Representing Fine-Grained Co-Occurrences for Behavior-Based Fraud Detection in Online Payment Services

Cheng Wang, *Senior Member, IEEE* and Hangyu Zhu

Abstract—The vigorous development of e-commerce breeds cybercrime. Online payment fraud detection, a challenge faced by online service, plays an important role in rapidly evolving e-commerce. Behavior-based methods are recognized as a promising method for online payment fraud detection. However, it is a big challenge to build high-resolution behavioral models by using low-quality behavioral data. In this work, we mainly address this problem from data enhancement for behavioral modeling. We extract fine-grained co-occurrence relationships of transactional attributes by using a knowledge graph. Furthermore, we adopt the heterogeneous network embedding to learn and improve representing comprehensive relationships. Particularly, we explore customized network embedding schemes for different types of behavioral models, such as the population-level models, individual-level models, and generalized-agent-based models. The performance gain of our method is validated by the experiments over the real dataset from a commercial bank. It can help representative behavioral models improve significantly the performance of online banking payment fraud detection. To the best of our knowledge, this is the first work to realize data enhancement for diversified behavior models by implementing network embedding algorithms on attribute-level co-occurrence relationships.

Index Terms—Online payment services, fraud detection, network embedding, user behavioral modeling

1 Introduction

Online payment services have penetrated into people's lives. The increased convenience, though, comes with inherent security risks [1]. The cybercrime involving online payment services often has the characteristics of diversification, specialization, industrialization, concealment, scenario, and cross-region, which makes the security prevention and control of online payment extremely challenging [2]. There is an urgent need for realizing effective and comprehensive online payment fraud detection.

The behavior-based method is recognized as an effective paradigm for online payment fraud detection [3]. Generally, its advantages can be summarized as follows: First, behavior-based methods adopt the non-intrusion detection scheme to guarantee the user experience without user operation in the implementation process. Second, it changes the fraud detection pattern from one-time to continuous and can verify each transaction. Third, even if the fraudster imitates the daily operation habits of the victim, the fraudster must deviate from the user behavior to gain the benefit of the victim. The deviation can be detected by behavior-based methods. Finally, this behavior-based method can be used cooperatively as a second security line, rather than replacing with other types of detection methods.

However, the effectiveness of behavior-based methods often depends heavily on the sufficiency of user behavioral data [4]. As a matter of fact, user behavioral data that can be used for online payment fraud detection are often low-quality or restricted due to the difficulty of data collection and user privacy requirements [5]. In a word, the main challenge here is to build a high-performance behavioral model by using low-quality behavioral data. Then, this challenging problem can naturally be solved in two ways: *data enhancement* and *model enhancement*.

For behavioral model enhancement, a widely recognized way is to build models from different aspects and integrate them appropriately. For model classifications, one type is based on the behavioral agent since it is a critical factor of behavioral models. According to the granularity of agents, behavioral models can be further divided into the *individual-level models* [6], [7], [8], [9] and *population-level models* [10], [11], [12], [13].

In this work, we focus on the other way, i.e., behavioral data enhancement. As for this way, a basic principle is to deeply explore relationships underlying the transaction data. The more fine-grained correlations can possibly provide richer semantic information for generating high- performance behavioral models. Existing studies in data enhancement for behavioral modeling mainly focus on mining and modelling the correlations (including co-occurrences) between behavioral features and labels [14]. To further improve data enhancement, a natural idea is to investigate and utilize the more fine-grained

• *The authors are with the Department of Computer Science and Engineering, the Key Laboratory of Embedded System and Service Computing, Ministry of Education, and the Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 200092, China. E-mail: {cwang, 1830823}@tongji.edu.cn.*

Manuscript received 8 July 2019; revised 17 Feb. 2020; accepted 27 Apr. 2020. Date of publication 4 May 2020; date of current version 17 Jan. 2022. (Corresponding author: Cheng Wang.) Recommended for acceptance by T. Yu. Digital Object Identifier no. 10.1109/TDSC.2020.2991872

correlations in behavioral data, e.g., ones among behavioral attributes.

As the main contribution of our work, we aim to effectively model the co-occurrences among transactional attributes for high-performance behavioral models. For this purpose, we propose to adopt the heterogeneous relation network, a special form of the knowledge graph [15], to represent the co-occurrences effectively. Here, a network node (or say an *entity*) corresponds to an attribute value in transactions, and an edge corresponds to a heterogeneous association between different attribute values. Although the relation network can express the data more appropriately, it cannot finally solve the data imperfection problem for behavioral modeling, that is, it has no effect on enhancing the original low-quality data.

An effective data representation preserving these comprehensive relationships can act as an important mean of relational data enhancement. To this end, we introduce network representation learning (NRL), which effectively capture deep relationships [16]. Deep relationships make up for low-quality data in fraud detection and improve the performance of fraud detection models. By calculating the similarity between embedding vectors, more potential relationships could be inferred. It partly solves the data imperfection problem. In addition to data enhancement, NRL transforms the traditional network analysis from the artificially defined feature to the automatic learned feature, which extracts deep relationships from numerous transactions.

The final performance of behavioral modeling for online fraud detection directly depends on the harmonious cooperation of data enhancement and model enhancement. Different types of behavioral models need matching network embedding schemes to achieve excellent performance. This is one of the significant technical problems in our work. We aim to investigate the appropriate network embedding schemes for population-level models, individual-level models, and models with different generalized behavioral agents. More specifically, for population-level models, we design a label-free heterogeneous network to reconstruct online transactions and then feed the features generated in embedding space into the state-of-the-art classifiers based on machine learning to predict fraud risks; while, for individual-level models, we turn to a label-aware heterogeneous network that distinguishes the relations between attributes of fraudulent transaction, and further design multiple naive individual-level models that match the representations generated from the label-aware network. Furthermore, we combine the population-level and individual-level models to realize the complementary effects by overcoming each other's weaknesses.

The main contributions can be summarized as follows:

- We propose a novel effective data enhancement scheme for behavioral modeling by representing and mining more fine-grained attribute-level co-occurrences. We adopt the heterogeneous relation networks to represent the attribute-level co-occurrences, and extract those relationships by heterogeneous network embedding algorithms in depth.
- We devise a unified interface between network embedding algorithms and behavioral models by customizing the preserved relationship networks according to the classification of behavioral models.
- We implement the proposed methods on a real-world online banking payment service scenario. It is validated that our methods significantly outperform the state-of-the-art classifiers in terms of a set of representative metrics in online fraud detection.

The rest of this paper is organized as follows. We provide a literature review in Section 2. Section 3 gives an overview of our solution. Then, we present our method in detail in Section 4 and make the validation in Section 5. Finally, we conclude the paper and envisage future work in Section 6.

2 Related Work

With the rapid development of online payment service, fraud in online transactions is emerging in an endless stream. Detecting fraud by behavioral models has become a widely studied area and attracted many researchers' attention.

2.1 Composite Behavioral Modeling

In this section, we briefly review different behavior-based fraud detection methods according to the types of behavioral agents [5], [17], [18].

2.1.1 Individual-Level Model

Many researchers concentrated on individual-level behavioral models to detect abnormal behavior which is quite different from individual historical behavior. These works paid attention to user behavior which was almost impossible to forge at the terminal, or focused on user online business behavior which had some different behavioral patterns from normal ones.

Vedran *et al.* [19] explored the complex interaction between social and geospatial behavior and demonstrated that social behavior could be predicted with high precision. Yin *et al.* [4] proposed a probabilistic generative model combining use spatiotemporal data and semantic information to predict user behavior. Naini *et al.* [7] studied the task of identifying the users by matching the histograms of their data in the anonymous dataset with the histograms from the original dataset. Egele *et al.* [8] proposed a behavior-based method to identify compromises of high-profile accounts. Ruan *et al.* [3] conducted a study on online user behavior by collecting and analyzing user clickstreams of a well known OSN. Rzecki *et al.* [20] designed a data acquisition system to analyze the execution of single-finger gestures on a mobile device screen and indicated the best classification method for person recognition based on proposed surveys. Alzubaidi *et al.* [9] investigated the representative methods for user authentication on smartphone devices in smartphone authentication including seven types of behavioral biometrics, which are handwaving, gait, touchscreen, keystroke, voice, signature and general profiling.

2.1.2 Population-Level Model

These works mainly detected anomalous behaviors at the population-level that are strongly different from other behaviors, while they did not consider that the individual-level coherence of user behavioral patterns can be utilized to detect online identity thieves. Mazzawi *et al.* [10] presented a novel approach for detecting malicious user activity in databases by checking user's self-consistency and global-consistency. Lee and Kim [21] proposed a suspicious URL detection system to recognize user anomalous behaviors on Twitter. Cao *et al.* [11] designed and implemented a malicious account detection system for detecting both fake and compromised real user accounts. Zhou *et al.* [12] proposed an FRUI algorithm to match users among multiple OSNs. Stringhini *et al.* [22] designed a system named EVILCOHORT, which can detect malicious accounts on any online service with the mapping between an online account and an IP address. Meng *et al.* [23] presented a static sentence-level attention model for text-based speaker change detection by formulating it as a matching problem of utterances before and after a certain decision point. Rawat *et al.* [24] proposed three methodologies to cope up with suspicious and anomalous activities, such as continuous creation of fake user accounts, hacking of accounts and other illegitimate acts in social networks. VanDam *et al.* [25] focused on studying compromised accounts in Twitter to understand who were hackers, what type of content did hackers tweet, and what features could help distinguish between compromised tweets and normal tweets. They also showed that extra meta-information could help improve the detection of compromised accounts.

2.2 Customized Data Enhancement

To enhance the representation of data in behavioral models, the researchers have focused on the deep relationships under the data. In the following, we summarize the related literature on previous researches.

Zhao *et al.* [26] proposed a semi-supervised network embedding model by adopting graph convolutional network that is capable of capturing both local and global structure of protein-protein interactions network even there is no any information associated with each vertex. Li *et al.* [27] incorporated word semantic relations in the latent topic learning by the word embedding method to solve that the Dirichlet Multinomial Mixture model does not have access to background knowledge when modelling short texts. Baqueri *et al.* [28] presented a framework to model residents$\bar{a}$travel and activities outside the study area as part of the complete activity-travel schedule by introducing the external travel to address the distorted travel patterns. Chen *et al.* [29] proposed a collaborative and adversarial network (CAN), which explicitly models the common features between two sentences for enhancing sentence similarity modeling. Catolino *et al.* [30] devised and evaluated the performance of a new change prediction model that further exploit developer-related factors (e.g., number of developers working on a class) as predictors of change-proneness of classes. Liu *et al.* [31] proposed a novel method for disaggregating the coarse-scale values of the group-level features in the nested data to overcome the limitation in terms of their predictive performance, especially the difficulty in identifying potential cross-scale interactions between the local and group-level features when applied to datasets with limited training examples.

3 Overview of Our Solution

We focus on the fraud detection issue in a typical pattern of online payment services, i.e., online Business-to-Customer (B2C) payment transactions. Here, to acquire the victim's money, frauds usually differ from the victim's daily behavior. This is the fundamental assumption of the feasibility of behavior-based fraud detection. Based on this assumption, the research community is committed to designing behavioral models to effectively distinguish the difference in terms of behavioral patterns. The main challenge of this problem is to build a high-quality behavioral model by using low-quality behavioral data. Naturally, from both aspects, there are two corresponding ways to solve this problem: *data enhancement* and *model enhancement*.

In this work, we aim at devising the corresponding data enhancement schemes for the state-of-the-art behavior models that act as the well-recognized approaches of model enhancement [14]. More specifically, to realize data enhancement for behavioral modeling effectively, we adopt the relation graph and heterogeneous network embedding techniques to represent and mine more fine-grained co-occurrences among transactional attributes. Then, based on the enhanced data, the corresponding behavioral models (or enhanced behavioral models) can be adopted to realize fraud detection. Thereout, as illustrated in Fig. 2, the whole flow of the data-driven fraud detection system consists of three main parts: data representation, data enhancement and model data enhancement.

Before describing the detailed methods, we summarize the relevant conceptions and notations in Table 1 as preparations.

3.1 Data Representation

Online payment transaction records are usually relational data that consist of multiple entities representing the attributes in transactions. We employ a relation graph, which express the data more appropriately in online payment services, to reconstruct losslessly transaction record data, including B2C and C2C transactions.

Lossless Native Graph. Every attribute of a transaction is regarded as the *entity*. For each transaction, we establish the relationships between each entity and its identifier, e.g., the *transaction number*. Furthermore, we attach each identifier a *label* to denote whether this transaction is fraudulent or normal. According to the property of transactions, the set of transactions, denoted by $\mathcal{T}$, can be divided into two disjointed subsets, i.e., the normal and fraudulent transaction sets, denoted by $\mathcal{T}_0$ and $\mathcal{T}_1$. Since an entity may appear in different transactions, we use the co-occurrence relationship to further connect the graphs formed by different transactions. Naturally, we call this graph formed by relational data a *native graph*, as illustrated in the left part of Fig. 1.

Note that the data reconstruction by relation graph merely acts as the initialization of our data enhancement scheme, while it has no real effect on solving the insufficiency of behavioral data. The so-called data insufficiency for behavioral modeling means that, for a given behavioral agent, the existing data are not sufficient to reflect the behavior pattern of this agent. For example, when some accounts with low-frequency behavioral records are regarded as the behavioral agents, their existing behavioral data are possibly too sparse to effectively serve as a data basis for behavioral modeling.

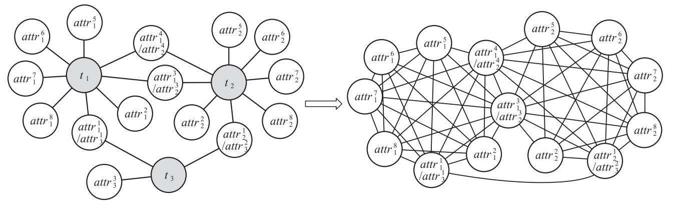


Fig. 1. An exemplary procedure from the native graph (left) to derivative network (right), where B2C transaction contains 8 attributes and C2C transaction contains 3 attributes.

3.2 Data Enhancement

In this work, we utilize network embedding techniques [16] to realize the data enhancement for behavioral modeling. Network embedding is outstanding in solving graph related problems and effectively mines deep relationships. Then, the network structure to be preserved should be determined before a network embedding operation is launched. The network embedding that preserves the network structure of native graph cannot directly help behavioral modeling for online payment fraud detection. The reasons can be summarized as follows:

(1) Under the real-time requirement of online payment fraud detection, it is intolerable to perform network embedding operation for every new transaction due to the response latency lead by large computing overhead. Thus, the uniqueness of transaction number (i.e., identifier) directly destroys the possibility of adopting network embedding online.

(2) There is no need to embed the identifier, say the transaction number, into the vector space, since it's not a valid feature to represent user behavioral patterns. We are interested in the co-occurrence relationships among different behavioral entities rather than the relationship between a unique identifier and its entities.

Therefore, we need generate a new *derivative network* of transaction attributes based on the native graph, preparing for the network embedding.

TABLE 1
Notations of Parameters

Variable	Description
$\mathcal{T}$	The set of transaction history records.
$\mathcal{T}_1$	The set of fraudulent transaction history records.
$\mathcal{T}_0$	The set of normal transaction history records.
$\mathcal{T}^{\mathrm{B}}$	The set of B2C transaction history records.
$\mathcal{T}^{\mathrm{C}}$	The set of C2C transaction history records.
t_i	A transaction with unique identifier i.
$attr_i^j$	A jth attribute of the transaction with unique identifier i.
$\varphi^{\mathrm{P}}(\cdot)$	The representation mapping function about the label-free network.
$\varphi^{\mathrm{I}}(\cdot)$	The representation mapping function about the label-aware network.
$sim(\vec{X},\vec{Y})$	The similarity between vectors $\vec{X}$ and $\vec{Y}$.
$\mathcal{I}_u^a$	The set of all identifiers involving with the agent g_u^a.
$\mathbf{P}_{g_u^a}$	The behavioral model with the agent g_u^a.
$r^a(i)$	The judgment of the a-type agent model on the transaction with unique identifier i.

Customized Derivative Networks. In the data we collected, there are both B2C and C2C transactions. The proportion of frauds in C2C transactions is infinitesimal to that in B2C transactions [32]. Moreover, the mechanism of C2C fraud transactions is essentially different from that of B2C ones [33]. Thus, we limit the scope of this work into online B2C fraudulent transaction detection. We utilize C2C transactions as supplementary (not necessary) information for extracting the relationships among behavioral agents of B2C transactions, i.e., account numbers, from the native graph. Then, we adopt different methods to handle B2C and C2C transactions in the native graph:

(1) For B2C transactions, we define two different vertices, say u and v, that originally connect the same unique identifier as a *vertex pair*, and view it as an *edge* $e=(u,v)$. For example, a B2C transaction with m attributes has $m+1$ vertices and m edges in the native graph, while it correspondingly has m vertices and $m(m-1)/2$ edges on the derivative network.

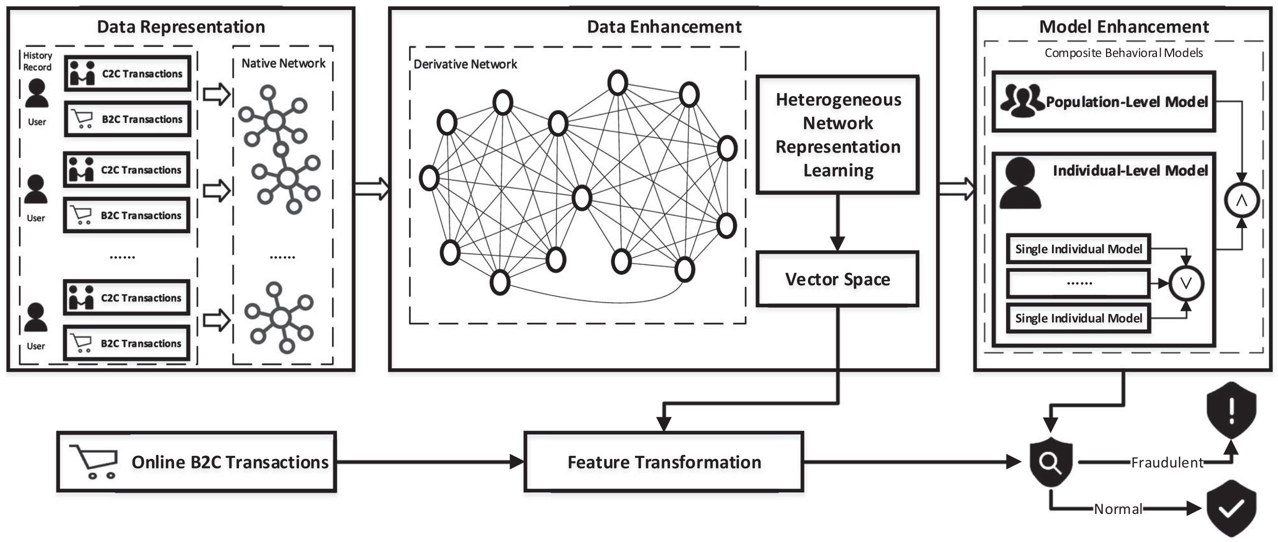


Fig. 2. Workflow of the fraud detection system.

(2) For C2C transactions, we only choose a special attribute pair that has at least one attribute appearing in B2C transaction records as vertex, e.g., the pair of *account number* and *account number*, and use other attributes of their transactions to weight the edges between the special attribute pair. We will analyze the impact of C2C transactions on the model and show the gains from C2C transactions in Section 5.3.1.

We refer to such a denser network generated from the native graph as a *derivative network*. An exemplar illustration is provided in Fig. 1.

The specific structure of derivative networks depends on the data requirements of specific behavioral models. We also have tried to assign different derivative network structure. From the complete graph to the minimum connected graph, we attempt to only consider the node pairs associated with *account_number* as edges. The results turn out to be a great poorer than the complete graph structure, and we analyze that special attribute, like *account_number*, may do not necessarily play a decisive role. So we adopt a complete graph structure including arbitrary node pairs in the derivative network, and computed the similarity between all node pairs.

Heterogeneous Network Embedding. The specific vector spaces corresponding to the derivative networks are learned by heterogeneous network embedding algorithms [34], [35], [36], [37], [38]. For the behavioral models, we obtain the mapping functions from vertices to vectors in specific vector spaces, denoted by $\varphi(\cdot)$. To infer more potential relationships, we calculate the metric $sim(\vec{X},\vec{Y})$ as features for each transaction, where the vectors $\vec{X},\vec{Y}$ stem from $\varphi(\cdot)$.

3.3 Model Enhancement

In this work, we classify user behavioral models into two kinds according to the granularity of behavioral agents, i.e., the *population-level model* [13] and *individual-level model* [6]. Accordingly, we establish the population-level model and individual-level model based on the customized derivative network, respectively:

Population-Level Models. The population-level models identify the fraud by detecting the population-level behavioral anomalies, e.g., behavioral outlier detection [39] and misuse detection [40]. The classifiers based on behavioral data can act as this type of models. For data enhancement for them, we need only data refactoring for classifiers by preserving the co-occurrence frequency of behavioral attributes. To this end, we generate a derivative network where the vertices are transaction attributes and the edges with weights represent the co-occurrence frequency, taking no account of transaction labels. We say such a derivative network is *label-free*. Transaction labels just come into play in the training process of models. By embedding the *label-free* network, we get the mapping relationship $\varphi^{\mathrm{P}}(\cdot)$. Then, we feed the features based on $\varphi^{\mathrm{P}}(\cdot)$ into the machine learning based classifiers [41].

Individual-Level Models. The individual-level models identify the fraud by detecting the behavioral anomalies of individuals. They are regarded as a promising paradigm of fraud detection. The efficacy heavily depends on the sufficiency of behavioral data. To build the individual-level regular/normal behavioral models, we need represent the regularity and normality of transaction behavioral data. Then, we should take the labels into account when generating the derivative network. We extract positive relationships generated from $\mathcal{T}_0$ and negative relationships generated from $\mathcal{T}_1$. The positive relationship enhances the correlation between the agents involved, while the negative relationship weakens the correlation. We say such a derivative network is *label-aware*. By applying the network embedding method to the *label-aware* network, we get the mapping relationship $\varphi^{\mathrm{I}}(\cdot)$. Further, we establish the individual-level models of probability in view of $\varphi^{\mathrm{I}}(\cdot)$ [42].

Composite Behavioral Models. Learning from different aspects model can lead to more reliable results. We adopt a union approach to reconcile the judgments from different individual models to improve reliability [43]. At the population-level and individual-level, we utilize the intersection to integrate judgments. That is, the fraud is determined only if the judgments of both models are fraudulent. Our fraud detection model consists of two levels of models and plays a complementary role.

After employing the composite behavioral models, a coming B2C transaction can be transformed into the high-quality feature based on the learned vectors, and further be predicted as either *fraudulent* or *normal*.

4 Method

4.1 Graph Representation of Transaction Records

First of all, our method needs to represent transactional data in the form of a heterogeneous information network, and applies the attribute vectors to subsequent tasks. These attribute vectors are obtained from heterogeneous network embedding in transactions. Next, we present the process of generating heterogeneous native graph.

Denote a set of transaction history records

$$\mathcal{T} = \mathcal{T}^{\mathrm{B}} \cup \mathcal{T}^{\mathrm{C}},$$

where $\mathcal{T}^{\mathrm{B}}$ and $\mathcal{T}^{\mathrm{C}}$ represent the set of B2C and C2C transaction history records, respectively. Let $t_i \in \mathcal{T}$ denote a transaction, where i is the unique identifier of t_i. Transactions are characterized by a sequence of attributes. We denote the jth attribute of a transaction t_i as $attr_i^j$. Usually, some attributes have consecutive values, we need to discretize these values and then naturally build a native graph based on the unique identifier of transactions. We choose the value of attributes and unique identifiers as vertices on the native graph. The pair of $(i, attr_i^j)$ appearing in a transaction is defined as the edge in the native graph.

In this work, we execute our method on an online banking payment dataset where a B2C transaction contains 8 attributes and a C2C transaction contains 3 attributes. To reconstruct the data losslessly, we build a native graph as illustrated in the left of Fig. 1. Here, an attribute value appears in multiple different transactions, leaving only one vertex in the native graph. Recall that we attach each identifier a label (0 or 1) to divide all transaction history records into two disjointed subsets, i.e., the normal and fraudulent transaction sets, denoted by $\mathcal{T}_0$ and $\mathcal{T}_1$, respectively.

4.2 Network Embedding

4.2.1 Derivative Network

A heterogeneous information network that reflects the impact of transaction labels is what our model needs. We focus on treating the relationships generating from normal transactions or fraudulent transactions unequally in population-level and individual-level models. For the population-level model which learns the difference between normal and fraudulent transactions from all transactions, we only represent the transactions and leave the task of identifying labels to the model. For the individual-level model which establishes user behavioral patterns by normal transactions, it is necessary to embody the label of transaction in the derivative network. For that we set two hyperparameters, β and γ, for fraudulent transactions to distinguish other transactions, and formulate a weight w_e of an edge e as

$$w_e(\beta,\gamma) = \sum_{e\to\mathcal{T}_0} \varepsilon_e + \sum_{e\to\mathcal{T}_1} (-\gamma)^{\beta} \cdot \varepsilon_e, \tag{1}$$

where the operator $\to$ means that a given edge in a derivative network corresponds to the relation between two attributes of the transactions in a specific transaction set; and $\varepsilon_e > 0$ is the *primary weight* of edge e depending on its type: When $e \to \mathcal{T}_0$, the weight of e is equal to $\varepsilon_e > 0$, when $e \to \mathcal{T}_1$, and the weight of e is equal to $(-\gamma)^{\beta} \cdot \varepsilon_e$ with $\gamma \geq 0$ and $\beta = 0$ or 1 cooperatively acting as the adjustment coefficient.

A larger weight of an edge indicates that its two vertices (corresponding to two transaction attributes) are more closely relevant. In this work, we simply divide the edges into two kinds according to whether or not the edges are directly relevant to account numbers. We set the primary weights of the latter kind of edges by a proportion of those of the former kind. For example, we set this proportion to be 0.55, whose adjustment procedure will be introduced later in Section 5.2.1.

We follow two principles in the process of constructing derivative networks. The principle of relationship extraction, as in Equation (1), is that the more co-occurrences in $\mathcal{T}_0$, the greater weights of edges. The other one is to remove corresponding vertices of transactional unique identifier on native graph. For the transactions in $\mathcal{T}^{\mathrm{C}}$, we retain the attributes of transactional account number which appeared in $\mathcal{T}^{\mathrm{B}}$ on the derivative network. For $e \to \mathcal{T}^{\mathrm{C}}$, e merely contains one type (account number, account number), and other attributes are defined as the influence factor of the edge's weight. In the B2C scenario, we retain all other attributes except the unique identifier, and then define two different vertices that connect the same unique identifier in the native graph as a vertex pair, and view it as an edge in derivative networks.

In the above description, we find that the weights of edges generate a marked disproportion due to using the summation in large datasets. For instance, the weight of an edge is small between an account number and a transactional time when there are very few transactions related to the account number in a dataset. But the weight of an edge is tremendous between the transactional type and transactional time because it can appear in transactions with various account numbers. This huge gap is not conducive to reflect real relationships of different vertices in reality. We introduce a mapping function to smooth the gap in the weights of edges, and map the weight w to an interval [0,1], that is

$$S(w_e) = \frac{1}{1+\exp(-\ln(\alpha \times w_e)+\theta)}, \tag{2}$$

where the parameters α and θ are important to change the weight, and control how fast the gap reduces; the parameter α controls the changing degree of weights; the parameter θ also controls the changing degree of weights, but it plays an important role when w is relatively large. We set α to be a low value in order to ensure that the ratio of two edges' weight becomes smaller, and set θ to be a high value for ensuring that the ratio of two edges' weight keeps as constant as possible when w is relatively small.

In the dataset adopted in our work, we set α to 1.8 and θ to 5, whose adjustment procedures will be introduced later in Section 5.2.1. This strategy encourages the gap moderately reduces when the weight w is tremendous and the gap changes as little as possible when it is a small value.

4.2.2 Heterogeneous Network Embedding

Heterogeneous network embedding is a specific kind of network embedding. To transform networks from network structure to vector space, the commonly used models mainly include random walk [34], matrix factorization [16], and deep neural networks [37].

We use a well-recognized heterogeneous network embedding algorithm called HIN2Vec [35] to represent the derivative networks. Compared with other similar algorithms, HIN2Vec distinguishes the different relationships among vertices, and treats them differently by learning the relationship vectors together. Besides, it does not rely on artificially defined meta-paths. The parameter settings in HIN2Vec affect the representation learning and application performance. We explain some main parameters in Table 2, which shows the parameters of our experiments for reference. Note that the settings are related to the size of the input network. A small dimensionality is not sufficient to capture the information embedded in relationships among nodes, but a large value may lead to noises and cause overfitting. A larger network might need a larger dimensionality to capture the information embedded. The number and length of random walks determine the number of sample data, that is, the greater the value, the more the sample data. Generally, the performance continues to improve and converge when the values are large enough. Though a large length of

TABLE 2
Main Parameters

Attribute	Value	Explanation
Dimensionality	128	Dimensionality of node vectors.
Number of random walks	160	Number of random walks starting from each node.
Length of random walks	10	Max length of each random walk.
Length of meta-paths	5	Max window length of context.
Negative sampling rate	5	Number of examples for negative sampling.
Initial learning rate	0.025	Initial learning rate in stochastic gradient descent.

meta-path cannot possibly affect the performance significantly, it is still helpful in capturing high-hop relationships. The negative sampling rate determines the proportion of negative samples in representation learning.

4.3 Fraud Detection Models

4.3.1 Fraud Detection in Population-Level Model

A heterogeneous information network that fully reflects all transactions contains all the edges and vertices that have appeared in the native graph. We treat the co-occurrence relationships generated from $\mathcal{T}_0$ or $\mathcal{T}_1$ equally by setting $\gamma = 1$ and $\beta = 0$, that is, free to transactional label. In the population-level model, we need draw a lesson from fraudulent transactions, and learn the manifestations of fraudulent and normal transactions by advanced classifiers. So we select the label-free network as the input of the heterogeneous network embedding method. Then we get a mapping function $\varphi^{\mathrm{P}}(\cdot)$, which is the vector representation of attributes in transactions. For an attribute $attr_i^j$, $\varphi^{\mathrm{P}}(attr_i^j)$ is the representation in vector space from the label-free network.

In the simplest case, we replace the attribute with a vector representation in a transaction. Then a transaction with m attributes is represented as a matrix of size $d \times m$, where d is the dimension size of the vector representation. But we observe that this solution does not work well and takes up plenty of computing and storage resources. What we need are the features that can summarize a bunch of transactions, so the features should be shared in similar transactions. To this end, we choose to calculate the similarity of any two vector representations as new features based on the above matrix. Specifically, we can get $m(m-1)/2$ similarities to represent a transaction record. The procedure of computing the similarity is formalized as follows: Given a transaction with m attributes and unique identifier i, $\varphi^{\mathrm{P}}(attr_i^1), \varphi^{\mathrm{P}}(attr_i^2), \ldots, \varphi^{\mathrm{P}}(attr_i^m)$, for $\varphi^{\mathrm{P}}(\cdot)$ represents a d-dimensional vector, for $\varphi^{\mathrm{P}}(attr_i^j), \varphi^{\mathrm{P}}(attr_i^k)$ we have

$$sim(\vec{X}, \vec{Y}) = \frac{\sum_{s=1}^{m}(x_s \times y_s)}{\sqrt{\sum_{s=1}^{m} x_s^2} \times \sqrt{\sum_{s=1}^{m} y_s^2}}, \tag{3}$$

by using the Cosine similarity, where $\vec{X}, \vec{Y}$ respectively represent $\varphi^{\mathrm{P}}(attr_i^j), \varphi^{\mathrm{P}}(attr_i^k)$ and x_s, y_s respectively represent the value on the sth dimension of the vector $\vec{X}, \vec{Y}$. The Cosine similarity pays more attention to the difference between two vectors in direction and is not sensitive in numerical value. The population-level model fits well with the Cosine similarity since it focuses on the tendency of most individuals. To better represent a transaction, we also calculate similarities' average and variance. We denote $sim\_avg(i), sim\_var(i)$ as the average and variance of a transaction with unique identifier i. For a transaction without missing values, they are calculated as follows:

$$sim\_avg(i) = \frac{2}{m(m-1)} \sum_{j=1}^{m-1} \sum_{k=i+1}^{m} sim(\varphi^{\mathrm{P}}(attr_i^j), \varphi^{\mathrm{P}}(attr_i^k)),$$

$$sim\_var(i) = \frac{2}{m(m-1)} \sum_{j=1}^{m-1} \sum_{k=i+1}^{m} v(i,j,k), \tag{4}$$

where

$$v(i,j,k) = \left(sim(\varphi^{\mathrm{P}}(attr_i^j), \varphi^{\mathrm{P}}(attr_i^k)) - sim\_avg(i)\right)^2.$$

In reality, a transaction may have some missing values, we also consider its similarity as missing values. When calculating the average and variance, we do not consider the items corresponding to those missing values. In this work, we design the cosine similarity between vectors and their average and variance as new features. All the new features can be quickly calculated, thus ensuring that our model can easily complete feature transformation based on network embedding.

In the real online payment scenario, we divide training samples and testing samples in time order to avoid *time-crossing* problems [44]. Time-crossing means using some information that has not yet occurred when a transaction is tested. We use all the data from the training samples to build a label-free network, and get the mapping function $\varphi^{\mathrm{P}}(\cdot)$ by heterogeneous network embedding. Then we complete feature transformation on all data, training and testing samples, based on mapping function $\varphi^{\mathrm{P}}(\cdot)$. We get the population-level model by fitting training samples on existing classifiers, e.g., XGBoost. For an incoming transaction or testing samples, we input them into the population-level model after feature engineering, and make a discriminant prediction to obtain the probability of fraud in the transaction.

4.3.2 Fraud Detection in Individual-Level Models

In the individual-level model, the derivative network needs to reflect the behavioral distribution of all normal transactions without wasting information brought by fraudulent transactions. Our idea is that the information on normal transactions enhances the association of attribute vertices in the derivative network. On the contrary, the information brought by fraudulent transactions weakens its connection. Therefore, we stipulate that an edge has a positive weight value when it is generated from $\mathcal{T}_0$, and an edge has a negative weight value when the relationship occurs in $\mathcal{T}_1$ by setting $\gamma = 1$ and $\beta = 1$. The strategy effectively utilizes label information, which is also the biggest difference from the label-free network for the population-level models. In some cases, the special relationship number in $\mathcal{T}_1$ is much bigger than ones in $\mathcal{T}_0$, that causes the weight of some edges to become negative or zero. Our solution is to remove these edges in derivative networks. One reason is that these relationships reflected by edges are negligible in the behavioral distribution of all normal transactions we want to get, when the weight of an edge is negative or zero. The other reason is, negative weights cannot be applied to the random walk process of network embedding method we adopt. Similar to the population-level model, we get a mapping function $\varphi^{\mathrm{I}}(\cdot)$, which is the vector representation of attributes in $\mathcal{T}$.

Next, we discuss how to model behavioral models based on network embedding. We denote the agent as the basic unit in models, that is an agent is an individual and all transactions sharing a common agent's value reflect the agent's stable pattern. Taking our online transaction record as an example, the attribute, *account_number*, is a common choice as an agent. Under this agent, transactions are divided into different parts, so that all transactions in each part have the same account number. We can detect anomalies by comparing with behavioral models when we assume that an agent's behavioral pattern is stable. We discuss behavioral models from the perspectives of single-agent and multi-agent, respectively.

Single-Agent Behavioral Model. Similar to feature transformation on the population-level model, we choose to calculate the similarity of any two vector representations based on a size of $d \times m$ matrix, which is represented by a transaction with m attributes. Here, d is the dimension size of vector $\varphi^{\mathrm{I}}(\cdot)$. One difference is that similarity is calculated differently. Given vector $\vec{X}$ and $\vec{Y}$, x_s, y_s respectively represent the value on the sth dimension of the vector $\vec{X}$ and $\vec{Y}$. We have

$$sim'(\vec{X},\vec{Y}) = \sqrt{\sum\nolimits_{s=1}^{m}(x_s - y_s)^2}, \tag{5}$$

by using the euclidean distance, which emphasizes the difference in numerical value and therefore appropriates to characterize each individual. For the vectors $\varphi^{\mathrm{I}}(attr_i^j)$ and $\varphi^{\mathrm{I}}(attr_i^k)$, we calculate the similarity $sim'(\varphi^{\mathrm{I}}(attr_i^j), \varphi^{\mathrm{I}}(attr_i^k))$ according to Equation (5). We introduce *cohesivity* to express the importance of a transaction in the behavioral model and denote $C(i)$ as the *cohesivity* of a transaction with unique identifier i. The *cohesivity* $C(i)$ can be computed in the following way:

$$C(i) = \frac{1}{cm_0 + \sum_{j=1}^{m-1}\sum_{k=i+1}^{m} cm_l \times v'(i,j,k)}, \tag{6}$$

where $v'(i,j,k) = sim'(\varphi^{\mathrm{I}}(attr_i^j), \varphi^{\mathrm{I}}(attr_i^k))$ and $l = m(j-1) + k - 1$. The value cm_l represents the lth value in the coefficient matrix

$$\left[cm_0,\ cm_1,\ cm_2,\ \ldots,\ cm_{m(m-1)/2}\right],$$

where cm_l, for $l = 0, 1, 2, \ldots, m(m-1)/2$, can be determined by adopting the method of linear regression. For all samples without missing values, we calculate the similarity as new features according to Equation (5), and then fit them on linear regression to get the coefficient matrix. We can get the regression coefficient and offset corresponding, corresponding to $\left[cm_1,\ cm_2,\ \ldots,\ cm_{m(m-1)/2}\right]$ and cm_0, respectively. Denote an agent as g_u^a, where a is the attribute type corresponding to the agent, and u represents the value of attribute a of the agent. Accordingly, we denote the set of agents refer to a as $\mathcal{G}^a$. Let $\mathcal{I}_u^a$ denote the set of all transactional identifiers involving with g_u^a. Furthermore, we define $\mathcal{I}^a := \bigcup_u \mathcal{I}_u^a$. At this point, we formally denote the behavioral model as follows. For a given agent $g_u^a \in \mathcal{G}^a$, its behavioral model is defined as $\mathbf{P}_{g_u^a}$, which is a discrete probability distribution function reflecting the normal transactional patterns. For every possible transaction identifier i, we have its corresponding probability $p_{g_u^a}(i)$ of occurrence in $\mathbf{P}_{g_u^a}$.

The procedure of computing the corresponding probability $p_{g_u^a}(i)$ is formalized as follows:

$$p_{g_u^a}(i) = \frac{\sigma(C(i))}{\sum_{i' \in \mathcal{I}_u^a}\sigma(C(i'))}, \tag{7}$$

where $\sigma(z) = \frac{1}{1+\exp(z)}$ is the sigmoid function. In practice, the size of $\mathcal{I}_u^a$, denoted by $|\mathcal{I}_u^a|$, is dependent on the product of the number of available values for all other attribute types except the attribute of agent a. So our behavioral model is a special case, discrete probability distribution, by calculating the probability of each transaction in fraud detection. We adopt the same method as the population-level model to divide the training samples and test samples, and only use the training samples to build the model.

For some u, $|\mathcal{I}_u^a|$ is often a large value and the computational overhead of probability distribution will be unbearable. We use the clustering algorithm to overcome this problem. For vectors referring to the same attribute type in vector space, the vectors of the same cluster are represented by cluster vectors, that is, similar vectors are treated as one vector, which can quickly reduce the value of $|\mathcal{I}_u^a|$. In this work, we choose the account number as the agent's type. In other words, we establish behavioral models for all account number which appears in a label-aware network. We observe that single-agent models based on account number or other attributes are often hard to achieve an excellent performance due to the absence of agents. An effective way to solve the problem is modelling in multiple agents.

Multi-Agent Behavioral Model. To cope with insufficient or missing historical transactions of the single agent, we prefer to the models under different agents without acquiring more complete and adequate historical transactions. This part describes how we build the behavioral model to detect a transaction better under multiple agents in case of insufficient transactions. Similar to the commonly-used agent, i.e., account number, some other attributes, e.g., merchant number and location number, can also act as the agents to build behavioral models. Note that the value space of attribute types that can act as agents should not be too small. That will lead to a lack of advantage for the individual-level behavioral model. Let $\mathcal{A}$ denote a set of attribute types that can act as agents. For each attribute in $\mathcal{A}$, we repeatedly model the single-agent behavioral model and then add those models to the final set $\mathcal{F}$. We can detect the fraud probability of a transaction under different agents with $\mathcal{F}$. The procedure of building multi-agent behavioral models is described in Algorithm 1.

Algorithm 1. Building Multi-Agent Behavioral Models

```
Input: The set of attribute types $\mathcal{A}$
Output: The set of multi-agent behavioral models $\mathcal{F}$
1  Initialize $\mathcal{F}$;
2  foreach $a \in \mathcal{A}$ do
3      foreach $g_u^a \in \mathcal{G}^a$ do
4          Initialize $\mathbf{P}_{g_u^a}$;
5          foreach $i \in \mathcal{I}_u^a$ do
6              Compute $C(i)$ using Equation (6);
7              Compute $p_{g_u^a}(i)$ using Equation (7);
8              Add $i, p_{g_u^a}(i)$ into $\mathbf{P}_{g_u^a}$;
9          end
10         Add $\mathbf{P}_{g_u^a}$ into $\mathcal{F}$;
11     end
12 end
13 Return the set of multi-agent behavioral models $\mathcal{F}$
```

We define the fraud detection problem in individual-level behavioral models as follows: Given a transaction, its fraud score rated by its corresponding probability in the single-agent behavioral model determines whether the transaction is fraud or not. This may include the following scenarios: (1) the transaction provides complete information; (2) the transaction miss values in some attributes. For the former, we can directly get its probability in behavioral models. Since all attributes are required to calculate the fraud score of the transaction in the behavior model, it is difficult to judge the transaction with missing values. So in our model, we compute the average possibility of all transactions, which are related to existing attributes of the transaction with identifier i, as the probability $p_{g_u^a}(i)$, and define the set of these transaction identifiers as $\mathcal{I}_i^{'}$. Then we get the behavioral model $\mathbf{P}_{g_u^a}$ corresponding to the agent $p_{g_u^a}(i)$, and denote the domain of $\mathbf{P}_{g_u^a}$ as $\mathcal{P}_{g_u^a}$. For a transaction identifier i, we get a new distribution $\mathbf{P}_{g_u^a}^{'}$ by removing the $\mathcal{I}_i^{'}$ from the domain of $\mathbf{P}_{g_u^a}$, and denote the domain of $\mathbf{P}_{g_u^a}^{'}$ as $\mathcal{P}_{g_u^a}^{'}$. Next, we calculate its score $score_{g_u^a}(i)$ as described in Equation (8):

$$score_{g_u^a}(i) = \frac{p_{g_u^a}(i) \times \exp(-H_{g_u^a})}{N_0 + \frac{1}{|\mathcal{P}_{g_u^a}^{'}|} \times \sum_{i' \in \mathcal{P}_{g_u^a}^{'}} p_{g_u^a}(i')}, \tag{8}$$

where

$$H_{g_u^a} = -\sum_{i \in \mathcal{P}_{g_u^a}} p_{g_u^a}(i) \times \log{}_2 p_{g_u^a}(i), \tag{9}$$

$|\mathcal{P}_{g_u^a}^{'}|$ is the cardinality of $\mathcal{P}_{g_u^a}^{'}$, N_0 is responsible for adjusting the influence degree of transactions other than the transaction t_i in the behavioral model on the score. The larger N_0 is, the lower the influence of other transactions on the score. In our work, we set $N_0 = 0$.

We observe that there is a clear distinction between fraudulent and normal transaction scores. For an attribute type $a \in \mathcal{A}$, we set an interval Ω^a and give the judgment result according to Equation (10):

$$r^a(i) = \begin{cases} 1, & score_{g_u^a}(i) \in \Omega^a \\ 0, & score_{g_u^a}(i) \notin \Omega^a \end{cases}. \tag{10}$$

We denote fraudulent transactions by label 1 and normal transactions by label 0. The upper and lower limits of the interval Ω^a depend on the scores distribution of training samples.

4.3.3 Fraud Detection in Composite Models

A single-agent behavior model can only give a certain fraud judgment. The normal judgment may not reliable due to the release of transactions that cannot be checked. In this work, we imitate the one-veto mechanism to synthesize the final results returned by multi-agent models. That is only an agent behavioral model returns a judgment marked as fraud, the final result is marked fraud. This strategy ensures that the multi-agent model is complementary enough to capture as many fraudulent transactions as possible.

So far, we already have two different level ways to identify whether a transaction is fraudulent or not. These two methods identify fraudulent transactions from different perspectives. Population-level models compare the similarity between a transaction and the learned transactional patterns. Individual-level models distinguish a transaction by contrasting the difference between its current and past patterns. We compose these two models to further improve the performance of our methods. The transaction is detected as fraudulent transactions if and only if the result from both models are judged as fraudulent transactions. The consistency of judgment on fraudulent transactions reduces the probability of misjudgment of normal transactions, and ensures that it has better performance than a single-model, i.e., the population model or individual model. For different performance objectives, other combinations can be also tried, which will be reserved for future research. The process of building a fraud detection model is described in Algorithm 2.

Algorithm 2. The Process of Fraud Detection

```
Input: The set of attribute types $\mathcal{A}$, The set of transactional
       identifiers $\mathcal{I}$
Output: The set of judgment results $\mathcal{R}$
1  Initialize $\mathcal{R} = \emptyset$;
2  foreach $i \in \mathcal{I}$ do
3      $r(i) := 0$;
4      foreach $a \in \mathcal{A}$ do
5          get $r^a(i)$ using Equation (10);
6          $r(i) := r(i) \vee r^a(i)$;
7      end
8      get $r^a(i)'$ in Section 4.3.1;
9      $r(i) := r(i) \wedge r^a(i)'$;
10     Add $r(i)$ into $\mathcal{R}$;
11 end
12 Return judgment result $\mathcal{R}$
```

5 Evaluation

To evaluate the performance of the proposed models based on co-occurrence relationships in transactions, we build heterogeneous information networks to represent these relationships, and apply the vectors obtained by heterogeneous network embedding to generate behavioral models. Through the empirical evaluation of real-world transactions, we mainly aim to answer the following three research questions:

RQ1: How do the key parameters affect the performance of our models?

RQ2: How much gain does the data enhancement scheme based on network embedding bring to the population-level, individual-level models?

RQ3: How does the design of enhancement scheme affect the performance of our models?

In what follows, we first introduce the experimental settings, and then answer the above research questions in turn.

5.1 Experiment Settings

5.1.1 Datasets

To validate the performance of proposed models, the evaluation is implemented on a real-world online banking payment transaction dataset from one of the biggest commercial banks in China, which contains three consecutive months of B2C and C2C transaction records. The main statistics of the transactions are summarized in Table 3. We use the dataset of

TABLE 3
The Transaction Information

Month	2017.04	2017.05	2017.06	Total
B2C Normal	1,217,101	1,176,680	1,003,461	3,397,242
B2C Fraudulent	13,271	27,122	24,898	65,291
C2C	166,356	205,614	\	371,970

TABLE 4
The Selected Attributes in B2C Transactions

Attribute	Value	Description
account_number	discrete	Each account_number represents a user's account.
merchant_number	discrete	Each merchant_number represents a merchant in a B2C transaction.
place_number	discrete	Each place_number represents an issuing area of banking cards used for transactions.
time	continuous	The exact time when the transaction occurred.
amount	continuous	The amount of money transferred to the merchant in a B2C transaction.
ip	discrete	Whether a commonly used ip or not in a transaction.
last_result	discrete	Judgment of the last transaction in the relevant account_number.
type	discrete	Each type represents a transaction of different type.

April and May 2017 as the training samples, and set the dataset of June 2017 as the testing samples. We also utilize C2C transactions of April and May 2017, when we build heterogeneous information networks. All B2C transactions are labelled either *positive (fraudulent)* or *negative (normal)*, respectively. The training samples contain 2,393,817 normal transactions and 40,393 fraudulent transactions, and the testing samples contain 1,003,539 normal transactions and 24,898 fraudulent transactions. In the original set of transactions, each transaction is characterized by 64 attributes. However, most of them have sparsely valid values (about 10 to 30 percent on average). We finally choose 8 attributes in all to build our models, which are shown in Table 4. The attributes, *time* and *amount*, have continuous values, so we need the further discretization treatment for these attributes. All C2C transactions are represented by 3 attributes which are shown in Table 5. Note that the attribute *amount* in C2C transactions does not appear in the derivative network, which only has an impact on the weights of incident edges.

We discretize the attribute *time* inspired by [45]. The time of day can be divided into four time intervals. We set four intervals of the hour: [0,3), [6,11), [15,24), and $[3,6) \cup [11,15)$, according to the time distribution in transactions. We further divide the attribute *time* into 8 unique values by distinguishing whether it is a weekday. We make different approaches to discretize the *amount* attribute in B2C and C2C transactions because of the different functions of attribute *amount*. For B2C transactions, we discretize them into four different values according to the following intervals, [0,60), [60,300), [300,3600), and $[3600, +\infty)$. For C2C transactions, we assign them different values, i.e., (1,1.5,2,2.5,3), by the following intervals, [0,100), [100,1000), [1000,5000), [5000,50000), and $[50000, +\infty)$.

We also count the number of nodes with different attributes in the label-aware and label-free networks, which are detailed in Table 6. Note that the difference between the number of nodes with attributes *account_number* and *merchant_number* is caused by the removal of edges and nodes from label-free networks. In addition, we observe that the size of individual models is $2,406 \times 327 \times 8 \times\ 10 \times 2 \times 2 \times 11$ when we choose the attribute *account_number* as the agent. It is commonly too large to calculate. So we cluster 2,406 agents with attribute *merchant_number* and 327 agents with attribute *place_number* into 11 and 5 categories, respectively. Similarly, we cluster 190,268 agents with attribute *account_number* into 5 categories when we choose the attribute *merchant_number* or *place_number* as agents.

TABLE 5
The Selected Attributes in C2C Transactions

Attribute	Value	Description
account1_number	discrete	The account1_number is the initiator of the C2C transaction.
account2_number	discrete	The account2_number is the recipient of the C2C transaction.
amount	continuous	The amount of money transferred to the recipient in a C2C transaction.

TABLE 6
Attribute Details

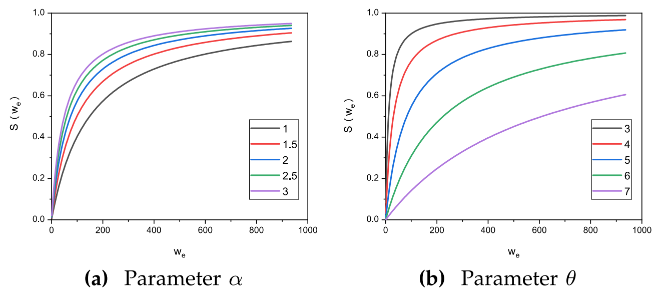
Attribute	Label-Aware Network	Label-Free Network
account_number	221,040	190,268
merchant_number	2,419	2,406
place_number	327	327
time	8	8
amount	10	10
ip	2	2
last_result	2	2
type	11	11

5.1.2 Metrics

To evaluate the performance of our methods, we choose five representative and well-performed techniques as the benchmarks: logistic regression (LR), random forest (RF), naive bayes (NB), XGBoost (XGB), and convolutional neural networks (CNN). Normally, according to the industry requirement, 1 percent is the tolerable upper limit for *FPR (False Positive Rate)*. So an achieved *TPR (True Positive Rate)* with an *FPR* higher than 1 percent makes no sense in this work. We only focus on the meaningful part of the *ROC curve* without considering the whole *AUC* (Area under The *ROC Curve*). In this paper, we use Precision, Recall (TPR), Disturbance (FPR) and F1-score to comprehensively evaluate our methods.

5.2 Parameter Sensitivity

In this set of experiments, we systematically evaluate the parameter sensitivity of our method. Different from the k-fold cross validations, we select the last $1/3$ of the training samples in time sequence as the validation samples, and the other $2/3$ to train the model during the parameters tune. Dividing the verification set in time sequence avoids the time-crossing problem and is more in line with the real application scenario than randomly selecting the verification set.



(a) Parameter α **(b)** Parameter θ

Fig. 3. Parameter tuning in the derivative network with different parameter pairs. We set α and θ at 1.8 and 5, respectively when testing the other.

5.2.1 Network Parameters

Parameter settings in Equation (2) have a significant impact on the weight of edges in derivative networks. In our work, most of the edge weights are less than 1,000, and the larger weights are only about 10,000. So we intend to make the transformation of weights satisfy a set of ratios, where the ratio is calculated by $S(w_e)/S(1)$. The set of ratios satisfy the following rules: When the weight is less than 25, the ratio is close to w_e; when the weight is about 100, the ratio is close to 50; when the weight is very large, beyond 1,000, the ratio is close to 100. To determine parameter settings, we examine such changes in weights. We vary parameters α and θ to determine their impacts on weight changes. Except for the parameters being tested, all other parameters assume default values. We first examine different choices of the parameter α, and choose values of α from 1 to 3. The weight changes under different α are shown in Fig. 3a, which shows that different α slightly change the weight, but the overall trend remains similar. Next, we examine different choices of parameter θ, and choose values of θ from 3 to 7. The weight changes under different θ are shown in Fig. 3a, which shows that different θ dramatically change the weight, especially if it's a huge value. Fig. 3 also shows that the parameter α is positively correlated with $S(w_e)$ and the parameter θ is negatively correlated with $S(w_e)$. Finally, we observe that setting α and θ at 1.8 and 5 respectively is an appropriate choice to reduce the huge gap between different weight values.

From Table 6, we observe that nodes with the attribute *account_number* far outnumber nodes with other attributes. This imbalanced phenomenon leads to an imbalanced network structure. So we further study the average degree of these agents and find that the average degree of nodes with the attribute *merchant_number* is similar to that with most attributes, but is about 90 times than that with the attribute *account_number*. Therefore, we introduce a scheme to balance the network structure. Facing the node with a special attribute, which has q times average degree than the minimum one, we set the weight of edges associated with the special attribute is q^* $(q^* = 1 - q/2 \times 0.01)$ times of the weight corresponding to the minimum average degree. In this work, we set the weights of edges associated with other attributes are 0.55 $(= 1 - 90/2 \times 0.01)$ times of that with the attribute *account_number*.

5.2.2 Embedding Parameters

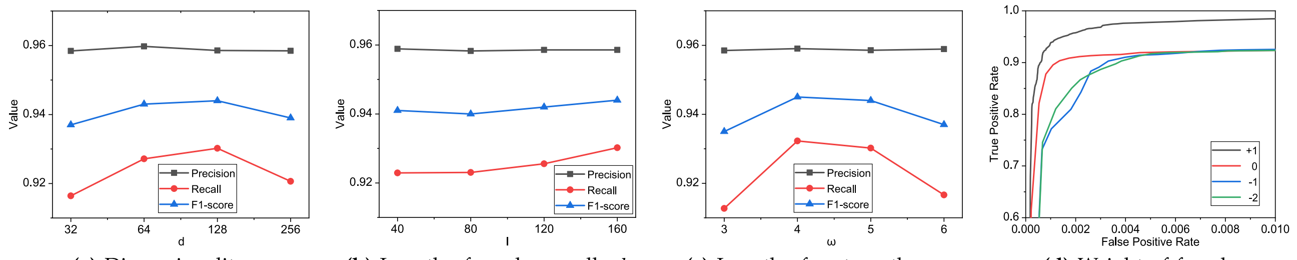
Parameter settings in network embedding methods usually make a difference to the performance of node representation for an application. To tune the appropriate settings, we vary the values of important parameters to observe how the performance changes under population-level models.

Dimensionality of Vector Space. First of all, Fig. 4a shows the impact of setting different numbers of dimension d. Generally, a small d is not sufficient to capture the information embedded in relationships among nodes, but a large d may lead to noises, and cause overfitting. In our work, the best performance is achieved when d is 128. Generally, a larger network might need a larger d to capture the information embedded in relationships between nodes.

Length of Random Walks. A longer random walk can generate more sample data. Fig. 4b manifests that the performance continues to improve when the length of random walks l is increased (then resulting in more sample data), and converges when l is large enough. Meanwhile, the more sample data, the more training time. When l is set as a great value, it brings slight performance growth but dramatic time increase. In our network, we set l to be 160, since it achieves a balance between time-consuming and performance.

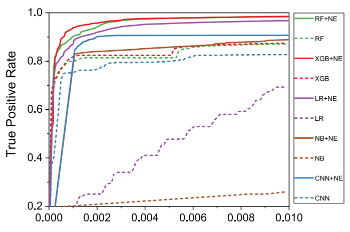
Length of Meta-Paths. Fig. 4c shows that the maximum length of meta-paths has a significant impact on the performance. Capturing meta-paths with larger ω is crucial because some long meta-paths have an important semantic meaning. Note that a large ω will bring in useless semantic information to affect performance. Setting the number of ω to 4 or 5 is a good option in this work.

We verify the performance of different network embedding schemes on the population-level model. Fig. 4d shows the results of our experiment on the XGBoost classifier, and explains why we propose a customized network to deal with labeled transactions. By setting the hyperparameters β and γ, we adjust the ratio of the edge weights of fraudulent transactions to normal transactions in the network at 1, 0,

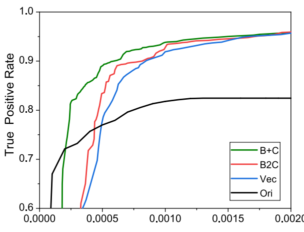


(a) Dimensionality **(b)** Length of random walks l **(c)** Length of meta-paths ω **(d)** Weight of frauds

Fig. 4. Parameter tuning in network embedding with different parameter pairs. Figures (a), (b), and (c), respectively show the model performance of different parameters d, l, and ω, when we set d, l, and ω at 128, 160, and 5, respectively in the process of testing others. Figure (d) shows the influence of different γ and β on the population-level model.



(a) Different models



(b) Different features

Fig. 5. The *ROC curves* of population-level models. Figure (a) shows the performance of different population-level models with or without NE. Figure (b) shows the impacts of different features on population-level models.

-1, -2, respectively. The ratio '1' represents that fraudulent and normal transactions are treated in the same way. That is equivalent to label-free networks. The ratio '0' represents that we only use normal transactions to build the network without fraudulent transactions. The ratios '-1' and '-2' represent the weights of edges generated by fraudulent transactions are -1 or -2 times that of normal transactions, respectively. We observe that the label-free network outperforms the label-aware network in the population-level models from Fig. 4d. We also observe that the ratios '-1' and '-2' have similar performance, which shows that small changes in the ratio have little impact on the model when the ratio is a negative value. In our naive individual-level model, it learns the normal behavioral pattern from the user historical pattern, and cannot exploit fraudulent transactions in the process of building models. The label-aware network integrates the information of fraudulent transactions into the network structure, which is more suitable for individual-level models than label-free networks.

5.3 The Gain of Network Embedding

5.3.1 Performance Gain for Population-Level Models

We compare the performance of five representative classification models described in Section 5.1.2 with those under the help of customized network embedding (NE) schemes. We set the parameters of network embedding in Section 5.2. The *ROC curves* of different classifiers are depicted in Fig. 5a. We observe that the models cooperating with network embedding, i.e., RF+NE, XGB+NE, LR+NE, NB+NE, and CNN+NE, all outperform their counterparts without network embedding. XGBoost gives the best results at different *FPR*, followed by RF, CNN, LR, and NB with network embedding. When the *FPR* is 0.001, XGBoost with network embedding obtains a recall of 93.9 percent, which means that it can prevent about 94 percent of fraudulent transactions when the fraudster begins to act, and just interfere 0.1 percent legal transactions. Random forest performs the second best, just slightly poorer than XGBoost when the *FPR* is small than 0.002. When we decrease the *FPR* to 0.0005, the performances of most methods do not change stupendously despite a partial drop in the *TPR*. It is worth noting that the performances of all models drop dramatically as the *FPR* decreases to 0.0001. The *TPR* of XGBoost is slightly lower than 50 percent. Except for the poor performance of NB and CNN, the others have almost similar recalls when *FPR*$= 0.0001$.

Now we already have a basic understanding of the approximate performance of all candidate classifiers. XGBoost is outstanding in all candidate machine learning models when we set the same *FPR*.

To explore how much gain C2C transactions can bring to our model, we design the following four groups of experiments: (1) 'B+C', both using B2C and C2C transactions on our model; (2) 'B2C', only using B2C transactions on our model; (3) 'Ori', applying original transactions directly to the population-level model; (4) 'Vec', using B2C and C2C transactions to build the label-free network and adopting HIN2Vec method to get the embedding vectors, but detecting fraud by feeding a vector matrix, which consists of representations corresponding to attributes in a transaction, into the population-level model. From Fig. 5b, we find that our model is superior to other comparisons when the *FPR* is less than 0.15 percent. When the *FPR* is greater than 0.15 percent, the gain of our model decreases, and the performance is gradually consistent with other comparisons. Note that the poor performance on 'Vec' explains why we do not use the representation directly but introduce $sim(\vec{X}, \vec{Y})$ for the subsequent tasks. We also observe that the C2C transactions are effectively utilized by our model. When the *FPR* is 0.75 percent, the gain of *TPR* reaches 2.5 percent.

5.3.2 Performance Gain of Individual-Level Models

In this part, we evaluate the performance of the individual-level models in fraud detection with customized network embedding. We present the performance of single-agent behavioral models and discuss the improvements by the multi-agent model compared with the single-agent models. The improvements depend on the following two principles.

The first is the completeness principle of multi-agent models. If a transaction has no historical data under a specific agent, then the single-agent model is impossible to detect the transaction. To give a more straightforward sense, we define a measure called *check rate*, which stands for the proportion of transactions that can be checked with fraud detection techniques under a given agent. The union of subsets of transactions that can be checked by single-agent models should be the complete set of transactions. That is, the *check rate* of the final multi-agent model should be 1.

The second is the preferential principle of single-agent models under the completeness principle. Before integrating different single-agent models into the multi-agent model, we need to evaluate the performance of every single-agent model. If the performance of a single-agent model is too poor, it will harm the performance of the final multi-agent model.

In the implementation of our proposed models, the multi-agent model can apply to all transactions, and the *check rate*s under different single-agent models are shown in Table 7. By calculating the Precision and Recall with different fixed Disturbances, we investigate the performance of proposed single-agent models under the verifiable dataset as presented in Table 7. The Disturbances are fixed as 0.0010, 0.0015, 0.0020, 0.0050, 0.0075, and 0.0100, respectively. It is evident from Table 7 that these single-agent models have a stable and good performance in partial data which can be checked.

When we compare the performance of multiple single-agent models and the multi-agent model, we experiment with all test transactions for all behavioral models. We focus on the performance at different Disturbances between 0.001

TABLE 7
Performance of Single-Agent Models

Attribute/*Check_rate*	*account_number* / 0.92633						*merchant_number* / 0.53844						*place_number* / 0.99997					
Disturbance	0.0010	0.0015	0.0020	0.0050	0.0075	0.0100	0.0010	0.0015	0.0020	0.0050	0.0075	0.0100	0.0010	0.0015	0.0020	0.0050	0.0075	0.0100
Precision	0.81375	0.79173	0.74798	0.54692	0.44728	0.37581	0.96880	0.95994	0.95147	0.90104	0.85672	0.81889	0.93515	0.93433	0.91924	0.82686	0.76180	0.69708
Recall	0.68418	0.91701	0.92648	0.93479	0.93677	0.93825	0.66302	0.76501	0.84403	0.95988	0.96295	0.96485	0.58559	0.86690	0.91517	0.96405	0.96582	0.96658

and 0.0022. From Fig. 6, we can obtain three observations as follows:

First, the performance of single-agent models have a good performance in the partial data, but do not have a stable performance on the whole data. Achieving a good performance in the partial data is a necessary but not sufficient condition for that on the whole data. Then, it is worth considering that the adoption of a multi-agent model by combining multiple complementary single-agent models.

Second, the *check rate* for the single-agent model of *place_number* has a very close to '1', while the single-agent model of *place_number* underperforms the multi-agent model in terms of the Precision and Recall. The reason why the multi-agent model is superior to the single-agent model of *place_number* is that the former combines the advantages of different single-agent models and has a more complete judgment on detection transactions.

Third, we find that the *merchant_number* curve provides the most stable precision and recall, regardless of the disturbance rate. From Table 7, we observe that except for the single-agent model of *merchant_number*, which only has a *check rate* of about 50 percent, the other two single-agent models have a *check rate* of over 90 percent. In a real scenario, the single-agent model of *merchant_number* can not be used alone to implement anti-fraud tasks because of its low *check rate*. In our work, the single-agent model can only detect fraud, but can not ensure that non-fraud is normal. The high performance of the *merchant_number* model comes from its release of nearly half of transactions, so its performance is not representative and credible. In online payment services, the judgment results with high performance and low credibility are not acceptable. By combining the judgment of multiple single-agent models, we can make more accurate judgment results with the same credibility.

5.4 Performance of Enhancement Scheme

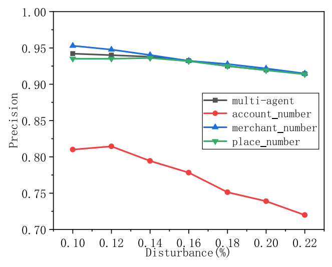
5.4.1 Performance of Data Enhancement

The framework of the proposed data enhancement scheme is compatible with most network embedding methods. We compare the effects of the state-of-art network embedding methods in the population-level model. Besides HIN2Vec, we also investigate the performance of node2vec [46], transE [47] and metapath2vec [48]. For similar parameters, we use the same values as HIN2vec, and we use the default values for the others. Fig. 7a shows the *ROC curves* of different network embedding methods in the population-level model. We observe that all models cooperating with different network embedding methods have a similar performance. HIN2Vec and metapath2Vec have a better performance than node2vec and transE. The lower performance of node2vec mainly stems from its inability to distinguish the types of nodes. In transE, the method focuses on resolving relationships between different entities but taking no account of the weight of relationships. Most embedding methods are feasible as data enhancements, with only slight differences in terms of performance.

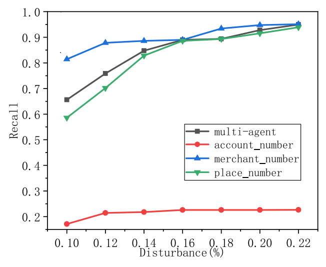
We compare the performance of behavioral models under different similarity calculations to illustrate the need to treat two levels of behavioral models differently. Fig. 8 shows the advantages of proper similarity calculation. From Fig. 8a, we observe that the population-level models of the Cosine similarity have a better performance than that of the euclidean distance in most cases except that the XGB# and RF# curves provide a better performance when the false positive rate exceeds 0.002. In reality, online payments service pay attention to the low intervention for normal users, so we tend to adopt the Cosine similarity which is stable in cooperation with other classifiers. In Fig. 8b, the individual-level models with the euclidean distance are better than that of the Cosine similarity, especially when the disturbance is low. It shows that the model of the euclidean distance can better distinguish the characteristics of each individual.

5.4.2 Performance of Model Enhancement

We compare the composite models of different population-level and individual-level models with the pure population-

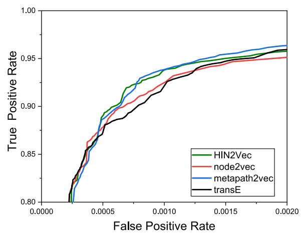


(a) Precision

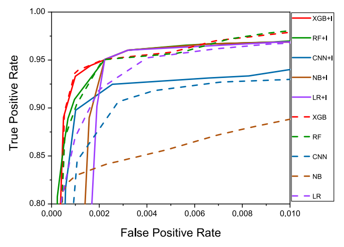


(b) Recall

Fig. 6. The performances of Precision (a) and Recall (b) under different fixed Disturbances in behavioral models.

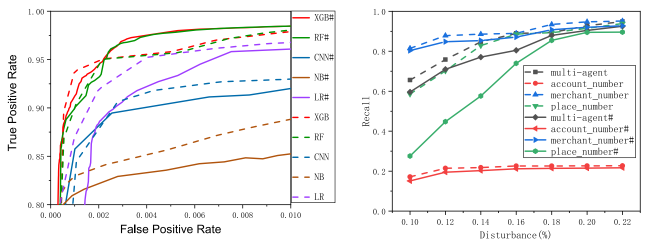


(a) Data enhancement



(b) Model enhancement

Fig. 7. The performance of data enhancement and model enhancement in our model. Figure (a) shows the performance of different network embedding methods as data enhancement in population-level model. Figure (b) shows the performance of model enhancement by combining the individual-level and population-level models.



(a) Population-level **(b)** Individual-level

Fig. 8. The performances of the population-level model (a) and individual-level model (b) under different similarity calculations. We use "#" to represent the versions with euclidean distance on the population-level model and Cosine similarity on the individual-level model, respectively.

level models. Fig. 7b shows the *ROC curves* of five widely-used population-level models, as described in Section 5.1.2, cooperating with the individual-level model.

We observe that the individual-level model has a slight complementary effect on the population-level models with the highest performances, i.e., XGB and RF, than that with the lower performances. When the Disturbance is in the range of 0.002 to 0.005, the individual-level model can bring about 1 percent Recall increase for XGB and RF. In addition to XGB and RF, other population-level models have been greatly improved, especially LR and NB almost have similar performance to XGB and RF with the highest performances when the Disturbance is greater than 0.002. Although CNN cannot match other methods after cooperating with the individual-level model, it has also achieved an obvious improvement. That means that the individual-level model can effectively improve the performance of population-level models.

6 Conclusion and Future Work

For behavioral models in online payment fraud detection, we propose an effective data enhancement scheme by modelling co-occurrence relationships of transactional attributes. Accordingly, we design customized co-occurrence relation networks, and introduce the technique of heterogeneous network embedding to represent online transaction data for different types of behavioral models, e.g., the individual-level and population-level models. The methods are validated by the implementation on a real-world dataset. They outperform the state-of-the-art classifiers with lightweight feature engineering methods. Therefore, our methods can also serve as a feasible paradigm of automatic feature engineering.

There are some interesting issues left to study: (1) An interesting future work is to extend the data enhancement scheme into other types of behavioral models, e.g., the group-level models and generalized-agent-based models, except the population-level and individual-level models studied in this work. (2) It would be interesting to investigate the dedicated enhancement schemes for more advanced individual-level models, since the adopted naive individual-level model does not fully capture the advantages of the proposed data representation scheme based on the techniques of heterogeneous network embedding. (3) It is anticipated to demonstrate the generality of the proposed method by applying it to different real-life application scenarios.

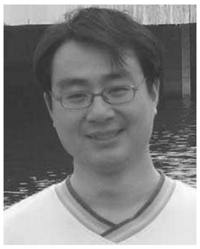
Acknowledgments

The authors sincerely thank the anonymous referees for their helpful comments and suggestions. The research of authors was partially supported by the National Natural Science Foundation of China (NSFC) under Grants 61972287, the Major Project of Ministry of Industry and Information Technology of China under Grant [2018] 282, and Municipal Human Resources Development Program for Outstanding Young Talents in Shanghai.

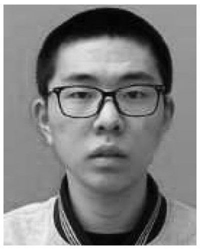
References

[1] B. Cao, M. Mao, S. Viidu, and P. S. Yu, "HitFraud: A broad learning approach for collective fraud detection in heterogeneous information networks," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 769–774.
[2] M. A. Ali, B. Arief, M. Emms, and A. P. A. van Moorsel, "Does the online card payment landscape unwittingly facilitate fraud?" *IEEE Security Privacy*, vol. 15, no. 2, pp. 78–86, Mar./Apr. 2017.
[3] X. Ruan, Z. Wu, H. Wang, and S. Jajodia, "Profiling online social behaviors for compromised account detection," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 176–187, Jan. 2016.
[4] H. Yin *et al.*, "Discovering interpretable geo-social communities for user behavior prediction," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 942–953.
[5] Y.-A. De Montjoye *et al.*., "Unique in the shopping mall: On the reidentifiability of credit card metadata," *Science*, vol. 347, no. 6221, pp. 536–539, 2015.
[6] A. Khodadadi, S. A. Hosseini, E. Tavakoli, and H. R. Rabiee, "Continuous-time user modeling in presence of badges: A probabilistic approach," *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 3, pp. 37:1–37:30, 2018.
[7] F. M. Naini, J. Unnikrishnan, P. Thiran, and M. Vetterli, "Where you are is who you are: User identification by matching statistics," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 358–372, Feb. 2016.
[8] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "Towards detecting compromised accounts on social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 4, pp. 447–460, Jul./Aug. 2017.
[9] A. Alzubaidi and J. Kalita, "Authentication of smartphone users using behavioral biometrics," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1998–2026, Third Quarter 2016.
[10] H. Mazzawi, G. Dalaly, D. Rozenblatz, L. Ein-Dor, M. Ninio, and O. Lavi, "Anomaly detection in large databases using behavioral patterning," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 1140–1149.
[11] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 477–488.
[12] X. Zhou, X. Liang, H. Zhang, and Y. Ma, "Cross-platform identification of anonymous identical users in multiple social media networks," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 2, pp. 411–424, Feb. 2016.
[13] T. Wüchner, A. Cislak, M. Ochoa, and A. Pretschner, "Leveraging compression-based graph mining for behavior-based malware detection," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 99–112, Jan./Feb. 2019.
[14] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.
[15] B. Jia, C. Dong, Z. Chen, K. Chang, N. Sullivan, and G. Chen, "Pattern discovery and anomaly detection via knowledge graph," in *Proc. 21st Int. Conf. Inf. Fusion*, 2018, pp. 2392–2399.
[16] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2019.
[17] M. Abouelenien, V. Pérez-Rosas, R. Mihalcea, and M. Burzo, "Detecting deceptive behavior via integration of discriminative features from multiple modalities," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 5, pp. 1042–1055, May 2017.
[18] W. Youyou, M. Kosinski, and D. Stillwell, "Computer-based personality judgments are more accurate than those made by humans," *Proc. Nat. Academy Sci. United States Amer.*, vol. 112, no. 4, pp. 1036–1040, 2015.

[19] V. Sekara, A. Stopczynski, and S. Lehmann, “Fundamental structures of dynamic social networks,” *Proc. Nat. Academy Sci. United States Amer.*, vol. 113, no. 36, pp. 9977–9982, 2016.
[20] K. Rzecki, P. Plawiak, M. Niedzwiecki, T. Sosnicki, J. Leskow, and M. Ciesielski, “Person recognition based on touch screen gestures using computational intelligence methods,” *Inf. Sci.*, vol. 415, pp. 70–84, 2017.
[21] S. Lee and J. Kim, “WarningBird: Detecting suspicious URLs in twitter stream,” in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp.*, 2012, pp. 1–13.
[22] G. Stringhini, P. Mourlanne, G. Jacob, M. Egele, C. Kruegel, and G. Vigna, “EVILCOHORT: Detecting communities of malicious accounts on online services,” in *Proc. USENIX Secur. Symp.*, 2015, pp. 563–578.
[23] Z. Meng, L. Mou, and Z. Jin, “Hierarchical RNN with static sentence-level attention for text-based speaker change detection,” in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 2203–2206.
[24] A. Rawat, G. Gugnani, M. Shastri, and P. Kumar, “Anomaly recognition in online social networks,” *Int. J. Secur. Appl.*, vol. 9, no. 7, pp. 109–118, 2015.
[25] C. VanDam, J. Tang, and P. Tan, “Understanding compromised accounts on Twitter,” in *Proc. ACM Int. Conf. Web Intell.*, 2017, pp. 737–744.
[26] W. Zhao, J. Zhu, M. Yang, D. Xiao, G. P. C. Fung, and X. Chen, “A semi-supervised network embedding model for protein complexes detection,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 8185–8186.
[27] C. Li, Y. Duan, H. Wang, Z. Zhang, A. Sun, and Z. Ma, “Enhancing topic modeling for short texts with auxiliary word embeddings,” *ACM Trans. Inf. Syst.*, vol. 36, no. 2, pp. 11:1–11:30, 2017.
[28] S. F. A. Baqueri, M. Adnan, B. Kochan, and T. Bellemans, “Activity-based model for medium-sized cities considering external activity-travel: Enhancing FEATHERS framework,” *Future Gener. Comput. Syst.*, vol. 96, pp. 51–63, 2019.
[29] Q. Chen, Q. Hu, J. X. Huang, and L. He, “CAN: Enhancing sentence similarity modeling with collaborative and adversarial network,” in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 815–824.
[30] G. Catolino, F. Palomba, A. D. Lucia, F. Ferrucci, and A. Zaidman, “Enhancing change prediction models using developer-related factors,” *J. Syst. Softw.*, vol. 143, pp. 14–28, 2018.
[31] B. Liu, P. Tan, and J. Zhou, “Enhancing predictive modeling of nested spatial data through group-level feature disaggregation,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1784–1793.
[32] E. K. H. Leung, K. L. Choy, P. K. Y. Siu, G. T. S. Ho, C. H. Y. Lam, and C. K. M. Lee, “A B2C E-commerce intelligent system for re-engineering the e-order fulfilment process,” *Expert Syst. Appl.*, vol. 91, pp. 386–401, 2018.
[33] Z. Li *et al.*, “FAIR: Fraud aware impression regulation system in large-scale real-time E-commerce search platform,” in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1898–1903.
[34] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, “Heterogeneous information network embedding for recommendation,” *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, Feb. 2019.
[35] T. Fu, W. Lee, and Z. Lei, “HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning,” in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1797–1806.
[36] X. Huang, J. Li, N. Zou, and X. Hu, “A general embedding framework for heterogeneous information learning in large-scale networks,” *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 6, pp. 70:1–70:24, 2018.
[37] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, “SHINE: Signed heterogeneous information network embedding for sentiment link prediction,” in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 592–600.
[38] S. Fan, C. Shi, and X. Wang, “Abnormal event detection via heterogeneous information network embedding,” in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1483–1486.
[39] M. U. K. Khan, H. S. Park, and C. Kyung, “Rejecting motion outliers for efficient crowd anomaly detection,” *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 541–556, Feb. 2019.
[40] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, “MADAM: Effective and efficient behavior-based android malware detection and prevention,” *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 1, pp. 83–97, Feb. 2018.
[41] W. Tang, B. Li, S. Tan, M. Barni, and J. Huang, “CNN-based adversarial embedding for image steganography,” *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 8, pp. 2074–2087, Aug. 2019.
[42] J. Yang and C. Eickhoff, “Unsupervised learning of parsimonious general-purpose embeddings for user and location modeling,” *ACM Trans. Inf. Syst.*, vol. 36, no. 3, pp. 32:1–32:33, 2018.
[43] J. Xie, D. Yang, and J. Zhao, “Composite anti-disturbance model reference adaptive control for switched systems,” *Inf. Sci.*, vol. 485, pp. 71–86, 2019.
[44] I. Molloy *et al.*, “Graph analytics for real-time scoring of cross-channel transactional fraud,” in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2016, pp. 22–40.
[45] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King, “STELLAR: Spatial-temporal latent ranking for successive point-of-interest recommendation,” in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 315–322.
[46] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.
[47] A. Bordes, N. Usunier, A. Garc ía-Durán, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.
[48] Y. Dong, N. V. Chawla, and A. Swami, “metapath2vec: Scalable representation learning for heterogeneous networks,” in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 135–144.



Cheng Wang (Senior Member, IEEE) received the MS degree from the Department of Applied Mathematics, Tongji University, Shanghai, China, in 2006, and the PhD degree from the Department of Computer Science, Tongji University, Shanghai, China, in 2011. He is currently a professor with the Department of Computer Science, Tongji University. His research interests include cyberspace security and intelligent information services.



Hangyu Zhu received the bachelor’s degree in computer science from Shanghai Maritime University, Shanghai, China, in June 2018. He is currently working toward the master’s degree with the Department of Computer Science, Tongji University, Shanghai, China. His research interests include network embedding and online fraud detection.