Future Generation Computer Systems 116 (2021) 145–155

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Large-scale online multi-view graph neural network and applications ☆

Zhao Li [a,*], Yuying Xing [a], Jiaming Huang [a], Haobo Wang [b], Jianliang Gao [c], Guoxian Yu [d]

[a] Alibaba Group, Hangzhou 311121, China
[b] College of Comp. Sci. and Tech., Zhejiang University, Hangzhou 310027, China
[c] Inf. Sci. and Eng., Central South University, Changsha 410083, China
[d] School of Software, Shandong University, Jinan 250101, China

ARTICLE INFO

Article history:
Received 23 June 2020
Received in revised form 27 September 2020
Accepted 15 October 2020
Available online 28 October 2020

Keywords:
Heterogeneous
Graph Neural Network
Multi-view
Online

ABSTRACT

Recently popularized Graph Neural Network (GNN) has been attaching great attention along with its successful industry applications. This paper focuses on two challenges traditional GNN frameworks face: (i) most of them are transductive and mainly concentrate on homogeneous networks considering single typed nodes and edges; (ii) they are difficult to handle the real-time changing network structures as well as scale to big graph data. To address these issues, a novel attention-based Heterogeneous Multi-view Graph Neural Network (aHMGNN) solution is introduced. aHMGNN models a more intricate heterogeneous multi-view network, where various node and edge types co-exist and each of these objects also contain specific attributes. It is end-to-end, and two stages are designed for node embeddings learning and multi-typed node and edge representations fusion, respectively. Experimental studies on large-scale spam detection and link prediction tasks clearly verify the efficiency and effectiveness of our proposed aHMGNN. Furthermore, we have implemented our approach in one of the largest e-commerce platforms which further verifies that aHMGNN is arguably promising and scalable in real-world applications.

1. Introduction

As one of the most popular deep learning frameworks for graph data modelling and analysis, Graph Neural Network (GNN) [1–3] has been attaching great attention in recent years. The basic idea of this promising technique is to acquire the embedded representation of the target node by capturing and aggregating the information of its local neighbourhoods, which is closely related to graph/network embedding [4,5]. Due to its strong expressiveness and superior performance,GNN was widely used for various industrial applications such as social network analysis [6], recommendation systems [7,8] and anomaly detection [9]. There are also many downstream network learning tasks, e.g. node classification [10], link inferences [11] (e.g. edge classification and link prediction) and community detection [12], can be disposed by this paradigm. Despite that current GNN models (GNNs) [13] have shown the great success on link inference applications [14], they still face several challenges due to the complexity of real-world graphs. In this paper, we mainly focus on two difficulties of traditional GNNs.

On the one hand, most existing GNN approaches [15,16] target to address homogeneous graphs (used interchangeably with *networks*) with single typed nodes and edges. However, they ignore the network heterogeneity of multiple node types and edge types. To tackle this problem issue, several works [17,18] have been presented to mine specific network structures, such as multi-typed nodes [9,17] or edges [19], and attributed multiplex heterogeneous graphs [13] considering different types of nodes and edges as well as node features.

Fig. 1 shows the user–item bipartite graph of e-commerce fraud transaction detection scenario [20,21], where multi-typed node and edge objects recording different operations co-exist, and each type of objects has specific contents (given as attribute vectors). Unfortunately, to the best of our knowledge, none of the previous state-of-the-art GNNs can handle such complicated non-Euclidean domain. Besides, most of them [15] are transductive and limited for capturing the evolving network inputs [22]. In fact, one application platform may generate billions of data every day, and it is unrealistic to conduct repetitive training to address previously unseen nodes and edges.

On the other hand, the majority of current GNNs [15,23] assume the loaded graph data is static. Factually, graphs are in nature dynamic in a way that the newly-added nodes or edges may cause the appearance and overlaying of information in graphs [14]. Typically, in some practical industrial applications (e.g. the above mentioned e-commerce spam detection task), it is

☆ An earlier version of this paper was presented at DASFAA2020.

* Corresponding author.
E-mail addresses: lizhao.lz@alibaba-inc.com (Z. Li), yuying.xyy@alibaba-inc.com (Y. Xing), jimmy.hjm@alibaba-inc.com (J. Huang), wanghaobo@zju.edu.cn (H. Wang), gaojianliang@csu.edu.cn (J. Gao), gxyu@sdu.edu.cn (G. Yu).

https://doi.org/10.1016/j.future.2020.10.018
0167-739X/

very essential to detect real-time transactions [24] since they may greatly affect the user experience and service quality. However, how to deal with such graph dynamicity for online model deployment is still not well studied for homogeneous/heterogeneous GNNs. Moreover, the scalability of GNNs is also challenged since the limitation of core assumptions underlying these algorithms' designs.

To remedy these challenging issues, this work introduces a novel idea to handle multiple types of attributed nodes and edges in **a**ttention-based **h**eterogeneous **m**ulti-view graphs based on **G**raph **N**eural **N**etwork (aHMGNN). The developed aHMGNN model is composed of two stages where the first one is inspired by the widely-used GraphSAGE algorithm [15] which captures the network structure of our studied heterogeneous scenario to extract embeddings for each type of node objects. The second stage designs a multi-view generator based on subspace learning strategy and attention mechanism in an inductive manner. It then fuses multiple node and edge feature views (i.e. feature representations) to infer different types of edges by aggregating the learned virtual subspaces. In addition, we also introduce a new variable graph data storage module and Cumulative Distribution Function (CDF) based double accumulates approach for online system implementation of our proposed aHMGNN.

The main contributions of this paper are summarized as follows:

- A novel end-to-end heterogeneous multi-view graph neural network algorithm is studied, which makes the first attempt to handle heterogeneous network structure considering multiple node types and edge types, as well as attributes of these objects.
- Our designed inductive multi-view generator can not only fuse multi-typed node objects and edge objects, but also takes the commonality and individuality information of multiple node and edge views into account.
- This paper develops a new and efficient variable network storage method and dynamic node neighbourhood sampling strategy for dynamic graphs update and online system implementation of aHMGNN.
- We conducted comprehensive studies on large-scale online and offline spam detection application and offline link prediction task. The experimental results show that our aHMGNN outperforms existing representative and related deep graph solutions.

The rest of this paper is organized as follows: Section 2 reviews several related heterogeneous and dynamic graph studies; Section 3 elaborates on the proposed aHMGNN model; The offline effectiveness validation and online system implementation of this approach can be found in Section 4 and Section 5, respectively; Section 6 concludes our work.

2. Related work

GNNs have been studied for both homogeneous and heterogeneous graph embedding [14]. In recent years, limited dynamic GNN works are also proposed to capture the evolving network structures. We focus our discussion of related works on several representative heterogeneous and dynamic graph learning algorithms. A recent comprehensive review of GNN can be found at [14].

2.1. Heterogeneous graph embedding methods

Conventional homogeneous networks [15,25] assume only single typed nodes and edges in graphs, where all nodes share the same model parameter and dimensional feature space. While in a heterogeneous graphs, more than one kind of node type and edge type are allowed, as well as different feature dimension of multi-typed nodes. There are several GNN-based algorithms [9,17,18,26–30] are developed to capture specific heterogeneous graph structures. For instance, [9] proposes a graph neural network approach named GEM for detecting malicious accounts at Alipay (a mobile cashless payment platform), which aims to learn the discriminative embeddings of nodes with various types from heterogeneous account-device graphs. [17,27,28] are introduced to address heterogeneous scenarios considering multi-typed edge information. To further handle multiple node and edge types, [17] presents a heterogeneous GNN solution based on the hierarchical attentions which simultaneously exploits the node-level and semantic-level importance of different nodes and meta-paths. Besides, [31] develops a multi-view graph convolutional neural network (MV-GCN) model based on matrix completion method. It solves the interactive relationship and the content information of different node objects by directly concatenating these information as a single view for link prediction task.

Furthermore, some heterogeneous works [13,19,32] incorporate node features into graph representation learning. Typically, [13] designs a GNN-based framework named GATNE which supports both transductive and inductive learning for embedding generation of multi-typed nodes. It first combines meta-path based random walk and features of nodes to learn node sequences, and then performs skip-gram over the generated node sequences to learn embedded representations.

Most of these node embedding learning schemes adopt meta-path based strategy, and have proven their power in heterogeneous graph modelling. However, all of them do not generalize to a more complicated but really existed heterogeneous multi-view graph structure, where multiple types of nodes and edges co-exist, may be accompanied by their rich attributes. In addition, most of them [33,34] also challenge to capture unseen network node/edge inputs and adapt to large-scale graph tasks.

2.2. Dynamic GNNs

Most existing GNNs are mainly designed for offline tasks, and lately limited efforts [35–39] have been made to overcome the difficulty of the evolving nodes and edges. [35] introduces a dynamic graph neural network framework named DGNN that keeps updating node information by capturing the sequential information of edges (interactions), the time intervals between edges and information propagation coherently. [36] presents Dynamic Self-Attention Network (DySAT) to operate on dynamic graphs and learn node representations by jointly employing self-attention layers along two dimension including structural neighbourhood and temporal dynamics. Then, [37] develops a approach named EvolveGCN, which adapts the graph convolutional network (GCN) model along the temporal dimension without resorting to node embeddings. It captures the dynamism of the graph sequence through using an RNN to evolve the GCN parameters.

All these models are designed to address the dynamic information of the evolving networks from the perspective of algorithm innovations. In addition, [40] designs an in-memory, distributed graph data management system aimed at managing a large-scale dynamically changing graph, and supporting low-latency query processing over it by engineering technology. However, it only considers graph data update, but ignores the difference of intrinsic structures (i.e. the nearest neighbourhood information) of nodes evolving nodes in real-time.

This study develops a new attention-based heterogeneous multi-view GNN method (aHMGNN) to address the above mentioned heterogeneity and dynamicity challenges of graphs. Sections 3 and 5 describe the architecture and online system implementation of this model, respectively.

3. aHMGNN architecture

In this section, we first define our studied heterogeneous problem, and then detail the proposed aHMGNN model.

3.1. Problem formulation

Denote a heterogeneous network by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{A})$ with R kinds of nodes and T types of edges between these nodes. The following contents show the definitions of these objects. Table 1 illustrates the notations used in this paper.

- **Node Definition:** $\mathcal{V} = \{\mathcal{V}^r\}_{r=1}^{R}$ collects R types of nodes, where $\mathcal{V}^r = \{v_1^r, v_2^r, \ldots, v_{n_r}^r\}(1 \le r \le R)$ represents the rth type node set including n_r nodes. $\mathcal{X} = \{\mathcal{X}^r\}_{r=1}^{R}$ show attributes of these multi-typed nodes. $\mathcal{X}^r = \{\mathbf{x}_1^r, \mathbf{x}_2^r, \ldots, \mathbf{x}_{n_r}^r\}$ is feature set of the rth type nodes with $\mathbf{x}_i^r \in \mathbb{R}^{d^r}(1 \le i \in n_r)$ representing the feature vector of a certain node v_i^r.
- **Edge Definition:** Similarly, we define $\mathcal{E} = \{\mathcal{E}^t\}_{t=1}^{T}$ and $\mathcal{A} = \{\mathcal{A}^t\}_{t=1}^{T}$ to represent T types of edge sets and corresponding edge features. $\mathcal{E}^t = \{e_{ij}^t\}(1 \le t \le T, 1 \le i \le n_{r_1}, 1 \le j \le n_{r_2})$ is the tth type edge set which contains m_t links, and $\mathcal{A}^t = \{\mathbf{a}_{ij}^t\}$ show the attributes of these edges. If two nodes (can be from the same or different kinds of node sets) in graph, e.g. $v_i^{r_1}$ and $v_j^{r_2}$, are linked by the tth type of edge, then $e_{ij}^t \in \mathcal{E}^t$, and $\mathbf{a}_{ij}^t \in \mathbb{R}^{d^t}$ represents edge features of e_{ij}^t. Vector $\mathbf{Y}_{ij}^t \in \mathbb{R}^q$ is the encoded label set of e_{ij}^t, where $\mathbf{Y}_{ij}^t(l) = 1$ represents that e_{ij}^t is annotated with the lth concept label; $\mathbf{Y}_{ij}^t(l) = 0$ otherwise.

Given $\mathcal{G}$, the objective of aHMGNN is to make predictions for multi-typed edges (i.e $\mathcal{F} = \{\mathbf{F}^t\}_{t=1}^{T}$) by learning complicated relations of $\mathcal{G}$, where $\mathbf{F}^t \in \mathbb{R}^{m_t \times q}$ represents the predicted likelihoods of w.r.t $\mathcal{E}^t$. Notice that the presented aHMGNN can also handle multi-typed edges annotated with different sizes of label set. Since the datasets used in this paper share the same labels across various kinds of edges, thus we use the unified symbol representation for a better model understanding.

3.2. Methodology

The presented aHMNN framework are composed of two stages. The first stage targets to learn the node embedded representations for each type of nodes. The second stage develops an inductive multi-view generator based on subspace learning and attention mechanism to aggregate multi-typed node and edge representations for link inference tasks. Next, we will introduce these two stages individually.

3.2.1. Node embeddings generation via heterogeneous network learning

Fig. 1 depicts an illustrate example of the first stage of aHMGNN, which aims to learn the node embeddings for multi-typed nodes. Let $v_i^r \in \mathcal{V}^r(1 \le i \le n_r, 1 \le r \le R)$ be a certain typed node of $\mathcal{G}$, we split its overall representation $\mathbf{h}_i^r$ into two parts: the node embedded representation with structural information $\mathbf{s}_i^r$, and the known node attributes $\mathbf{x}_i^r$. In order to obtain $\mathbf{s}_i^r$, this paper first reconstructs the heterogeneous network $\mathcal{G}$ as a new one as shown in Fig. 2. Specifically, for two nodes (e.g. v_i^r and $v_j^{r_1}(1 \le r, r_1 \le R)$) from the same or different node types, we construct a link between them if there exists at least one type of edges between these two nodes in the original $\mathcal{G}$. That is to say, if the condition $\exists e_{ij}^t(1 \le t \le T)$ is satisfied, we have $\tilde{e}_{ij} \in \tilde{\mathcal{E}}$. By this means, a new heterogeneous network $\tilde{\mathcal{G}}(\mathcal{V}, \tilde{\mathcal{E}})$ is constructed. The reconstruction of $\mathcal{G}$ can not only tackle the challenge that existing graphsage models are difficult to generalize to heterogeneous networks, but also provide more guidance information for node embeddings learning of multi-typed nodes, as well as alleviate node sparse problem of sampled neighbourhoods in case of considering single typed edges.

After that, we combine the complicated network information and node attributes of the reconstructed $\tilde{\mathcal{G}}$ for node embedding (i.e. $\mathbf{s}_i^r$) generation of v_i^r. Due to the heterogeneity of $\tilde{\mathcal{G}}$ and in order to better match the data distribution of different node types, this paper first samples fixed-size K depth heterogeneous neighbourhood nodes for v_i^r rather than homogeneous one as done in original GraphSAGE [15]. Fig. 2 shows an example of this sampling process. In the user–item bipartite graph, we sample specific number of item nodes as 1-depth neighbours of the target user node, and users with different node type (i.e. items) from 1-depth neighbourhood as its 2-depth neighbours. Following the same idea, we can sample Kth depth neighbours for the target nodes. Note that to avoid an explosive increase of training complexity, this paper does not use all heterogeneous neighbours for sampling.

Denote $\tilde{N}(v_i^r)$ is the sampled neighbourhood node number of v_i^r drawn from the set $\{v_j^{r_1} \in \mathcal{V}^{r_1} : (v_i^r, v_j^{r_1}) \in \tilde{\mathcal{E}}\}$, aHMGNN then adopts forward propagation algorithm to generate node embedding of v_i^r by aggregating the neighbourhood features of the local sampled node sets. In the kth ($k \in 1, 2, \ldots, K$) iteration, we utilize the mean aggregator [15] to calculate $\mathbf{s}_i^r$ as follows:

$$\mathbf{s}_i^r(k) \longleftarrow \delta(\widetilde{\mathbf{W}} \cdot MEAN(\{\mathbf{s}_i^r(k-1)\} \cup \{\mathbf{s}_j^{r_1}(k-1), \forall v_j^{r_1} \in \tilde{N}(v_i^r))\}) \tag{1}$$

where $\mathbf{s}_i^r(k)(1 \le k \le K)$ is the obtained embedded representation of node v_i^r at the kth step. It incorporates the feature information from the local heterogeneous neighbourhoods of v_i^r, and the elementwise mean of the neighbourhood vectors simply are taken in $\{\mathbf{s}_j^{r_1}(k-1), \forall v_j^{r_1} \in \tilde{N}(v_i^r)\}$. $\delta(.)$ is the sigmoid function, and $AGGREGATE_k = \widetilde{\mathbf{W}} \cdot MEAN(\cdot)$ is the mean aggregator function. $\widetilde{\mathbf{W}} = \{\widetilde{\mathbf{W}}^k\}_{k=1}^{K}$ collects the weight parameter matrices of K aggregator functions, which is responsible for information propagation between different layers of the model. Since our model involves supervised information of edges, thus we adopt general cross-entropy loss function [41] for the parameter learning of K aggregator functions. Through Eq. (1), we can learn the node embedding with structural information of v_i^r by iteratively aggregating their K depth sampled neighbourhood node information, where $k = 0$ representations are defined as the input node features (i.e. $\mathbf{s}_i^r(0) = \mathbf{x}_i^r$). Then the overall representation of this node can be learned via $\mathbf{h}_i^r = [\mathbf{s}_i^r(K), \mathbf{x}_i^r]$ with the graph embedded information and attributes are both considered, where $\mathbf{h}_i^r$ represents the concatenation of $\mathbf{s}_i^r(K)$ and $\mathbf{x}_i^r$.

To sum up, the first stage of aHMGNN not only captures complicated relations of multi-typed nodes of $\mathcal{G}$, but also learn the node embeddings in an inductive manner. Empirical studies in experiments also demonstrate the effectiveness of our model on disposing heterogeneous structure of $\mathcal{G}$.

3.2.2. Data fusion of multiple node and edge views

Existing heterogeneous GNNs generally assume the attributes of node objects is single-viewed. However, in the real-world applications, there might be different types of nodes and edges which can be naturally treated as multiple views. To bridge this gap, we introduce a multi-view generator based on subspace learning and attention mechanism to fuse multi-typed node and edge feature views for link inference task as shown in Fig. 3. Considering there are more than one edge type in graphs, and the number of different edge types may be imbalanced extremely in practice, we make prediction for each type of edges respectively.

Table 1
The notations used in this paper.

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{A})$	The input heterogeneous graph
$\mathcal{V}^r/\mathbf{X}^r (1 \le r \le R)$	The rth type (node set/corresponding node attributes) of $\mathcal{G}$
$\mathcal{E}^t/\mathbf{A}^t (1 \le t \le T)$	The tth type (edge set/corresponding edge contents) of $\mathcal{G}$
n_r/m_t	The number of the rth type nodes/ the tth type edges
q	The total number of edge labels
v_i^r	The ith node from the rth type node set
$\mathbf{h}_i^r$	The overall representation of v_i^r
$\mathbf{s}_i^r/\mathbf{x}_i^r$	The generated embedding/attributes w.r.t v_i^r
$\mathbf{H}$	Shared subspace across multi-typed node and edge feature views
V	The total number of node and target edge views
$\mathbf{P}^v (1 \le v \le V)$	The projected matrix w.r.t the vth node/edge view
$\mathbf{C}^v (1 \le v \le V)$	The generated virtual subspace w.r.t the vth node/edge view

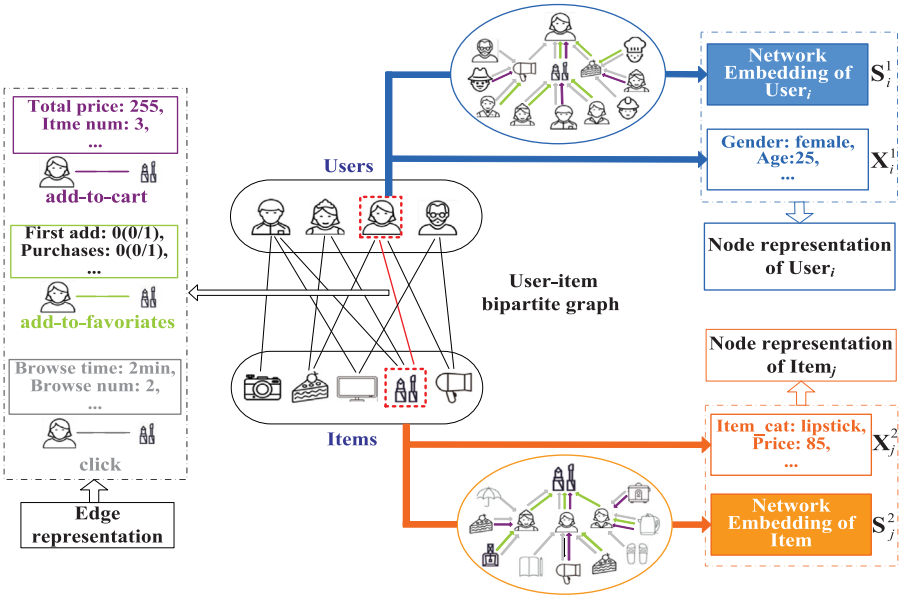


Fig. 1. An illustrate example and the first stage of our studied attention-based heterogeneous multi-view graph neural network(aHMGNN). There are two kinds of nodes (i.e. users and items) and three types of edges (i.e. click, add-to-favourites and add-to-cart) associated with them co-exist in the user–item bipartite grape, and each of these node/edge objects also contain specific attributes.

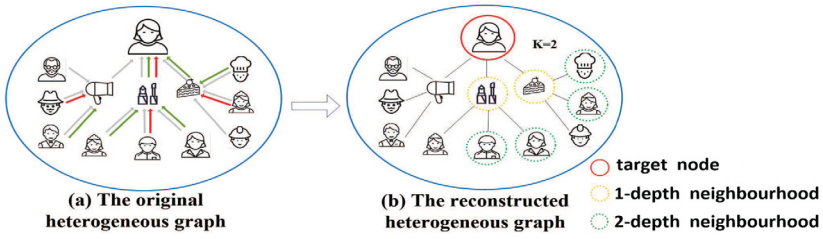


Fig. 2. Heterogeneous network reconstruction (The left figure shows the original heterogeneous network $\mathcal{G}$, and the right one is the new graph $\tilde{\mathcal{G}}$ which is reconstructed by creating a link if there exists at least one type of edges between two nodes).

Take the tth type of edges (i.e. $\mathcal{E}^t (1 \le t \le T)$) as an example, we assume there are V views including $V-1$ node representations (i.e. $\mathbf{H}^r (1 \le r \le R)$) w.r.t $\mathcal{E}^t$, and the Vth view is the edge feature space of $\mathcal{E}^t$ (i.e. $\mathbf{A}^t$). To better fuse representations of these views, aHMGNN first creates a virtual subspace for each node/edge feature view, and simultaneously approximates these generated virtual subspaces to the shared subspace $\mathbf{H}$ across multiple views as follows:

$$Loss_{MV} = \frac{1}{V}\sum_{v=1}^{V} \alpha_v \parallel \mathbf{C}^v - \mathbf{H} \parallel_2^2, \ where \ \mathbf{C}^v = \mathbf{H}^v\mathbf{P}^v \tag{2}$$

where $\mathbf{H}^v \in \mathbb{R}^{m_t \times d_v} (d_v = d^r/d^t)$ represents the vth node ($1 \le v \le V-1$)/edge ($v = V$) feature view, and m_t means the total edge number of $\mathcal{E}^t$. $\mathbf{P}^v \in \mathbb{R}^{d_v \times d}$ is the projection matrix, which patterns specific characteristics of the vth view, and can be directly used for model testing. $\mathbf{C}^v = \mathbf{H}^v\mathbf{P}^v$ represents the target generated virtual subspace of the vth view. $\mathbf{H} \in \mathbb{R}^{m_t \times d}$ is the shared subspace across multiple node feature views and the target edge representation. Eq. (2) maps V original node and edge features with high dimension to the multi-view shared low-dimensional subspace $\mathbf{H}$. By this means, aHMGNN can predict $\mathcal{E}^t$ in an inductive manner, and reduce the time costs of making prediction for unseen edge inputs by directly using the aggregated virtual subspace $\frac{1}{V}\sum_{v=1}^{V} \alpha_v \mathbf{C}^v$.

Furthermore, we adopt the widely used self-attention mechanism [42] to measure the contributions of different views for the

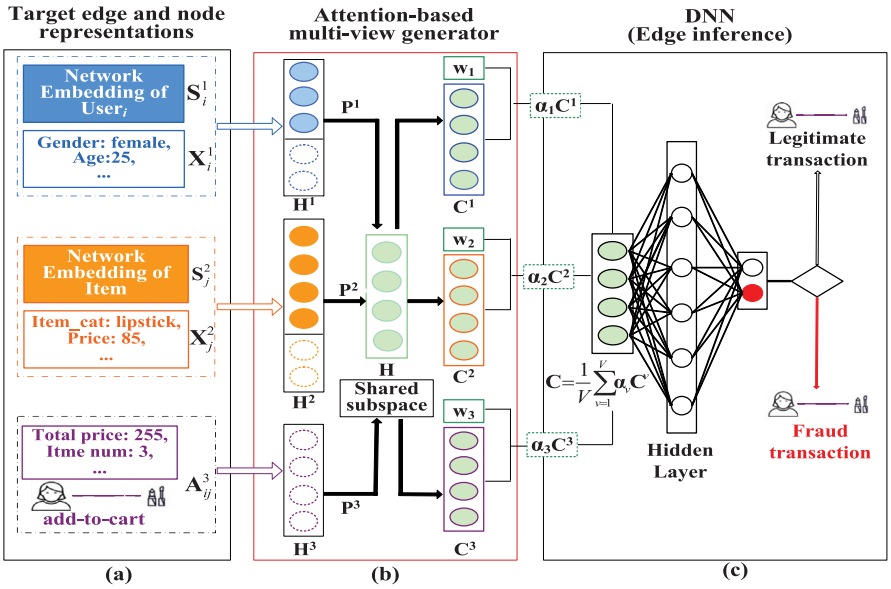


Fig. 3. The second stage of heterogeneous multi-view graph neural network. (a) The target edge type and corresponding node representations are used as inputs to the second stage of aHMGNN; (b) aHMGNN designs an attention-based multi-view generator which first learns a virtual subspace for each node/edge view; (c) It then exploits the aggregated virtual subspace across multiple views for link inference task.

data fusion, where α_v represents the importance of the vth view. Eq. (2) defines the calculation of α_v as follows:

$$\alpha_v = softmax(tanh(\mathbf{C}^v \mathbf{W}_v)) \tag{3}$$

where the patterned $\mathbf{W}_v$ is the weight parameter matrix w.r.t the vth view. This paper first learns $\mathbf{W}_v$ for the vth node/edge feature view during model training, and it then uses the general tanh function and softmax function to obtain the final coefficients α_v as shown in Eq. (3).

Any off-the-shelf classification models can be applied to predict the labels of edges $\mathcal{E}^t$. In this work, we use a common Deep Neural Network method (DNN [43]) as base classifier. Thus, for a seen or unseen edge e_{ij}^t, we can make prediction for it w.r.t the lth label by applying the following equation:

$$\mathbf{F}_{ij}^t(l) = f(\mathbf{a}^T \mathbf{c}_{ij}^t)(1 \le l \le q) \tag{4}$$

where $f(.)$ is general softmax function, $\mathbf{c}_{ij}^t$ represents the aggregated virtual subspace vector w.r.t e_{ij}^t, and $\mathbf{a}$ is the corresponding weighted vector of e_{ij}^t. To this end, aHMGNN can infer specific type of links in an inductive manner.

3.2.3. The unified objective function of aHMGNN

In summary, the formulated unified objective function of our proposed aHMGNN are summarized as shown in Eq. (5).

$$Loss = \lambda Loss_{MV} + Loss_{CE} \tag{5}$$

where $Loss_{MV}$ is responsible for the data fusion of multiple node and target edge views, and λ is a trade-off parameter. $Loss_{CE}$ represents the cross entropy loss which is composed of two parts: one is used for the parameter learning of multiple aggregator functions in the first stage, and the other is classification loss of the utilized DNN model in the second stage. Eq. (6) defines this loss function as follows:

$$Loss_{CE} = -\sum_{l=1}^{q}\sum_{i=1}^{m_t}(\mathbf{Y}_i^t(l) log(\mathbf{F}_i^t(l)) + (1 - \mathbf{Y}_i^t(l)) log(1 - \mathbf{F}_i^t(l))) \tag{6}$$

where $\mathbf{Y}_i^t(l)$ and $\mathbf{F}_i^t(l)$ represent the ground truth and the predicted likelihoods of the ith edge w.r.t the lth label, respectively. To compute the parameters of the model, this paper adopts the common gradient descent method [44] for optimization until the overall model loss is converged. To sum up, the objective of aHMGNN is to minimize the overall losses of two terms in Eq. (5), which makes node embedding generation, multi-typed node and edge representations fusion, and link inference task in a coordinate fashion. The proposed aHMGNN is end-to-end and Algorithm 1 summarizes the procedure of our model.

Algorithm 1 The procedure of aHMGNN

Input:
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{A})$: the input heterogeneous graph with R node types and T edge types;
K: the depth for neighbourhoods sampling; λ: the trade-off parameter;
$\{\mathbf{Y}^t\}_{t=1}^T$: The label sets of T types of edges;
Output:
$\{\mathbf{F}^t\}_{t=1}^T$: The predicted label spaces of T types of edges;
Reconstruct $\mathcal{G}$ into $\tilde{\mathcal{G}}(\mathcal{V}, \tilde{\mathcal{E}})$ as described in Section 3.2.1;
For $r = 1 : R$
  For $v_i^r \in \mathcal{V}^r (1 \le i \le n_r)$
    Sample fixed-size (i.e. $\tilde{N}(v_i^r)$) K depth neighbourhoods drawn from the corresponding heterogeneous node sets $\{v_j^{r1} \in \mathcal{V}^{r1} : (v_i^r, v_j^{r1}) \in \tilde{\mathcal{E}}\}(1 \le j \le n_{r_1}, 1 \le r_1 \le R)$;
    Compute the node embeddings of v_i^r, i.e. $\mathbf{s}_i^r(k)(1 \le k \le K)$, by the k-th aggregator functions via Eq. (1);
    Obtain the overall node representation of v_i^r by $\mathbf{h}_i^r = [\mathbf{s}_i^r(K), \mathbf{x}_i^r]$;
  End For
End For
For $t = 1 : T$
  Learn the V virtual subspaces $\{\mathbf{C}^v\}_{v=1}^V$ via Eq. (2);
  Obtain the aggregated virtual subspace $\mathbf{C}$ via $\frac{1}{V}\sum_{v=1,v}^{V}\mathbf{C}^v$;
  Minimize the edge classification loss w.r.t. q labels via Eq. (5);
  Output the predicted labels of the t-th type of edges (i.e. $\mathbf{F}^t$);
End For

4. Offline evaluation

In this section, we first introduce the experimental setup of this paper. Then, we evaluate the superiority of the proposed aHMGNN on two offline graph-mining tasks, i.e. large-scale e-commerce spam detection and link prediction. Finally, we discuss the scalability, parameter sensitivity and time complexity of our model.

Table 2
Statistics of three datasets used for the experiments. n and m represent the number of nodes and edges in the heterogeneous graph, respectively. R and T are the number of node types and edge types, respectively.

Dataset	n	m	R	T
Amazon	10,166	148,865	1	2
Twitter	10,000	331,899	1	4
Taobao	1500,000,000	4104,603,557	2	3

4.1. Experimental setup

4.1.1. Datasets

Three datasets including two public heterogeneous network datasets [13] and one real dataset acquired from Taobao platform are adopted in this paper for model effectiveness validation of our developed model. Table 2 lists the node and edge information of these datasets.

- **Amazon**[1]**:** only uses the product meta data of electronics category, where two types of edges between products are considered including coviewing and co-purchasing, while price, sales-rank, brand, and category are selected as the product attributes.
- **Twitter**[2]**:** contains four types of links to measure the relationship between different Twitter users including re-tweet, reply, mention, and friendship/follower.
- **Taobao:** consists of two types of node sets (i.e. users and items), and three types of interactions (click, add-to-favourites, add-to-cart) between users and items.

4.1.2. Comparing methods

To evaluate the offline effectiveness of aHMGNN, three parts of experiments are designed in this paper. In the first experiment part, we apply aHMGNN to the large-scale e-commerce spam detection problem on the acquired real Taobao dataset to explore the performance of aHMGNN on edge classification task. Five algorithms including two related spam detection methods (DNN [43] and IFCM [24]) and three variants of the presented aHMGNN are used for comparisons in this part. The second part is designed for validation of aHMGNN on link prediction task. Four representative and related network embedding methods (ANRL [16], MNE [28], DGMI [30], GATNE [13] (GATNE_T, GATNE_I) and three aHMGNN variants are compared on Amazon and Twitter datasets to evaluate the performance of aHMGNN on integrating structural information and attributed information of multi-typed nodes and edges. The third part shows the running environment, feature selection, parameter sensitivity, scalability analysis and time complexity of our model. The following content details the competitors used in this paper and we turn the parameters of these methods based on the suggestions given in the respective code or papers.

- **Heterogeneous Network Representation Learning Methods**
  (i) **ANRL**[16]: a homogeneous network method which models attributed single-typed nodes and without edge contents.
  (ii) **MNE**[28]: handles heterogeneous network with single node type and multiple edge types.
  (iii) **GATNE**[13]: is designed for addressing multiplex heterogeneous network data where more than one type of nodes and edges are considered, and each type of nodes are attributed. Two algorithms are introduced in this approach, where GATNE_T is transductive which cannot handle unobserved data and only uses network structure information. GATNE-I is an inductive model which considers both structure information and node attributes of multiplex network.
  (iv) **DGMI**[30]: is an attention-based unsupervised framework which aims to learn node embeddings from the attributed multiplex network.
- **Spam Detection Methods**
  (i) **DNN**[43]: Deep Neural Network(DNN), a typical deep neural network learning method which uses forward propagation calculation for sample learning.
  (ii)**IFCM**[24]: a deep learning model designed for fraud detection in large-scale e-commerce platform, which exploits multi-layer DNN to predict spam activities.
- **Variants of aHMGNN**
  (i) **HMGNN_Nh**: only considers one type of edges during the heterogeneous graph reconstruction to evaluate the importance of integrating multi-typed edges in heterogeneous scenario.
  (ii) **HMGNN_Nm**: targets to fuse the feature views of different node and edge types by simply concatenating them into a single view for link inference application.
  (iii) **HMGNN**: vanilla aHMGNN model, which does not adopt attention mechanism and directly use $\frac{1}{V}\sum_{v=1}^{V}\mathbf{C}^v$ where $\mathbf{C}^v = \mathbf{H}^v\mathbf{P}^v$ for data fusion of multiple node and edge representations.

4.1.3. Evaluation metrics

This paper adopts the widely used binary classification metrics including the area under the Precision–recall curve (PR-AUC) and F1-score for the effectiveness evaluation of aHMGNN. The definitions of them can be found in [45,46], and the larger value of these metrics, the better the performance aHMGNN achieves.

4.2. Experimental results and analysis

4.2.1. Offline test on large-scale spam detection task

In recent years, the popularity of online transaction platforms (e.g., Taobao, Jingdong and Amazon) draw growing attention of some illegitimate users [10,47] (i.e. spammers) to unfairly overwhelm legitimate users with fraudulent transactions. They generally use fake clicks, add-to-favourites, and add-to-cart to gain higher ranking without search pages. By this means, consumers will be mislead to purchase these items with high scales [24,48,49]. In fact, the bipartite grape of e-commerce spam detection scenario is a natural heterogeneous network where two types of nodes (i.e. users and items) and three types of edges (click, add-to-favourites, and add-to-cart) between them are considered. Besides, each type of node objects and edge objects both have specific attribute information. This paper models such complex data for e-commerce fraud transaction (i.e. 'add-to-cart' edge) detection, and compares it with two related deep learning frameworks [24,43]. For the performance investigation of aHMGNN, we sample 2 depth with 10 fixed-size neighbourhoods for each user node and item node. The dimension of the learned space d and the learning rate of the used DNN classifier are set as 128 and 0.01, respectively. Table 3 reports the corresponding experimental results.

From Table 3, we have the following observations: (i) Both DNN, IFCM and aHMGNN use DNN as base classifier, aHMGNN achieves a better performance. This fact shows the proposed aHMGNN can effectively integrate relations of multi-typed nodes and edges in the heterogeneous user–item bipartite graph. Besides, aHMGNN outperforms IFCM, which does not use the structural information between users and items, this observation

[1] http://jmcauley.ucsd.edu/data/amazon/.
[2] https://snap.stanford.edu/data/higgs-twitter.html.

Table 3
Results of aHMGNN on real Taobao dataset.

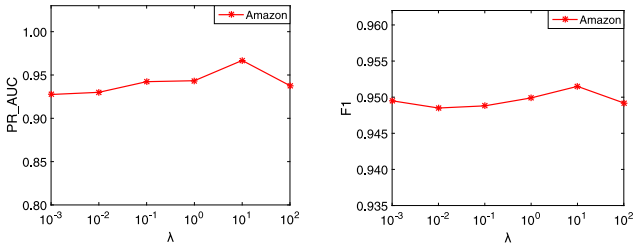
Dataset	Metrics	DNN	IFCM	**HMGNN_Nh**	**HMGNN_Nm**	**HMGNN**	**aHMGNN**
Taobao	*PR*-AUC	77.23	79.04	79.15	79.13	79.76	**80.18**
	F1-score	72.08	73.79	75.28	75.81	76.09	**76.32**

shows the importance of capturing dependence between different node objects; (ii) HMGNN_Nh, HMGNN_Nm, HMGNN are three variants of aHMGNN, where HMGNN_Nh only considers single-typed edges (i.e. add-to-cart edge) for node sampling, HMGNN_Nm splices multiple feature views of node objects and the target edge objects into a single view, and HMGNN does not use the attentions of each node/edge view for multi-view data fusion. aHMGNN is outperformed by them, which reflects the significance of aHMGNN on integrating multi-typed edge information of heterogeneous network, fusing multi-view data for link inference, and using attention mechanism, respectively.

4.2.2. Offline test on link prediction task

We conduct the presented aHMGNN for addressing the other one key problem of link inference tasks, i.e. link prediction [50], to predict the possible links by using the known network structures. Two public heterogeneous networks and the recently proposed homogeneous network embedding method ANRL [16] and two heterogeneous network embedding algorithms (MNE [28], GATNE [13](GATNE_T, GATNE_I) are adopted for the predictions of multi-typed edges to evaluate the performance of aHMGNN. Furthermore, to better evaluate the attention contribution of our introduced aHMGNN, we also extend DGMI [30], an attention-based heterogeneous node embedding approach, to link inference task by concatenating the features of two nodes as inputs and then adopt Logistic Regression model for edge predictions. The following shows the parameters setting of aHMGNN on this task: 2 depth 5 fixed-size neighbourhoods are sampled for the single-typed nodes of both Amazon and Twitter datasets. Notice that both the acquired two heterogeneous datasets only consist of one node type, thus the local node sets we sampled for each node by aHMGNN in each depth only contains homogeneous rather than heterogeneous neighbourhoods. The dimension of the shared subspace d is set as 128, and the learning rate of DNN classifier on Amazon and Twitter datasets are set as 0.015 and 0.001, respectively. Table 4 records the averaged predicted results of multi-typed edges on these two benchmark datasets.

From Table 4, we can see that: (i) aHMGNN outperforms the related works ANRL, MNE, GATNE_T and GATNE_I, and this fact indicates the effectiveness of aHMGNN on integrating heterogeneous graphs with multi-typed nodes and edges. In contrast, ANRL can only handle homogeneous networks and it achieves the lowest performance. This observation demonstrates the heterogeneity of graphs can provide more information for node embeddings learning. Besides, both GATNE and aHMGNN are heterogeneous approaches which consider the commonality and individualities of multi-typed edges, and adopt self-attention mechanism for node representations fusion. GATNE loses to aHMGNN, and this fact indirectly shows the effectiveness of data fusion designment of our proposed model; (ii) HMGNN_Nm simply splices multiple node feature views and the target edge feature view into a single one, and aHMGNN outperforms it, this observation again proves the effectiveness of multi-view generator designment of aHMGNN; (iii) Besides, we also observe that aHMGNN achieves a better performance than HMGNN, which ignores the contributions specific to each node and the target edge representations for multi-view data fusion. This phenomenon demonstrates that using the attention mechanism can indeed help the model boosting of aHMGNN; (iv) Both aHMGNN and DGMI adopt attention mechanism for a better model performance and DGMI is outperformed by aHMGNN. This fact reflects the superiority of our model on disposing multi-typed nodes and edges.



(a) Sensitivity of λ (b) Sensitivity of λ

Fig. 4. *PR-AUC*, *F1-score* of aHMGNN on Amazon dataset.

4.2.3. Parameter sensitivity

In this paper, one important parameter is used for the aHMGNN model, i.e. the trade-off parameter λ. To explore its sensitivity, we range it from different values of [0.001, 0.01, 0.1, 1, 10, 100, 1000] and report *RR-AUC* and *F1-score* of aHMGNN on Amazon dataset in Fig. 4:

From Fig. 4, we can see that as λ increases, the performance of aHMGNN firstly trends to rise, and it then reduces. An overall good performance can be achieved when $\lambda = 10$. A too small or too large λ value causes a low performance of aHMGNN, the possible reason of such phenomenon is that a small or large λ value will directly affect DNN input features obtained by the fusion of multi-typed node and edge representations, thus result in a low model performance. All these observations show the importance of a reasonable λ for aHMGNN.

4.2.4. Running environment

This paper conducts both offline and online experiments on alibaba's distributed cloud compute platform, which consists of hundreds of workers and each of them assigns one CPU core with 10G memory. The code of our proposed aHMGNN is implemented with tensorflow 1.4 in python 2.7. Specifically, for online transactions detection, we use eight F53 machines with 512G memory to update graph and sample node dynamically, and 200 workers are utilized for the real-time edge predictions.

4.2.5. Feature selection

This paper applies aHMGNN for two different link inference tasks, including link prediction task which is tested on two open heterogeneous networks, and spam detection (i.e. edge prediction) scenario on the real Taobao data. For link prediction application, we directly use the node features given in the obtained benchmark datasets [13]. In spam detection experiment, we select features of user nodes, item nodes, and the 'add-to-cart' edges by combining experiences of expert and statistical methods such as Pearson Correlation Coefficient as model inputs.

4.2.6. Scalability analysis

To further investigate the scalability of our presented aHMGNN, we statistic the running time of our model under different number of workers and multiple scale of data sizes on Taobao dataset, Fig. 5 and Table 5 show the recorded results, respectively.

From Fig. 5 and Table 5, we have two observations: (i) The time cost (including node sampling, model training and testing processes) decreases significantly as the number of worker increases, and aHMGNN converges when the work number is close

Table 4
Results of aHMGNN on different heterogeneous network datasets.

Datasets	Methods	*PR*-AUC	F1-score	Datasets	Methods	*PR*-AUC	F1-score
	ANRL	71.68	67.72		ANRL	70.04	64.6
	MNE	90.28	83.25		MNE	91.37	84.32
	DGMI	74.02	65.52		DGMI	75.00	55.62
	GATNE_T	**97.44**	92.87		GATNE_T	92.30	84.69
Amazon	GATNE_I	96.25	91.36	Twitter	GATNE_I	92.04	84.38
	HMGNN(Nh)	79.10	81.10		**HMGNN(Nh)**	89.21	92.06
	HMGNN(Nm)	78.50	73.48		**HMGNN(Nm)**	79.86	81.73
	HMGNN	97.44	95.35		**HMGNN**	93.38	92.83
	aHMGNN	**97.75**	**95.59**		**aHMGNN**	**93.67**	**93.47**

Table 5
The running time of aHMGNN under different data sizes on Taobao dataset.

Data size	10,000	100,000	1,000,000	10,000,000	100,000,000
Running time (s)	505.668	735.292	887.505	1016.548	4047.940

Fig. 5. The scalability of aHMGNN under different number of workers on Taobao dataset.

to 100. This fact shows that our approach is quite scalable on the distributed cloud compute platform; (ii) When the data size is added up at a rate of 100 times, the speedup of running time is much lower than the data growth ratio. This nonlinear growth rate further indicates the advantage of our model on handle large-scale tasks. Based on these analysis, we can conclude that our aHMGNN is scalable enough to be applied in practice.

4.2.7. Time complexity

The time complexity of aHMGNN are composed of two parts. In the first stage, we use the extended graphsage method for the embeddings learning of each type of nodes. The time complexity of this stage is $O(\prod_{k=1}^{K}|Nei(k)|)$, where R and T represent the number of node and edge types, respectively, and we sample fixed-size, i.e. $|Nei(k)|$, K depth nodes as neighbourhoods and aggregate their features to obtain node embedded representation. The second stage targets to fuse multi-typed nodes and edges for edge prediction. Suppose t is the number of maximum iterations for multi-view subspace learning, the time complexity of learning parameter matrices $\{\mathbf{P}^v\}_{v=1}^{V}$ and $\mathbf{C}$ is $O(tVnd^2) + O(tndk)$, respectively. To sum up, the overall time complexity of our model is $O(\prod_{k=1}^{K}|Nei(k)|) + O(tVnd2) + O(tndk)$.

5. Online system implementation

This section mainly introduces online system implementation of the proposed aHMGNN for e-commerce real-time fraud transaction detection application [49,51,52]. Before detailing the specific system deployment, we first give an overview of the system and then elaborate on our presented technique modules. Fig. 6 shows the workflow of the spam detection platform at Taobao as follows:

- When an user places an order on Taobao App, the corresponding logs of this order will be submitted to the Blink (Online trading platform of Taobao), to generate a new transaction edge between this user and item;
- Graph Engine (A large-scale distributed computing engine, which is responsible for loading offline historical transactions and heterogeneous graphs initialization) receives the new edge record sent by Blink, it then updates the user–item bipartite graph in real time, and dynamically sampling neighbourhoods of the node w.r.t this target edge;
- After completing the graph update and dynamic node neighbourhood sampling, the presented aHMGNN algorithm, which is distributed Tensorflow platform, will be exploited to detect whether this transaction is a fraud one or not;
- Finally, the detected fraud transactions will be used for online data real-time cleaning, and meanwhile, the Risk Management Platform will manage the corresponding merchants [51] and feed back the results to aHMGNN for further model iteration.

5.1. Online graph data capture

To update the real-time acquired data in user–item bipartite graph for online spam detection system implementation, this paper develops a variable graph data storage (VGDS) method and dynamic node neighbourhood sampling (DNNS) strategy on graph engine to address this problem. The following contents describe these two techniques.

5.1.1. Variable graph data storage

Traditional GNN algorithms generally assume single formed data structure for graph storage, such as adjacency matrix, adjacency array and adjacency list. Most of them generally use historical graph data for model training. To efficiently load the real-time added nodes and edges for online system implementation, this paper introduces a novel variable graph data storage method as shown in Fig. 7.

Firstly, we use static graph $\mathcal{G}_1$ to collect the offline historical transactions, and use dynamic graph $\mathcal{G}_2$ to capture the real-time data. Considering different characteristics of these two graphs, e.g., $\mathcal{G}_1$ has a larger data size and a more stable network structure compared to $\mathcal{G}_2$, this paper adopts adjacency array and adjacency list for offline and online transactions data storage, respectively. Next, for each newly added edge (i.e. transaction), we query corresponding information from the static graph $\mathcal{G}_1$ and the dynamic graph $\mathcal{G}_2$ in parallel. If the edge is already existed in the original static/dynamic graph, we then directly update the feature information of this edge and the connected user node and item node. Constantly, if there is no such edge in these two graphs, we will add this new node and edge inputs to $\mathcal{G}_2$. Let Nei_1 and Nei_2 be the neighbourhoods number of the nodes linked by the newly added edges in $\mathcal{G}_1$ and $\mathcal{G}_2$ within a certain time stamp, respectively. The node search efficiency in $\mathcal{G}_1$ and $\mathcal{G}_2$ are $O(log(Nei_1))$ and $O(Nei_2)$, respectively. Therefore, the overall efficiency of real-time graph data update is $O(log(Nei_1)) + O(Nei_2)$, where $Nei_2 \ll Nei_1$.

To further save the time costs of graph data query, this paper also exploits a rebuild component on graph engine for regularly updating the structure of $\mathcal{G}_1$. Specifically, at certain intervals, we

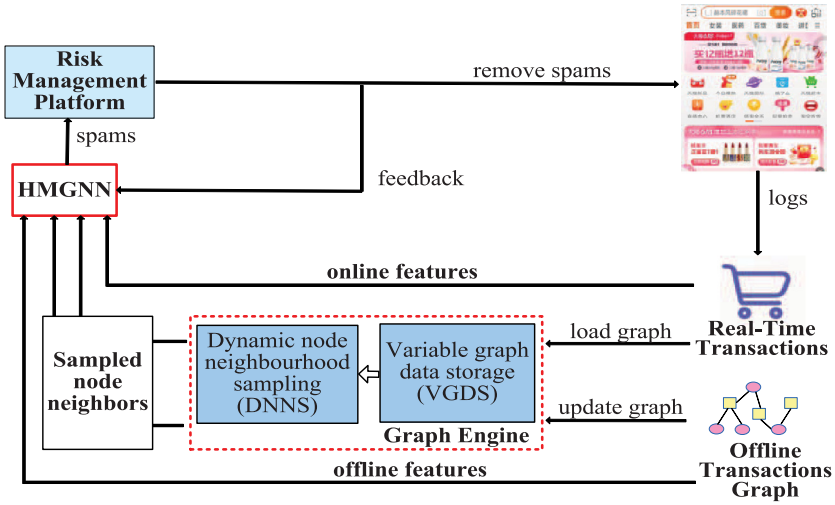


Fig. 6. The workflow of the spam detection platform at Taobao.

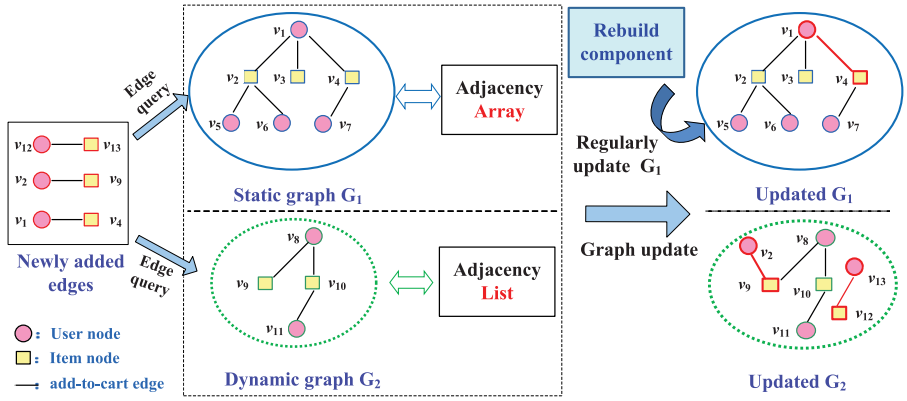


Fig. 7. The handling process of Various Graph Data Storage module.

update the dynamic graph $\mathcal{G}_2$ data to the static graph $\mathcal{G}_1$, and clear $\mathcal{G}_2$ to ensure that $Nei_2 \ll Nei_1$. Due to the fact that the operation on the rebuild component is parallel to the online real-time $\mathcal{G}_2$ updating, thus the current graph data loading efficiency will not be greatly affected.

By this means, we can update the real-time evolving network inputs by adopting our designed variable graph structure storage module efficiently.

5.1.2. Dynamic node neighbourhood sampling

After the real-time transactions storage, it is needed to update the neighbour information of nodes since the appearance of the newly-added node and edge inputs. A simple solution is to traverse all the neighbourhoods of target nodes, however, it will cause time waste of node searching especially in the case of millions of graph nodes. To improve the efficiency of neighbourhood nodes update, this paper develops a novel dynamic node neighbourhood sampling (DNNS) strategy based on Cumulative Distribution Function (CDF). The introduced DNNS approach are based on the node sampling method for the historical loaded data as we introduced in Section 3.2.1, which can dynamically capture the temporal neighbour information of target nodes.

In DNNS module, we first assign an accumulate array for each node w.r.t $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. For a certain node, we denote $acc1$ and $acc2$ be two assigned accumulate arrays of this node w.r.t static graph $\mathcal{G}_1$ and dynamic graph $\mathcal{G}_2$, respectively. Meanwhile, we also maintain two variables b_1 and b_2 recording the weight of neighbourhoods of the target node for $acc1$ and $acc2$, respectively. The formulation of these two accumulate arrays adopt the same definition as follows:

$$\begin{cases} acc_{ij} = b_i, f_i(j), & \text{if } j = 1 \\ acc_{ij} = acc_{ij-1} + b_i, f_i(j), & \text{if } j > 1 \end{cases} \tag{7}$$

where $f_i(j)$ is an efficient index, which returns the jth element node ID of the ith node. $b_i, f_i(j)$ records the sampling weight between the ith node and its neighbourhood node $f_i(j)$. Thus for a target node which connects to the newly-added edges, if VGDS module shows it should be added to $\mathcal{G}_1$, we will update the acc1 array; otherwise, the acc2 array is updated. To this end, we can capture the dynamic neighbourhood information for the target node inputs by continuously updating the weight lists in the accumulate arrays of them. After that, during the node sampling process, we generate a random number $r \in [0, b](b = b_1 + b_2)$ at each iteration and uses the binary search method to find the location of this value. If the condition of r satisfies $r < b_1$, we then return the sampled results based on acc1; otherwise, the neighbourhood nodes are sampled based on acc2. The time complexity of dynamically sampling neighbourhoods for the newly added nodes are $O(log(Nei_1))$ and $O(log(Nei_2))$, respectively.

Generally, since that most of the newly added edges trend to be updated to $\mathcal{G}_2$ instead of $\mathcal{G}_1$, thus we usually update the neighbourhood sampling information of target nodes in $\mathcal{G}_2$, and the time complexity of this process is $O(Nei_2)$ where $Nei_2 \ll Nei_1$. Compared to the solution of traversing all the neighbours, adopting our sampling strategy can greatly reduce the time complexity of dynamic node sampling.

Table 6
Comparison of different graph data storage methods. (**Aa**: Adjacency array; **Al**: Adjacency list; **VGDS**: Variable Graph Data Storage (no rebuild components using); **VGDS (rebuild)**: Variable Graph Data Storage (considering rebuild components).

Aa	Al	VGDS	VGDS (rebuild)
4k/s	3k/s	200k/s	210k/s

Table 7
Comparison of different node neighbourhoods sampling strategies.

Traverse all neighbours	CDF	DNNS
6k/s	160k/s	250k/s

5.2. Performance of aHMGNN on online spam detection application

To validate the effectiveness of our proposed VGDS and DNNS modules on online spam detection task, this paper simultaneously conducts aHMGNN on acquired history data stored in $\mathcal{G}_1$ and the real-time transaction data in $\mathcal{G}_2$. Considering the large scale of $\mathcal{G}_1$ includes 3.57 billion user and item nodes and 0.595 billion transaction edges, and the real-time data of $\mathcal{G}_2$ considering 1.71 billion nodes and 0.284 billion edges, Tables 6 and 7 report the comparing performance of our designed two modules, respectively. Notice that k/s represents 1000/s, the meaning of this unit is the speed of processing requests. For example, 5k/s means disposing 5000 requests of graph update/node sampling per second.

From Table 6, we can conclude that: Under the same data size and experimental setup conditions, our introduced VGDS method outperforms other traditional graph data storage structures, especially compared to adjacency array and adjacency list strategies. The efficiency of our module designment is improved by 50 times than simply using these two storage structures. Besides, we also observe that considering the rebuild components can further improve the efficiency of dynamic graph updating. By using them, we can not only separate historical and real-time transactions, but also avoid the shortcomings of slow update of adjacency arrays and adjacency lists caused by the appearance of hot-keys in the heterogeneous graph (nodes which degrees exceed one million) to ensure the overall graph updating efficiency.

Table 7 shows the experimental results of our presented DNNS method and several representative sampling approaches. From this table, we have an observation that adopting our developed dynamic node neighbourhood sampling module can improve the algorithm efficiency up to 26.7 times than traversing all neighbours. In addition, compared to the original CDF algorithm, there is 1.44 times improvement by using our sampling design. These facts both demonstrate the effectiveness and efficiency of our dynamic sampling designment, and further improve the overall efficiency for online fraud transaction detection application.

6. Conclusions

In this paper, we present a new idea named aHMGNN which models a more intricate network structure considering multi-typed nodes and edges as well as attributes of these objects, and extend our preliminary research [53]. Two stages are designed in this approach where the first stage exploits the complicated network structure for heterogeneous node embeddings learning. The second stage proposes an attention-based multi-view generator for edge predictions by aggregating multiple node and edge representations. Besides, we also introduce a variable graph data storage method and dynamic node neighbourhood sampling strategy for online system implementation of our approach. Experimental offline and online studies on both public and real-world large-scale Taobao datasets show the good scalability, high efficiency and effectiveness of our model. In the future, we are looking forward to developing our algorithm for a more accurate recall score and dynamic node sampling strategy, as well as exploring the collective fraud detection by combining aHMGNN and several machine learning technologies.

CRediT authorship contribution statement

Zhao Li: Investigation, Conceptualization, Writing - original draft, Methodology, Validation. **Yuying Xing:** Data curation, Writing - original draft, Methodology, Validation. **Jiaming Huang:** Investigation, Writing - original draft, Methodology, Validation. **Haobo Wang:** Investigation, Writing - review & editing. **Jianliang Gao:** Writing - review & editing. **Guoxian Yu:** Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is sponsored by Zhejiang Lab, China (No. 2019KE0AB01).

References

[1] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: IJCNN, Vol. 2, 2005, pp. 729–734.
[2] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[3] C. Gallicchio, A. Micheli, Graph echo state networks, in: IJCNN, 2010, pp. 1–8.
[4] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, IEEE Trans. Cybern. (2019) 1–13.
[5] X. Shen, S. Pan, W. Liu, Y. Ong, Q. Sun, Discrete network embedding, in: IJCAI, 2018, pp. 3549–3555.
[6] M. Zhang, Y. Chen, Link prediction based on graph neural networks, in: NIPS, 2018, pp. 5165–5175.
[7] A. Pucci, M. Gori, M. Hagenbuchner, F. Scarselli, A. Tsoi, Investigation into the application of graph neural networks to large-scale recommender systems, Syst. Sci. 32 (4) (2006) 17–26.
[8] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: AAAI, Vol. 33, 2019, pp. 346–353.
[9] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, L. Song, Heterogeneous graph neural networks for malicious account detection, in: CIKM, 2018, pp. 2077–2085.
[10] C. Li, S. Wang, L. He, S.Y. Philip, Y. Liang, Z. Li, Ssdmv: Semi-supervised deep social spammer detection by multi-view data fusion, in: ICDM, 2018, pp. 247–256.
[11] M. Muzammal, F. Abbasi, Q. Qu, R. Talat, J. Fan, A decentralised approach for link inference in large signed graphs, FGCS 102 (2020) 827–837.
[12] J. Li, T. Cai, K. Deng, X. Wang, T. Sellis, F. Xia, Community-diversified influence maximization in social networks, Inf. Syst. 92 (2020) 1–12.
[13] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, J. Tang, Representation learning for attributed multiplex heterogeneous network, in: KDD, 2019, pp. 1358–1368.
[14] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, 2019, arXiv preprint arXiv:1901.00596.
[15] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NIPS, 2017, pp. 1024–1034.
[16] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, C. Wang, ANRL: Attributed network representation learning via deep neural networks, in: IJCAI, Vol. 18, 2018, pp. 3155–3161.
[17] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: WWW, 2019, pp. 2022–2032.
[18] X. Fu, J. Zhang, Z. Meng, I. King, MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding, in: WWW, 2020, pp. 2331–2341.

[19] C. Zhang, D. Song, C. Huang, A. Swami, N.V. Chawla, Heterogeneous graph neural network, in: KDD, 2019, pp. 793–803.
[20] H. Weng, S. Ji, F. Duan, Z. Li, J. Chen, Q. He, T. Wang, CATS: Cross-platform e-commerce fraud detection, in: ICDE, 2019, pp. 1874–1885.
[21] H. Wang, Z. Li, J. Huang, P. Hui, W. Liu, T. Hu, G. Chen, Collaboration based multi-label propagation for fraud detection, in: IJCAI, 2020.
[22] S. Aridhi, J.A.F.d. Macêdo, E.M. Nguifo, K. Zeitouni, Special issue on "advances on large evolving graphs", FGCS 110 (2020) 310.
[23] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: KDD, 2018, pp. 974–983.
[24] Q. Guo, Z. Li, B. An, P. Hui, J. Huang, L. Zhang, M. Zhao, Securing the deep fraud detector in large-scale e-commerce platform via adversarial machine learning approach, in: WWW, 2019, pp. 616–626.
[25] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[26] T. Fu, W. Lee, Z. Lei, Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning, in: KDD, 2019, pp. 2478–2486.
[27] Y. Dong, N.V. Chawla, A. Swami, Metapath2vec: Scalable representation learning for heterogeneous networks, in: KDD, 2017, pp. 135–144.
[28] H. Zhang, L. Qiu, L. Yi, Y. Song, Scalable multiplex network embedding., in: IJCAI, Vol. 18, 2018, pp. 3082–3088.
[29] J. Gao, T. yu, F. Xiong, J. Wang, Z. Li, MGNN: A multimodal graph neural network for predicting the survival of cancer patients, in: SIGIR, 2020.
[30] C. Park, D. Kim, J. Han, H. Yu, Unsupervised attributed multiplex network embedding, in: AAAI, 2020, pp. 5371–5378.
[31] Z. Li, Z. Liu, J. Huang, G. Tang, Y. Duan, Z. Zhang, Y. Yang, MV-GCN: Multi-view graph convolutional networks for link prediction, IEEE Access 7 (2019) 176317–176328.
[32] L. Zheng, Z. Li, J. Li, Z. Li, J. Gao, AddGraph: Anomaly detection in dynamic graph using attention-based temporal GCN, in: IJCAI, 2019, pp. 4419–4425.
[33] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: KDD, 2016, pp. 855–864.
[34] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: KDD, 2016, pp. 1225–1234.
[35] Y. Ma, Z. Guo, Z. Ren, E. Zhao, J. Tang, D. Yin, Streaming graph neural networks, 2018, arXiv preprint arXiv:1810.10627.
[36] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, Dynamic graph representation learning via self-attention networks, 2018, arXiv preprint arXiv:1812.09430.
[37] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T.B. Schardl, C.E. Leiserson, EvolveGCN: Evolving graph convolutional networks for dynamic graphs, 2020, arXiv preprint arXiv:1902.10191.
[38] S. Wang, Z. Chen, X. Yu, D. Li, J. Ni, L. Tang, J. Gui, Z. Li, H. Chen, P.S. Yu, Heterogeneous graph matching networks, in: IJCAI, 2019, pp. 3762–3770.
[39] Y. Gao, X. Wu, W. Yan, L. Zhang, T. Wu, Dynamic network embedding enhanced advisor–advisee relationship identification based on internet of scholars, GGCS 108 (2020) 677–686.
[40] J. Mondal, A. Deshpande, Managing large dynamic graphs efficiently, in: SIGMOD, 2012, pp. 145–156.
[41] Z. Zhang, M. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in: NIPS, 2018, pp. 8778–8788.
[42] Z. Lin, M. Feng, C.N.d. Santos, M. Yu, B. Xiang, B. Zhou, Y. Bengio, A structured self-attentive sentence embedding, 2017, arXiv preprint arXiv:1703.03130.
[43] B. Kim, J. Kim, H. Chae, D. Yoon, J.W. Choi, Deep neural network-based automatic modulation classification technique, in: ICTC, 2016, pp. 579–582.
[44] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[45] J. Davis, M. Goadrich, The relationship between precision-recall and ROC curves, in: ICML, 2006, pp. 233–240.
[46] O.O. Koyejo, N. Natarajan, P.K. Ravikumar, I.S. Dhillon, Consistent binary classification with generalized performance metrics, in: NIPS, 2014, pp. 2744–2752.
[47] Z. Li, J. Song, S. Hu, S. Ruan, L. Zhang, Z. Hu, J. Gao, FAIR: Fraud aware impression regulation system in large-scale real-time e-commerce search platform, in: ICDE, 2019, pp. 1898–1903.
[48] N. Su, Y. Liu, Z. Li, Y. Liu, M. Zhang, S. Ma, Detecting crowdturfing add to favorites activities in online shopping, in: WWW, 2018, pp. 1673–1682.
[49] H. Weng, Z. Li, S. Ji, C. Chu, H. Lu, T. Du, Q. He, Online e-commerce fraud: A large-scale detection and analysis, in: ICDE, 2018, pp. 1435–1440.
[50] L. Lü, T. Zhou, Link prediction in complex networks: A survey, Phys. A 390 (6) (2011) 1150–1170.
[51] M. Zhao, Z. Li, B. An, H. Lu, Y. Yang, C. Chu, Impression allocation for combating fraud in e-commerce via deep reinforcement learning with action norm penalty, in: IJCAI, 2018, pp. 3940–3946.
[52] A. Li, Z. Qin, R. Liu, Y. Yang, D. Li, Spam review detection with graph convolutional networks, 2019, arXiv preprint arXiv:1908.10679.
[53] Y. Xing, Z. Li, P. Hui, J. Huang, X. Chen, L. Zhang, G. Yu, Link inference via heterogeneous multi-view graph neural networks, in: DASFAA, 2020.

Zhao Li completed the Ph.D. degree with a Graduate Award from the Computer Science Department, University of Vermont. He is currently a Senior Staff Scientist with the Alibaba Group, specializing in ecommerce ranking and recommendation systems. He has published over 50 papers in prestigious conferences and journals, including NIPS, AAAI, IJCAI, and KDD. His current research interests include adversarial machine learning, network representation learning, knowledge graphs, multi-agent reinforcement learning, and big data-driven security. He is also a Technical Committee Member of the China Computer Federation on Database. He received a Fellowship from NSF EPSCoR.

Yuying Xing received the B.S. degree in Computer and Information Science from Southwest University, Chongqing, China. She received B.Sc. in Computer Science & Engineering from Northwest Normal University, Lanzhou, China in 2017. Her research interests include machine learning and data mining, especially on multi-view and multi-label learning.

Jiaming Huang received the B.S. degree and M.S. degree in Computer Science and Technology from Guangdong University of Technology, Guangzhou, China, in 2015 and 2018, respectively. He is currently working at Alibaba as an Algorithm Engineer. His research interests include graph neural network, abnormal detection, and machine learning.

Haobo Wang received the B.S. degree in computer science from Zhejiang University, Hangzhou, China, in 2018. He is currently working toward the Ph.D. degree in the College of Computer Science, Zhejiang University, China. His research interests include machine learning and data mining, especially on multi-label learning and weakly-supervised learning.

Jianliang Gao received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, China. He is currently a professor with the School of Computer Science and Engineering, Central South University, China. His main research interests include big data processing, machine learning and graph data mining.

Guoxian Yu is a Professor at the School of Software, Shandong University, Jinan, China. He received the Ph.D. in Computer Science from South China University of Technology, Guangzhou, China in 2013. His research interests include data mining and bioinformatics. He has served as (senior) PC member for KDD, NeurIPS, AAAI, IJCAI, ICDM, ECML/PKDD and reviewer for IEEE/ACM Trans. journals.