by modeling the heterogeneity of BN.

Ablation Study. We do the ablation analysis to study the contribution of each type of edge on the fraud detection task by masking the edges of a certain type. The percentage drops on AUC for removing different types of edges are presented in Figure 7, where Device ID is the most important relation type in identifying fraud users, as AUC drops the most (6.24%). Furthermore, we refer Device ID, IMEI, IMSI as deterministic types because they convey more certain relations, e.g., two people share the same device must be related to each other, while IP address, GPS, GPS_{Dev}, Wi-Fi MAC, and workplace are considered as probabilistic types as the user nodes connected by edges of these types are not necessarily associated. As shown in the figure, most of the deterministic types have more contributions than the probabilistic types,

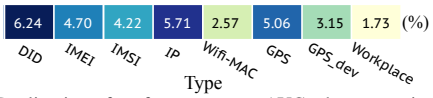
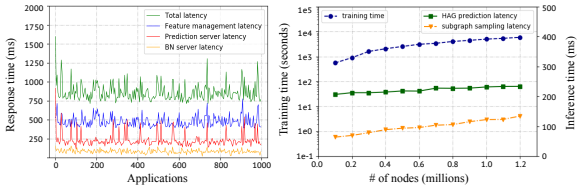


Fig. 7. Declination of performance w.r.t AUC when removing a certain type of edges in turn.



(a) Response time test (b) Scalability test

Fig. 8. The study of time efficiency. (a) presents the response time of the three online modules of Turbo. (b) presents the scalability test of graph computing operations.

which is aligned with our intuition that deterministic relations are stronger than the probabilistic ones.

D. Study of Response Time (Q3)

In this section, we evaluate Turbo's efficiency in the real-world application by testing the time cost of three modules involved in the real-time prediction part through 1,000 applications starting from Jan. 1, 2019. The response time (in milliseconds) is presented in Figure 8a, where the yellow, blue and red lines denote the response time of the BN server (subgraph sampling), feature management module and the prediction server (HAG prediction), respectively. Empirical evidence shows that the latency of feature engineering is around 500ms. The average latencies of the subgraph sampling and the real-time prediction are 87 ms and 230 ms, respectively, which means the overall time cost (green line) is less than 1 second and thus suitable for practical application. In addition, we conduct a scalability study to test the computational overhead of graph computing operations. As presented in Figure 8b, the training time cost on the entire BN (blue line) has a linear growth rate as BN scales up. The latency for each subgraph sampling (yellow line) and prediction request (green line) has a slow growth rate, indicating that Turbo is scalable enough to be adopted in practice.

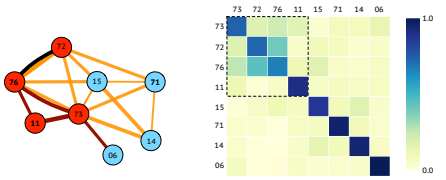
E. Application and Case Study

Although Jimi Store faces tremendous application volumes and involves million yuan per day before Turbo was deployed, **block-listing** and **rule-based scorecards** were still the major anti-fraud approaches used by the platform. Hitherto, Turbo has been deployed and running in Jimi Store since Jul. 2019. After integrated into Jimi's risk management system, Turbo makes real-time prediction for each application passed through the front risk control methods and will block it if its fraud probability exceeds the predetermined threshold. To strike a balance between reducing the fraud ratio and ensuring normal applications are not being blocked, a relatively high threshold should be dynamically preset based on experts' long-time observation of the prediction results made by Turbo.

To demonstrate the online effectiveness of Turbo, we conducted the online A/B test on the live applications spanning

from Jul. 1, 2019 to Jul. 30, 2019. Turbo kicks in after the new applications pass through Jimi's original risk management system. Thus the *test group* is Turbo plus the original risk management system, while the *baseline group* is the original risk management system. The threshold of fraud probability is set to 0.85 in Turbo and any application with a fraud probability above this threshold is considered as fraud. To also examine Turbo's precision and recall, we did not block the detected fraud applications. After a long-term monitoring on the applicants' rent payment status, we report the fraud ratio of applications that pass through the original risk management system. Due to privacy reason, we only report the percentages instead of the exact fraud count. The fraud ratio of the *test group* is 23.19% lower than that of the *baseline group*, which means Turbo can prevent an extra significant amount of assets from being defrauded by fraudsters per month. This represents a significant improvement based on Jimi's current risk management capability. Furthermore, we report Turbo's online precision and recall as 92.0% and 42.8%, respectively.

We next present a specific case identified by Turbo to demonstrate its effectiveness in the real-world application. In Figure 9a, four fraud nodes which connect to each other through Device ID (orange) , UID (black) and IP address (brown) are identified by Turbo. For each node, we visualize its influence distribution derived from HAG as a column of the heat map in Figure 9b, where the dotted box outlines the influence scores of fraud nodes. It can be seen that values inside the box are larger than those outsides, which indicates that fraud nodes have a greater tendency to impact each other during the embedding generation process than normal nodes. This discovery is consistent with previous observations exhibited by fraudsters and shows that HAG is capable of capturing the influence between fraud nodes.



(a) A subgraph case (b) Influence distribution of HAG

Fig. 9. A visualization of influence distributions on a subgraph. (a) is a subgraph in BN, where the color of nodes denotes HAG's classification results (red color represents the fraudsters). (b) is a heat map of the influence distributions of nodes in (a). Each column represents the influence distribution for a node.

VII. Related Work

Substantial efforts have been made to prevent or detect e-commerce fraud. Though the characteristics of attacks vary across industries such as finance, insurance, tax and online transactions, existing research methods mainly focus on the following three directions:

Hard-coding methods. These methods generate sophisticated and application-specific rules for identification. In this category, credit scorecards [1], [32] are widely used to obtain a score result that reflects experience of past cases for each "cardholder". When building a scorecard, it is crucial to define the difference between normal users and fraudsters, which requires extensive manual investigation and is sensitive to application scenarios. Another branch of hard-coding approaches is block-listing, which has been effectively employed in search engine and risk management systems [8], [11]. However, at least one malicious behavior or transaction has to be observed before the security mechanism can block-list them.

Machine-learning methods. Due to the limitation of hard-coding methods, more flexible supervised machine-learning approaches [18], [21], [26] have been applied to learn a decision model by exploiting a mass of history samples. In online payment scenario, Maes et al. [21] achieved fairly good results by applying Neural Networks and Bayesian Belief Networks to detect fraud transactions; Patidar et al. [26] advanced the usage of a neural network with a genetic algorithm to decide the optimal network topology. Weng et al. [34] identified a group of platform-independent features from the product reviews to discriminate fraud and normal items on different e-commerce platforms. Li et al. [18] utilized the reinforcement learning architecture to adjust the impression regulation strategy under different reward settings, which improves the capability of the search engine to combat fraud sellers.

The limitation of supervised algorithms lies in their poor generalization abilities to the unknown fraud types and patterns [22], [30]. In this case, unsupervised or semi-supervised methods [2], [30], [35] are preferred without necessitating domain expertise, rules, patterns, or pre-understanding about the datasets. In e-commece platforms like Alibaba, Weng et al. [35] proposed a time series-based method TSD that models the incoming traffic of user click logs with Poisson distribution to detect abnormal promoted products. Abraham et al. [2] assumed that the normal and anomalies are both generated from Gaussian distributions, while the anomalies have a larger variance. In finance, a geometry-based extraction method named Diffusion Maps [6] is widely used to obtain a faithful low-dimensional representation of the data, and additional anomaly detection methods can be applied to the embedded space.

Graph-based methods. Recently, there is an increasing number of fraud detection works that focus on graph-based methods. The intuition behind is that fraudsters cannot fake interactions with other entities easily, and the homophilic effect exhibited by graphs makes propagation algorithms [9], [17], [35] and GNNs [19], [20] widely used in fraud detection tasks. Bipartite graph propagation algorithms [9], [17] are proposed to detect frauds in the search engine and mobile advertising, etc. In e-commerce, Weng et al. [35] annotated abnormal promoted products as the fraud seeds and applied a weighted HITS algorithm on a user-item bipartite graph to find more fraud products. In the online-lending scenario, Min et al. [23] constructed an application-information bipartite graph and extrated graph-based features like quadrangles and cluster coefficient for online-lending fraud detection. Liang et al. [19] constructed a device-sharing network among customers and employs a GNN to separate fraudsters from regular ones. Liu

et al. [20] aggregated device and activity features jointly in a heterogeneous graph and build a supervised classifier for identifying malicious accounts.

VIII. Conclusion

In this paper, we study the fraud detection problem in deposit-free leasing service. By leveraging the abundant user behavior logs, we present an efficient graph-based anti-fraud system, Turbo, which captures the implicit relations between users with a time-evolving heterogeneous network called BN, and encodes the topology of each node into an expressive representation vector via HAG algorithm to support fraud detection. When evaluated on real-world datasets, the proposed method achieves significantly better results than other competitors. In addition, we deploy Turbo on Jimi Store, which is the first and largest smart-device online leasing platform in China. Through online evaluation, we demonstrate that Turbo is also very effective and scalable in practical scenarios. Our study is expected to shed light on defending against fraud behaviors in the deposit-free leasing industry.

Acknowledgment

This work was partly supported by NSFC under No. 61772466, U1936215, and U1836202, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, and the Fundamental Research Funds for the Central Universities (Zhejiang University NGICS Platform).

References

[1] R. J. Bolton, D. J. Hand, et al. Unsupervised profiling methods for fraud detection. *Credit scoring and credit control VII*, 2001.

[2] A. G. E. P. Box. Bayesian analysis of some outlier problems in time series. *Biometrika*, 1979.

[3] C. B. Bruss, A. Khazane, J. Rider, R. Serpe, A. Gogoglou, and K. E. Hines. Deeptrax: Embedding graphs of financial transactions. *arXiv preprint arXiv:1907.07225*, 2019.

[4] J. L. Carlson. *Redis in action*. Manning Publications Co., 2013.

[5] Y. Chi, G. Dai, Y. Wang, G. Sun, G. Li, and H. Yang. Nxgraph: An efficient graph processing system on a single machine. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016.

[6] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1), 2006.

[7] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[8] D. Hoffstadt, E. Rathgeb, and Liebig. A comprehensive framework for detecting and preventing voip fraud and misuse. In *Computing, Networking and Communications (ICNC), 2014 International Conference*. IEEE, 2014.

[9] J. Hu, J. Liang, and S. Dong. ibgp: A bipartite graph propagation approach for mobile advertising fraud detection. *Mob. Inf. Syst.*, 2017.

[10] G. Huang, X. Cheng, J. Wang, Y. Wang, D. He, T. Zhang, F. Li, S. Wang, W. Cao, and Q. Li. X-engine: An optimized storage engine for large-scale e-commerce transaction processing. In *Proceedings of the 2019 International Conference on Management of Data*, 2019.

[11] N. Jiang, Y. Jin, A. Skudlark, W.-L. Hsu, G. Jacobson, S. Prakasam, and Z.-L. Zhang. Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012.

[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] R. Klinkenberg. Learning drifting concepts with partial user feedback. *Beiträge zum Treffen der GI-Fachgruppe*, 1999.

[14] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*. JMLR. org, 2017.

[15] G. Koutrika and Y. Ioannidis. A unified user profile framework for query disambiguation and personalization. In *Proceedings of workshop on new technologies for personalized information access*, 2005.

[16] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[17] X. Li, M. Zhang, Y. Liu, S. Ma, Y. Jin, and L. Ru. Search engine click spam detection based on bipartite graph propagation. In B. Carterette, F. Diaz, C. Castillo, and D. Metzler, editors, *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014*. ACM, 2014.

[18] Z. Li, J. Song, S. Hu, S. Ruan, L. Zhang, Z. Hu, and J. Gao. Fair: Fraud aware impression regulation system in large-scale real-time e-commerce search platform. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019.

[19] C. Liang, Z. Liu, B. Liu, J. Zhou, and X. Li. Who stole the postage? fraud detection in return-freight insurance claims. 2018.

[20] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2077–2085. ACM, 2018.

[21] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick. Credit card fraud detection using bayesian and neural networks. In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, 2002.

[22] G. Markus, U. Seiichi, and Z. Dongxiao. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *Plos One*, 2016.

[23] W. Min, Z. Tang, M. Zhu, Y. Dai, Y. Wei, and R. Zhang. Behavior language processing with graph based feature generation for fraud detection in online lending. 2018.

[24] A. C. of China E-government Network. Annual report on china's sharing economy development (2019), 2019. http://www.sic.gov.cn/archiver/SIC/UpFile/Files/Default/20190301115908284438.pdf.

[25] C. A. of Information and T. Communications Technology. Digital finance anti-fraud: Insights and raiders, 2018. http://www.caict.ac.cn/kxyj/qwfb/bps/201811/P020181127615657923423.pdf.

[26] R. Patidar, L. Sharma, et al. Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(32-38), 2011.

[27] D. Paul, Y. Peng, and F. Li. Bursty event detection throughout histories. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019.

[28] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD*, 2014.

[29] T. Pham, T. Tran, D. Phung, and S. Venkatesh. Column networks for collective classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[30] G. Shabat, D. Segev, and A. Averbuch. Uncovering unknown unknowns in financial services big data by unsupervised methodologies: Present and future trends. In A. Anandakrishnan, S. Kumar, A. R. Statnikov, T. A. Faruquie, and D. Xu, editors, *Proceedings of the KDD 2017 Workshop on Anomaly Detection in Finance*, 2017.

[31] S. Si, H. Chen, W. Liu, and Y. Yan. Disruptive innovation, business model and sharing economy: the bike-sharing cases in china. *Management Decision*, 2020.

[32] N. Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*. John Wiley & Sons, 2012.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[34] H. Weng, S. Ji, F. Duan, Z. Li, J. Chen, Q. He, and T. Wang. Cats: Cross-platform e-commerce fraud detection. 2019.

[35] H. Weng, Z. Li, S. Ji, C. Chu, H. Lu, T. Du, and Q. He. Online e-commerce fraud: a large-scale detection and analysis. In *ICDE*. IEEE, 2018.

[36] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[37] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018.