World Wide Web (2024) 27:35
https://doi.org/10.1007/s11280-024-01275-2

A heterogeneous graph-based semi-supervised learning framework for access control decision-making

Jiao Yin[1,2] · Guihong Chen[3,4] · Wei Hong[2] · Jinli Cao[1] · Hua Wang[2] · Yuan Miao[2]

Received: 17 March 2024 / Revised: 1 May 2024 / Accepted: 9 May 2024 /
Published online: 24 May 2024

Abstract
For modern information systems, robust access control mechanisms are vital in safeguarding data integrity and ensuring the entire system's security. This paper proposes a novel semi-supervised learning framework that leverages heterogeneous graph neural network-based embedding to encapsulate both the intricate relationships within the organizational structure and interactions between users and resources. Unlike existing methods focusing solely on individual user and resource attributes, our approach embeds organizational and operational interrelationships into the hidden layer node embeddings. These embeddings are learned from a self-supervised link prediction task based on a constructed access control heterogeneous graph via a heterogeneous graph neural network. Subsequently, the learned node embeddings, along with the original node features, serve as inputs for a supervised access control decision-making task, facilitating the construction of a machine-learning access control model. Experimental results on the open-sourced Amazon access control dataset demonstrate that our proposed framework outperforms models using original or manually extracted graph-based features from previous works. The prepossessed data and codes are available on GitHub,facilitating reproducibility and further research endeavors.

Keywords Access control · Semi-supervised learning · Heterogeneous graph · Node embedding · Link prediction

1 Introduction

In the contemporary era of rapid technological progress, organizations and individuals enjoy notable benefits in terms of enhanced convenience and productivity [1]. However, technological advancement also brings forth concerns, particularly as the volume of sensitive data and system complexity increase, prompting a growing awareness and emphasis on data privacy issues and system security protection [2–4]. Access control serves as the first line of safeguard, mitigating the risk of unauthorized resource access or data breaches [5–8]. In an era where information is a valuable asset, effective access control strategies contribute significantly to organizations' overall security posture, fostering trust among stakeholders and ensuring compliance with regulatory requirements [9–11].

Extended author information available on the last page of the article

Traditional role-based access control (RBAC) strategies solely assign resource access permissions to users based on their roles, which suffer from limited context awareness and lack of granularity [12, 13]. Consequently, RBAC often grants users more data or resource access than necessary. On the other hand, attribute-based access control (ABAC) strategies make authorization decisions based on attributes or characteristics of users, resources, and even system environments rather than relying solely on roles [14]. While ABAC strategies offer more fine-grained and flexible access control policies than RBAC, they still face challenges due to the increasing complexity in design and implementation as the scale of users and attributes grows [15, 16].

Some scholars have attempted to develop machine learning (ML) and deep learning (DL) models for various applications including security [17, 18], data quality [19–22], health informatics [23–25] and access control decision-making to enhance efficiency and adaptability to concept drifts. While partial verification of their efficiency and adaptive capabilities has been achieved, addressing the explainability and reliability of ML/DL methods remains essential. With recent advancements in knowledge graphs (KGs), graph theory, and graph neural networks (GNNs) [26], more scholars are turning to graph-based methods to improve the efficiency, performance, explainability, and reliability of access control decision-making. For instance, Morgado, C., Baioco, GB., Basso, T., et al. proposed a security model to provide access control for NoSQL graph-oriented database management systems, preserving data integrity and protecting against unauthorized access [27]. Shan, D., Du, X., Wang, W., et al. introduced a critical provenance identification framework based on heterogeneous graph neural networks (HGNNs) to address dynamic attribute generation and multi-source aggregation challenges arising from big data resources in dynamic access control scenarios [28]. Specifically, Mingshan, Y. et al. devised an algorithm to construct an access control KG from user and resource attributes, then extracted topological features from the constructed KG to represent high cardinality categorical user and resource attributes for building ML-based access control models [29].

Despite the aforementioned progress, the unavailability of data and codes hinders reproducibility and comparison with traditional ML/DL models [16, 30–32]. Furthermore, existing literature lacks discussions on the impact of different relationship types on access control decision-making performance [33–36]. This paper aims to explore the capability of HGNNs in integrating multi-source and multi-relationship data from large-scale information systems comprising tens of thousands of users and resources. Specifically, we propose a semi-supervised learning framework based on an access control heterogeneous graph (ACHG). Firstly, we employ a self-supervised node embedding strategy based on an HGNN link prediction task to learn node embeddings of users and resources. Subsequently, a supervised ML model is trained as the classifier to make access control decisions, utilizing learned node embeddings and original user and resource attributes.

The contributions of this paper are threefold:

(1) We introduce a comprehensive HGNN-based semi-supervised learning framework for access control decision-making. This framework utilizes a self-supervised node embedding strategy to learn node embeddings from an ACHG. Subsequently, a supervised ML model is trained from access control log files by integrating node embeddings and original features of users and resources as the features of access requests.
(2) We conduct empirical research to explore the impact of different relationship types and node embedding lengths of heterogeneous graphs on access control performance. Our investigations validate insights from existing literature regarding the influence of heterogeneity and node embedding complexity on downstream task performance. These

findings offer valuable insights for designing and implementing future heterogeneous graph-based applications, including access control decision-making.

(3) We validate the effectiveness of node embeddings learned from an end-to-end HGNN-based self-supervised link prediction task in enhancing access control decision-making performance. This is demonstrated through experiments conducted on an open-sourced Amazon access control dataset [37]. We share our data and codes on GitHub to facilitate reproducibility and inspire future research endeavors.

We organize the rest of the paper as follows. Section 2 provides essential definitions, concepts, and mathematical notions for the subsequent sections, including heterogeneous graph, meta-path, node embedding, and link prediction, establishing a foundation for the remainder of the paper. Section 3 outlines the components and workflow of the proposed semi-supervised learning framework. Dataset introduction, experimental settings, evaluation metrics, experimental design, results, and discussion are presented in Section 4, followed by a conclusion in Section 5.

2 Preliminaries

Definition 1 (Heterogeneous graph) *A heterogeneous graph is defined as $G = \{\mathcal{V}, \mathcal{E}, \mathbf{T^V}, \mathbf{T^E}, \mathbf{X^V}, \mathbf{X^E}\}$, where $\mathcal{V}$ and $\mathcal{E}$ denote the node and edge sets respectively, $\mathbf{T^V}$ and $\mathbf{T^E}$ are the node type and edge type sets accordingly, and $\mathbf{X^V}$ and $\mathbf{X^E}$ are the original attribute (feature) matrices of nodes in $\mathcal{V}$ and edges in $\mathcal{E}$. Each node $v \in \mathcal{V}$ belongs to a node type $T^v \in \mathbf{T^V}$ and has a feature $x_v \in X^{T^v} \in \mathbf{X^V}$, where X^{T^v} is the feature matrix of all nodes belongs to node type T^v. Similarly, each edge $e_{vu} \in \mathcal{E}$ belongs to an edge type $T^e \in \mathbf{T^E}$ and has a feature $x_e \in X^{T^e} \in \mathbf{X^E}$, where $v \in \mathcal{V}$ is the edge head, $u \in \mathcal{V}$ is the edge tail, and X^{T^e} is the feature matrix of all edges belongs to edge type T^e.*

The edges within a heterogeneous graph can be described by a series of adjacency matrices, $\mathbf{A^E}$. Specifically, for an edge type (relationship) $T^e \in \mathbf{T}^{\mathcal{E}}$, $A^{T^vT^u} \in \mathbf{A^E}$ is the corresponding adjacency matrix from node type T^v to T^u, whose dimension can be described as $A^{T^vT^u} \in \mathbb{R}^{|V|\times|U|}$, where $|V|$ and $|U|$ are the total number of nodes belongs to node types T^v and T^u, respectively.

Given a heterogeneous graph, $G = \{\mathcal{V}, \mathcal{E}, \mathbf{T^V}, \mathbf{T^E}, \mathbf{X^V}, \mathbf{X^E}\}$, if $|\mathbf{T^V}| = |\mathbf{T^E}| = 1$, $|\mathbf{X^V}| \leq 1$ and $|\mathbf{X^E}| \leq 1$, it means that G contains only one node type, one relationship type, and at most one node feature matrix and one relationship matrix. In this case, it degrades into a homogeneous graph.

Example. An ACHG is illustrated in Figure 1 (b), which contains five types of nodes: user, resource, department, title, and manager (Figure 1 (a)). These nodes are connected by different types of edges, such as the has_manager relationship between node types user and manager and the has_department relationship between node types user and department.

Definition 2 (Meta-path) *A meta-path is the path with a predefined node-edge type pattern, denoted as $\mathcal{P} \triangleq T_1^v \xrightarrow{T_1^e} T_2^v \xrightarrow{T_2^e} \cdots \xrightarrow{T_{l-1}^e} T_l^v$, where $\{T_1^v, T_2^v, \cdots, T_l^v\} \in \mathbf{T^V}$ and $\{T_1^e, T_2^e, \cdots, T_{l-1}^e\} \in \mathbf{T^E}$. The total number of nodes in a meta-path is l, and the total number of edges in a meta-path is $l-1$. For convenience, the edge types can be omitted when describing a meta-path, i.e., $\mathcal{P} = T_1^v \to T_2^v \to \cdots \to T_l^v$. Given a meta-path $\mathcal{P}$, the paths that follow the pattern of $\mathcal{P}$ are the instances of the meta-path.*

Example. As demonstrated in Figure 1 (c), we list four instances of four different meta-paths: user-resource-user, user-department-user, user-title-user, and user-manager-user,

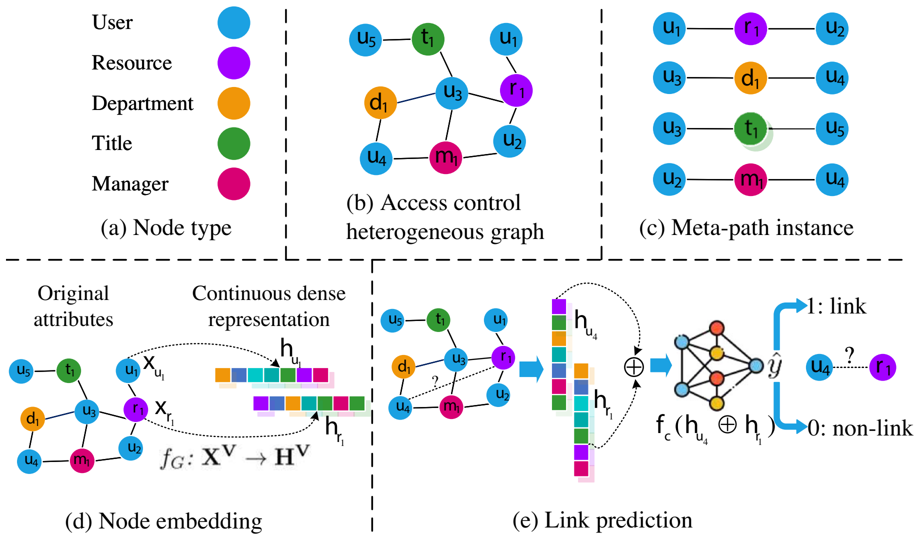

Figure 1 An illustrative example of an undirected ACHG. (a) Five types of nodes (i.e., user, resource, department, title, manager). (b) An ACHG consists of five types of nodes and four types of edges. (c) Four instances of meta-paths in ACHG (i.e., user-resource-user, user-department-user, user-title-user, user-manager-user). (d) An illustration of node embedding: mapping the original attributes of nodes into continuous dense representations. (e) An illustration of predicting whether a link exists between user node u_4 and resource node r_1

omitting the edge types and directions. Meta-paths convey semantic meanings hidden in a heterogeneous graph. For example, the meta-path user-department-user indicates that two users belong to the same department, which is crucial background knowledge for access control scenarios. In the corresponding instance, u_3 - d_1 - u_4, the user u_3's granted access to certain resources may be very similar to u_4's.

Definition 3 (Node embedding) *Given a heterogeneous graph, $G = \{\mathcal{V}, \mathcal{E}, \mathbf{T^V}, \mathbf{T^E}, \mathbf{X^V}, \mathbf{X^E}\}$, node embedding refers to the process of representing the original attributes of nodes, $\mathbf{X^V}$, into continuous dense representation matrices, $\mathbf{H^V}$, incorporating with edge and edge attribute information. The mapping relationships between $\mathbf{X^V}$ and $\mathbf{H^V}$ can be described as $f_G: \mathbf{X^V} \rightarrow \mathbf{H^V}$.*

Example. We demonstrate an ACHG node embedding process in Figure 1 (d). Given that all nodes belonging to the node type user have original attributes, denoted as $X^{T^u} \in \mathbf{X^V}$. For a specific user u_1, the original attribute is denoted as $x_{u_1} \in X^{T^u}$. Similarly, for all resource nodes, the original attribute matrix can be presented as $X^{T^r} \in \mathbf{X^V}$. The original attribute of a specific resource r_1 is denoted as $x_{r_1} \in X^{T^r}$. With a mapping function f_G, all original attribute matrices in $\mathbf{X^V}$ are transformed into continuous dense representation matrices in $\mathbf{H^V}$. Specifically, x_{u_1} and x_{r_1} are also embedded into h_{u_1} and h_{r_1}. Typically, HGNN models can be used to conduct node embedding.

Definition 4 (Link prediction) *For any nodes $u, v \in \mathcal{V}$, given their node embeddings, denoted as h_u and h_v, link prediction is the process of learning a classifier f_c that classifies whether a link exists between h_u and h_v, based on the feature fusion of h_u and h_v. It can be formulated as $\hat{y} = f_c(h_u \oplus h_v)$, where $\hat{y}$ is the predicted result and $\oplus$ is the operator of feature fusion, for example, dot product, concatenation, or average.*

Example. Figure 1 (e) illustrates the process of link prediction between the user node u_4 and the resource node r_1, given their node embeddings h_{u_4} and h_{r_1}. It involves feature fusion and classification stages. When user u_4 requests access to resource r_1, the link prediction result can be used as the access grant decision.

3 Methodology

To fuse attributes and features from multiple types of nodes and explore the intricate relationships between nodes in an ACHG, we propose a heterogeneous graph-based semi-supervised learning framework with a self-supervised node embedding component and a supervised link prediction module for access control decision-making. This section will detail the framework and its core elements.

3.1 The workflow of the proposed framework

The proposed semi-supervised learning framework consists of three stages: ACHG construction, self-supervised node embedding, and supervised access control decision-making.

As depicted in Figure 2, the workflow of the proposed framework begins with the construction of an ACHG, G, based on users' and resources' original attributes and organizational/operational information. We adopt the algorithm proposed by Mingshan, Y., et al. [29] in 2022 to build the ACHG, as we utilize the same open-sourced Amazon access control dataset. The algorithm starts by creating user and resource nodes and then iteratively builds new node types and develops relationships between new node types and existing node types or adds attributes to existing nodes or relationships based on the cardinality of the user's or resource's original attributes.

The general rule for determining whether to create a new node type from existing attributes is based on the cardinality of attributes. If the cardinality of attributes exceeds a preset thresh-

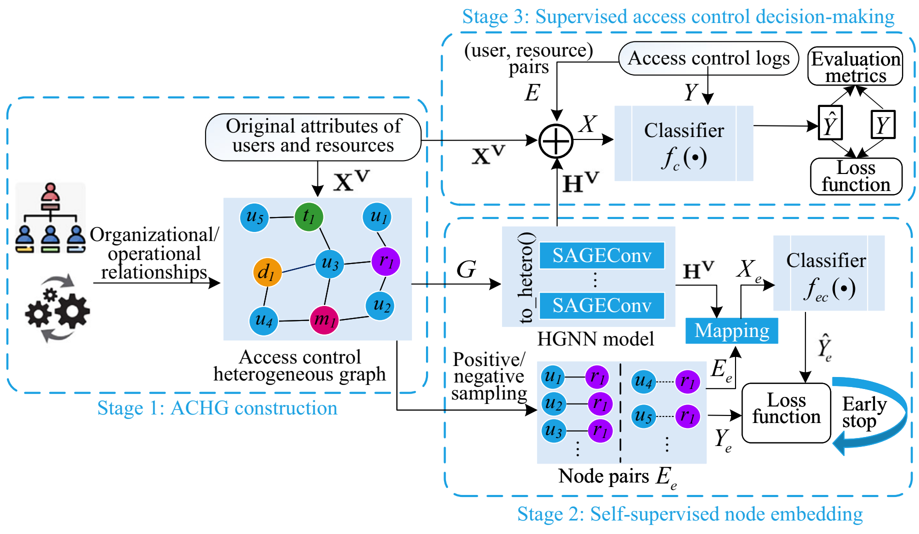
Stage 3: Supervised access control decision-making
(user, resource) pairs
E
Access control logs
Y
Evaluation metrics
X
Classifier
$f_c(\bullet)$
$\hat{Y}$
Y
Loss function
$\mathbf{X^V}$
$\mathbf{H^V}$
Original attributes of users and resources
$\mathbf{X^V}$
Organizational/ operational relationships
u_5 t_1 u_1 d_1 u_3 r_1 u_2 u_4 m_1
Access control heterogeneous graph
Stage 1: ACHG construction
G
to_hetero()
SAGEConv
SAGEConv
HGNN model
$\mathbf{H^V}$
X_e
Classifier
$f_{ec}(\bullet)$
Mapping
$\hat{Y}_e$
E_e
Positive/ negative sampling
u_1 r_1 u_2 r_1 u_3 r_1 u_4 r_1 u_5 r_1
Y_e
Loss function
Early stop
Node pairs E_e
Stage 2: Self-supervised node embedding

Figure 2 An illustration of the proposed semi-supervised framework

old, a new type of node will be created. Otherwise, the attributes will continue to be attributes of an existing node type. This approach helps avoid adding too many new relationships to decrease the value of each relationship. For more details on ACHG construction, please refer to the original paper [29].

Once the ACHG, G, has been constructed, it can be used as the input for Stage 2: self-supervised node embedding. During this process, an HGNN model consisting of a heterogeneous graph version of a two-layer GraphSAGE [38] is used to learn embedding matrices of all node types, $\mathbf{H^V}$. From G, we can select an edge type $T^e \in \mathbf{T^E}$ to define a link prediction task and construct a corresponding dataset. The existing edges of type T^e serve as positive links. Negative links can be generated through negative sampling methods, such as randomly selecting head and tail nodes with the same node type as the positive links. This process generates a set of node pairs, denoted as E_e. Subsequently, by mapping and integrating the node embeddings of nodes in E_e from $\mathbf{H^V}$, the features of E_e, denoted as X_e, can be used as the input for a link prediction classifier, denoted as $f_{ec}(\cdot)$. Then, by minimizing the loss function between the ground truth, denoted as Y_e, and the predicted link labels, $\hat{Y}_e$, the HGNN model parameters will be optimized. Finally, by partitioning the link prediction dataset into training and validation sets, we monitor the loss function's value on these sets and employ an early stopping strategy to ensure the HGNN model is properly trained, avoiding overfitting or underfitting.

Stage 3 involves supervised access control decision-making. Suppose we have access control logs, which record historical access control requests in the form of (user, resource) pairs, denoted as E, and the access grant or refusal results, which can be used as the ground truth Y of an access control dataset. To achieve better performance and retain potential information loss in the node embedding stage, both the original attributes of users and resources from $\mathbf{X^V}$ and the node embeddings $\mathbf{H^V}$ calculated from the well-trained HGNN model in Stage 2 are leveraged to generate the final features for access control decision-making, denoted as X. Subsequently, we use the generated dataset $\{X, Y\}$ to train a supervised classifier, $f_c(\cdot)$, to make access control decisions. Finally, the well-trained $f_c(\cdot)$ can be used for future access control decision-making.

The ACHG construction process implemented on the open-source Amazon access control dataset and the statistics of the constructed ACHG will be further described in Section 4.1. The detailed processes of Stage 2 and Stage 3 are presented in the following subsections.

3.2 Self-supervised node embedding

As described in Definition 1 in Section 2, a heterogeneous graph can be represented as $G = \{\mathcal{V}, \mathcal{E}, \mathbf{T^V}, \mathbf{T^E}, \mathbf{X^V}, \mathbf{X^E}\}$. In the context of the constructed ACHG based on the open-sourced Amazon access control dataset and the algorithm outlined in [29], $\mathbf{T^V}$ is defined as {user (T^u), resource (T^r), department (T^d), title (T^t), manager (T^m)} and $\mathcal{V}$ represents the set of nodes of all node types in $\mathbf{T^V}$. Similarly, $\mathbf{T^E}$ is defined as {has_potential_access ($T^{e_{ur}}$), has_department ($T^{e_{ud}}$), has_title ($T^{e_{ut}}$, has_manager ($T^{e_{um}}$)}, with $\mathcal{V}$ representing the set of edges of all edge types in $\mathbf{T^E}$. Additionally, $\mathbf{X^V}$ is given by $\{X^{T^u}, X^{T^r}\}$, where X^{T^u} and X^{T^r} represent the original attribute matrices of node types T^u and T^r, respectively. Finally, $\mathbf{X^E} = \emptyset$ indicates that the constructed ACHG contains no edge attributes.

HGNN model: We adopt the GraphSAGE [38] convolutional neural network model as the base model of HGNN. The message passing process of the l-th layer ($l \geq 1$) of a homogeneous GraphSAGE model can be formulated as (1).

$$h_i^{(l)} = W_1^{(l-1)} h_i^{(l-1)} + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} h_j^{(l-1)} W_2^{(l-1)} + b^{(l-1)}, \tag{1}$$

where $h_i^{(l)}$ is the embedding of the l-th layer for a node i; $W_1^{(l-1)}$, $W_2^{(l-1)}$, and $b^{(l-1)}$ are the learnable parameters of the homogeneous GraphSAGE model; $h_i^{(l-1)}$ is the $(l-1)$-th layer's embedding of node i; $\mathcal{N}(i)$ is the set of neighbour nodes of node i. Specifically, $h_i^{(0)} = x_i$, represents the original attributes of node i.

To generate the HGNN model used in the proposed framework, we first define a two-layer homogeneous GraphSAGE model, then convert it into its heterogeneous equivalent, following paper [39]. In HGNN, node embeddings are learned for all node types in $\mathbf{T^V}$ and messages are exchanged between all edge types in $\mathbf{T^E}$.

Let $f_\theta^{(l)}$ be the mapping function of the message passing process of GraphSAGE described in (1), where θ={$W_1^{(l-1)}$, $W_2^{(l-1)}$, $b^{(l-1)}$}. For the heterogeneous version, $f_\theta^{(l)}$ will be duplicated along with all edge types in $\mathbf{T^E}$ and stored in a set $\{f_\theta^{(l,T^e)} : T^e \in \mathbf{T^E}\}$. Then, the message passing process of the heterogeneous GraphSAGE model of layer l can be formulated as (2).

$$h_i^{(l)} = \underset{T^e \in \mathbf{T^E}}{Agg}\, f_\theta^{(l,T^e)}(h_i^{(l-1)}, \{h_j^{(l-1)} : j \in \mathcal{N}^{(T^e)}(i)\}), \tag{2}$$

where $\mathcal{N}^{(T^e)}(i)$ represents the set of all neighbor nodes of node $i \in \mathbf{T^V}$ under edge type $T^e \in \mathbf{T^E}$, and Agg denotes the aggregation strategy to use for fusing the node embeddings generated by different edge types. The general options for Agg include sum, mean, min, max, or multiplication.

Let the overall mapping function from $\mathbf{X^V}$ to $\mathbf{H^V}$ be $f_{HGNN}(\Theta, G)$, and the node embeddings for all node types in $\mathbf{T^E}$ can be calculated as (3).

$$\mathbf{H^V} = f_{HGNN}(\Theta, G), \tag{3}$$

where $\mathbf{H^V}$ is the node embeddings for all node types in $\mathbf{T^E}$, and G= $\{\mathcal{V}, \mathcal{E}, \mathbf{T^V}, \mathbf{T^E}, \mathbf{X^V}, \mathbf{X^E}\}$ is the constructed ACHG. The parameter matrix set $\Theta = \{\theta^{(l,T^e)}: l \geq 1, T^e \in \mathbf{T^E}\}$, where $\theta^{(l,T^e)}$=$\{W_1^{(l-1,T^e)}, W_2^{(l-1,T^e)}, b^{(l-1,T^e)}\}$.

Positive/negative sampling. To learn the parameter matrix set Θ of the HGNN model, we define a self-supervised learning link prediction task on the constructed ACHG. Specifically, we select the edge type has_potential_access ($T^{e_{ur}}$) from $\mathbf{T^E}$ to build a dataset for link prediction. The positive links can either be the entire set of edges with the relationship $T^{e_{ur}}$ or a randomly sampled subset, a technique known as positive sampling. In either case, these links' corresponding ground truth label, denoted as Y_e, is set to 1. To generate negative links, with $Y_e = 0$, we employ negative sampling by randomly selecting head nodes from nodes in the user type (T^u) and tail nodes from nodes in the resource type (T^r).

It's worth mentioning that if a user node u has potential access to a resource node r, it only suggests that u may request access to r, but this request may not necessarily be approved. This scenario is akin to situations in an online shopping heterogeneous graph, where a user clicks the link on a product, which does not necessarily result in the user making a purchase.

Mapping. Once the node pairs $E_e = \{(u, r) : u \in T^u, r \in T^r\}$ with ground truth Y_e are generated, we can correspondingly create the node embedding feature set for E_e, denoted as X_e, by mapping the node IDs of E_e with the node embeddings in $\mathbf{H^V}$. Specifically, the feature for an edge $e_{ur} \in T^{e_{ur}}\}$, where the head and tail node pair $(u, r) \in E_e$, is the concatenation of node embeddings of nodes u and r, shown in (4).

$$x_{e_{ur}} = [h_u || h_r], \tag{4}$$

where $||$ denotes the concatenation operation, $h_u \in H^{T^u} \in \mathbf{H^V}$, and $h_r \in H^{T^r} \in \mathbf{H^V}$. By mapping the node embeddings of node pairs $(u, r) \in E_e$ from $\mathbf{H^V}$, the corresponding feature set, X_e, for the link prediction task has been formed.

HGNN training and early stop. After forming the dataset, $\{X_e, Y_e\}$, for link prediction, a classifier $f_{ec}(\cdot)$ is adapted to generate the predicted results, $\hat{Y}_e$, as shown in (5).

$$\hat{Y}_e = f_{ec}(X_e, \theta_{ec}), \tag{5}$$

where θ_{ec} is the learnable parameter set of the classifier $f_{ec}(\cdot)$. The parameters of the HGNN model and the classifier, i.e., Θ and θ_{ec}, can be jointly learned by minimizing a binary classification loss function $\mathcal{L}(Y_e, \hat{Y}_e)$.

The early stop strategy is employed to avoid underfitting or overfitting the training process. Specifically, the dataset, $\{X_e, Y_e\}$, is split into a training set and a validation set. At the beginning of training, both training loss and validation loss decrease. However, when the HGNN model approaches the best model, validation loss increases when training loss decreases. The value of patience can be used to control when to stop the training process to avoid overfitting.

Once the training process stops, the HGNN model is well-trained, and the node embeddings in $\mathbf{H^V}$ calculated from the well-trained HGNN model can be used as the features of nodes for downstream applications, including node classification, link prediction, and graph-level tasks.

3.3 Supervised access control decision-making

Access control logs serve as comprehensive records documenting and monitoring activities related to access control systems. These logs provide detailed insights into who accessed specific resources, the timing of these access events, and any pertinent details associated with access attempts. Leveraging these logs is crucial for developing access control decision-making models tailored to contemporary access control scenarios.

Stage 3 of the proposed framework involves extracting a set of user and resource pairs, denoted as $E = \{(u, r) : u \in T^u, r \in T^r\}$, along with the corresponding ground truth of access requests, denoted as Y. In this context, if an access request from a user node u to a resource node r is approved, $y_{ur} \in Y$ equals 1; otherwise, it equals 0.

Features for an access request $u \to r$ are sourced from two places. The first one is the original attributes of users and resources, denoted as $x^u \in X^{T^u} \in \mathbf{X^V}$ and $x^r) \in X^{T^r} \in \mathbf{X^V}$. Another one is node embeddings obtained from the well-trained HGNN model, denoted as $h^u \in H^{T^u} \in \mathbf{H^V}$ and $h^r) \in H^{T^r} \in \mathbf{H^V}$. For each node pair $(u, r) \in E$, the final feature $x_{ur} \in X$ used for making access control decisions is calculated using (6).

$$x_{ur} = [x_u || x_r] \oplus [h_u || h_r], \tag{6}$$

where $\oplus$ is the feature fusion operation.

Subsequently, a binary classifier $f_c(\cdot)$ is trained and tested on the dataset $\{X, Y\}$ to make access control decisions for requests $u \to r$, where $(u, r) \in E$, following the general practice of machine learning classification tasks.

4 Experiments

4.1 Dataset and ACHG construction

This paper uses the open-sourced Amazon access control sample data [37] sourced from the UCI Machine Learning Repository, comprising two CSV files. The first file provides static information about original user and resource attributes, serving as the foundation for constructing the ACHG. We transcribe the provided variable and description details, organizing them into the first two columns of Table 1. Additionally, we introduce three columns-Cardinality, Usage, and Action-to describe the process of building an ACHG. The second CSV file contains a dynamic access control log with 716,063 historical records of user $\rightarrow$ resource requests and their corresponding actions, which include adding or removing access. This dynamic log is instrumental in generating the (user, resource) pair set E and the associated ground truth Y for Stage 3 of the proposed framework.

Following the knowledge graph construction algorithm outlined in [29], we set the cardinality threshold to 300. This threshold indicates that if a variable's cardinality exceeds 300, it will be utilized to establish a new node type and forge new relationship types between existing node types and the newly created one. Conversely, variables with a cardinality of 300 or less will be used as attributes of existing node types. Thus, as depicted in Table 1, apart from the user and resource node types, new node types such as manager, department, and title are established, along with their corresponding relationships. The schema of the final constructed ACHG is illustrated in Figure 3, and the statistical overview of the ACHG is provided in Table 2.

4.2 Experimental setting and self-supervised node embedding results

All experiments in this paper are implemented using Python as the programming language. The essential libraries employed for data processing include Pandas and NumPy, while Matplotlib is utilized for visualization purposes. Scikit-learn facilitates the implementation of classifiers during the supervised access control decision-making stage. PyTorch serves as the primary deep-learning framework, with PyTorch Geometric utilized for the implementation of HGNN models. More implementation details can be found at the GitHub repository associated with this paper[1]. An Nvidia GeForce RTX 2080 Ti GPU is utilized for HGNN self-supervised training.

To construct the HGNN model described in Figure 2, we build a two-layer homogeneous GraphSAGE model using the SAGEConv() module. Subsequently, a to_hetero() method is applied to convert this model into its heterogeneous equivalent. Additionally, we utilize a ToUndirected() method to convert the constructed ACHG into an undirected graph. This conversion involves adding reverse connections for all existing edge types.

Then, to build the link prediction dataset for self-supervised node embedding, the 595,506 edges of the form u has_potential_access r are used as positive links in E_e, where the corresponding $y_e \in Y_e = 1$. The RandomLinkSplit() method in the torch_geometric package is then adopted to split E_e into a training set and a validation set, with a split ratio of 0.9:0.1. Additionally, we set the ratio of sampled negative edges to the number of positive edges as 2, where the corresponding $y_e \in Y_e = 0$.

[1] https://github.com/happyResearcher/ACHG-SSLF

Table 1 Amazon access control dataset description and usage statement

Variable	Description	Cardinality	Usage	Action
PERSON_ID	id of the user	36,063	node type user's id	
PERSON_MGR_ID	id of the user's manager	3,207	node type manager's id	create a has_manager relationship between user and manager
PERSON_ROLLUP_1	user grouping id	12	node type user's attribute	add the attribute to user
PERSON_ROLLUP_2	user grouping id	111	node type user's attribute	add the attribute to user
PERSON_ROLLUP_3	user grouping id	12	node type user's attribute	add the attribute to user
PERSON_DEPTNAME	department description id	405	node type department's id	create a has_department relationship between user and department
PERSON_BUSINESS_TITLE	title id	4,979	node type title's id	create a has_title relationship between user and title
PERSON_BUSINESS_TITLE_DETAIL	description id	56	node type user's attribute	add the attribute to user
PERSON_JOB_CODE	job code id	13	node type user's attribute	add the attribute to user
PERSON_COMPANY	company id	49	node type user's attribute	add the attribute to user
PERSON_JOB_FAMILY	job family id	70	node type user's attribute	add the attribute to user
RESOURCE_ID	resource id that a users can possibly have access to	33,252	node type resource's id	create a has_potential_access relationship between user and resource
GROUP_ID	group type that a users can possibly have access to	3	node type resource's attribute	add the attribute to resource

Note: The first two columns are transcribed directly from the data source. The Cardinality column lists the unique values of the variables, while the Usage column specifies whether the variable represents an attribute or the ID of a node type. The Action column describes how to use the corresponding variable to build an ACHG

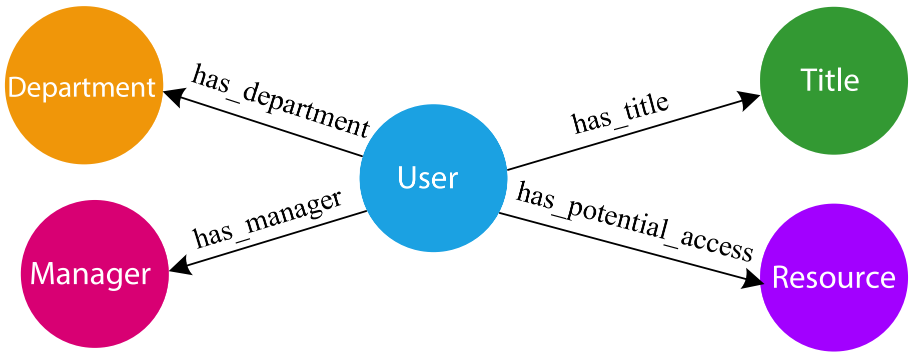

Figure 3 The schema of the constructed ACHG

Since the self-supervised node embedding is not the target downstream application, there is no need to set a separate test set to evaluate the performance of predicting the existence of the has_potential_access edges. Instead, only a training and validation sets are required to train the HGNN model and apply the early stop strategy to prevent overfitting and underfitting.

By mapping the node IDs of node pairs in E_e to the output of the HGNN model, i.e., the node embeddings in $\mathbf{H}^{\mathbf{V}}$, we form the feature set X_e, along with the corresponding ground truth labels in Y_e, to create the dataset for a link prediction task. We utilize the Adam optimizer with a learning rate of 0.001 to minimize the loss function on the training set, gradually training the parameters of the HGNN model. To prevent underfitting, we ensure a sufficiently large number of training epochs. Additionally, adopting the early stop strategy with a patience of 10 on the validation set to prevent overfitting.

Figure 4 illustrates the learning curve of the self-supervised node embedding process. The training loss shows a fluctuating decrease over the epochs during the training process. In contrast, the validation loss reaches its lowest value when the epoch is 112, as indicated by the red dashed line denoting the early stop checkpoint. Finally, the training process halts at epoch 122, given that the patience parameter is set to 10 in the experiments. The model parameters at the early stop checkpoint are preserved as the best-trained HGNN model for node embeddings.

Table 2 Statistics of the constructed ACHG

Node type	Node original attributes	Count	Relationship type	Count
user	PERSON_ID, PERSON_ROLLUP_1, PERSON_ROLLUP_2, PERSON_ROLLUP_3, PERSON_BUSINESS_TITLE_DETAIL, PERSON_JOB_CODE, PERSON_COMPANY, PERSON_JOB_FAMILY	36,063		
resource	RESOURCE_ID, GROUP_ID	33,252	has_potential_access	595,506
department	PERSON_DEPTNAME	405	has_department	36,063
title	PERSON_BUSINESS_TITLE	4,979	has_title	36,063
manager	PERSON_MGR_ID	3,207	has_manager	36,063

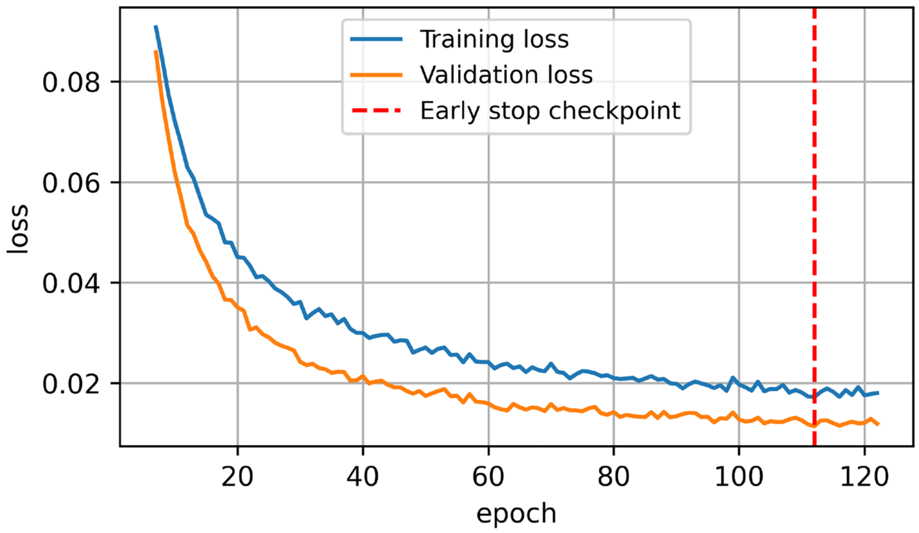

Figure 4 Early stop learning curve of the self-supervised learning process

4.3 Evaluation metrics

Since access control decision-making for a request where user u requests a resource r is a binary classification problem, this paper employs four well-known evaluation metrics: accuracy, precision, recall, and F1 score.

The real-world access control problems are often highly imbalanced, with most requests being legitimate and only a small proportion being unauthorized [40, 41]. To evaluate the performance of the proposed framework under different imbalance statuses, three datasets are manually constructed with ratios of access-approved samples to access-denied samples set to 5:5, 3:7, and 9:1, in addition to the original dataset with a ratio of 98.41:1.59.

Denying legitimate requests may lead to repeated access control requests or even human intervention, reducing the smooth operation and efficiency of information systems. Conversely, wrongly granting access to unauthorized requests may result in security breaches, compromised data, legal repercussions, and reputational damage for individuals, organizations, or systems involved. Therefore, the performance of the minority (negative) class is more crucial than that of the majority (positive) class for access control problems. In the following sections, we not only report the macro-average performance of an algorithm over the whole dataset but also focus on the minority (negative) class exclusively.

To compare the performance of the proposed method with other algorithms, we introduce a Δ F1 score defined as (7).

$$\Delta \mathrm{F1} = \frac{\mathrm{F1}_p - \mathrm{F1}_c}{\mathrm{F1}_c} \times 100\%, \tag{7}$$

where $\mathrm{F1}_p$ represents the F1 score of the proposed method and $\mathrm{F1}_c$ represents that of the comparison method. If $\Delta\mathrm{F1} > 0$, the proposed method outperforms the comparison method; otherwise, the comparison method performs better.

4.4 Access control performance evaluation

To evaluate the performance of the proposed framework in the Stage 3 access control decision-making task, we compare the HGNN node embedding-based feature extraction strategies with the non-topological (Nontopo) manual features and topological (Topo) manual features adopted in [29]. Specifically, the four feature extraction strategies in this section as described below:

- **Nontopo**: Denotes the manually extracted non-topological feature set from the original attributes of individual users and resources. These attributes include u.userID, u.titleDetail, u.Company, u.jobCode, u.jobFamily, u.Rollup1, u.Rollup2, u.rollup3, r.resourceID, and r.resourceType, listed in Table 3 of [29].
- **Topo**: Represents the topological feature set extracted from a constructed knowledge graph for the link prediction task. These features are derived through a series of manual topological feature extraction algorithms, including PageRank, ArticleRank, Betweenness, Degree, Closeness, Louvain, HarmonicCloseness, LabelPropagation, WCC, triangleCount and Modularity, following the implementation of [29].
- **NodeEmb**: Refers to the node embeddings of users and resources learned from the trained HGNN model.
- **NodeEmb+**: Encompasses NodeEmb and the original attributes of users and resources.

The reported results of Nontopo and Topo features in this section are directly excerpted from Table 4 of [29]. This paper shares exactly the same experimental settings and dataset splitting methods with [29], ensuring consistency and comparability in the evaluation process.

To assess the efficacy of the proposed methods across various classifiers and imbalanced dataset scenarios, we conducted experiments using different classifiers on a balanced dataset, as outlined in Section 4.4.1. Additionally, we present the experimental results for different imbalanced datasets in Section 4.4.2.

4.4.1 Performance comparison on a balanced dataset

Table 3 presents the comparison results on a balanced dataset. The first column, Classifier, lists four well-known machine learning algorithms: logistic regression (LR), multi-layer perceptron (MLP), random forest (RF), and support vector machine (SVM), serving as the $f_c(\cdot)$ in Figure 2. The implementations of these classifiers are sourced from the scikit-learn package, and their hyper-parameter settings can be found in detail in the GitHub project of this paper.

Table 3 reveals that NodeEmb+ achieves the highest accuracy (Acc(%)) across all classifiers, slightly surpassing NodeEmb and significantly outperforming both Topo and Nontopo. Specifically, RF stands out as the top-performing classifier, achieving an accuracy of 76.25%. This trend is also reflected in the macro average F1 scores. Regarding the macro average Δ F1, a value of 0.00% for NodeEmb+ signifies that it represents the proposed method. In contrast, positive values in other rows indicate the percentage improvements that NodeEmb+ achieves compared to the respective methods. For example, NodeEmb+ demonstrates a 7.94% improvement in macro average Δ F1 compared to Nontopo. It is worth noting that the Topo feature set outperforms NodeEmb+ and NodeEmb in terms of Recall and F1 scores for the negative class (access reject) when using the SVM classifier. This, however, does not imply superior overall performance of the Topo feature set, as it performs poorly on the positive class. This discrepancy could be attributed to the SVM classifier's proficiency in classifying the negative class in this specific application scenario.

Table 3 Performance comparison of different feature strategies on balanced dataset

Classifier	Feature	Acc(%)	Macro average				Negative class			
			Pre(%)	Rec(%)	F1(%)	Δ F1	Pre(%)	Rec(%)	F1(%)	Δ F1
LR	Nontopo	61.04	61.14	61.09	61.02	5.45%	62.61	57.55	59.97	6.45%
	Topo	62.89	63.08	62.96	62.83	2.41%	65.05	57.97	61.31	4.12%
	NodeEmb	63.94	64.01	63.98	63.93	0.66%	65.41	61.31	63.29	0.86%
	NodeEmb+	64.35	64.41	64.39	64.35	0.00%	65.74	62.04	63.84	0.00%
MLP	Nontopo	60.63	61.07	60.75	60.38	19.24%	63.68	52.04	57.27	27.62%
	Topo	61.46	61.65	61.53	61.39	17.27%	63.51	56.39	59.74	22.34%
	NodeEmb	69.28	69.29	69.29	69.28	3.91%	70.05	68.88	69.46	5.22%
	NodeEmb+	72.04	72.08	72.00	71.99	0.00%	71.38	74.87	73.09	0.00%
RF	Nontopo	70.65	70.74	70.69	70.64	7.94%	72.52	67.81	70.08	8.78%
	Topo	73.61	73.64	73.63	73.61	3.59%	74.86	72.22	73.51	3.70%
	NodeEmb	76.02	76.04	76.04	76.02	0.30%	77.18	74.84	75.99	0.31%
	NodeEmb+	76.25	76.27	76.27	76.25	0.00%	77.40	75.10	76.23	0.00%
SVM	Nontopo	61.07	61.18	61.13	61.04	5.13%	62.69	57.38	59.92	6.05%
	Topo	50.51	47.95	49.83	35.21	82.25%	50.62	97.71	66.69	-4.72%
	NodeEmb	63.45	63.57	63.50	63.42	1.18%	65.22	59.84	62.41	1.82%
	NodeEmb+	64.18	64.25	64.22	64.17	0.00%	65.66	61.56	63.54	0.00%

The F1 score comparisons of different feature extraction methods across different classifiers are visualized in Figure 5. It is evident that NodeEmb+ generally exhibits the best performance, both in terms of the macro average and the negative class. The exception of Topo on SVM can also be observed in Figure 5. Clearly, the RF classifier achieves the best performance among the four classifiers. Therefore, we utilize RF as the classifier for the remaining experiments in this paper unless otherwise specified.

4.4.2 Performance comparison on imbalanced datasets

To assess the robustness of the proposed methods on imbalanced datasets, we randomly select positive samples to construct datasets with proportions of negative samples set to 0.3 and 0.1. We also investigate the performance on the original access log of the open-sourced Amazon dataset, where the negative ratio is 0.0159. Table 4 presents the performance comparison results, indicating that NodeEmb and NodeEmb+ achieve higher accuracy than Nontopo and Topo on all imbalanced datasets. Notably, when the minority ratio equals 0.3, NodeEmb slightly outperforms NodeEmb+, reaching 73.92% on the macro average F1 score and 61.34% on the minority class F1 score, compared with 73.83% and 61.21% achieved by NodeEmb+, respectively. Conversely, when the minority ratio equals 0.1 and 0.0159, NodeEmb+ performs slightly better than NodeEmb. However, overall, the performance of NodeEmb and NodeEmb+ are comparable and significantly better than that of Nontopo and Topo. An exception is observed for the minority ratio of 0.0159, where Topo achieves the best F1 score for the minority class at 5.32%, surpassing NodeEmb+ by 6.03% at 5%. This result is consistent with the exception noted in Table 3.

Similarly, we visualize the F1 score comparison results over different minority ratios in Fig 6 to facilitate the interpretation of experimental findings. Overall, the proposed NodeEmb and NodeEmb+ feature sets consistently outperform the Nontopo and Topo feature sets proposed in [29].

4.4.3 Discussion

Experiments on various classifiers and imbalance settings consistently demonstrate that the proposed HGNN-based feature extraction strategies, NodeEmb and NodeEmb+, outperform

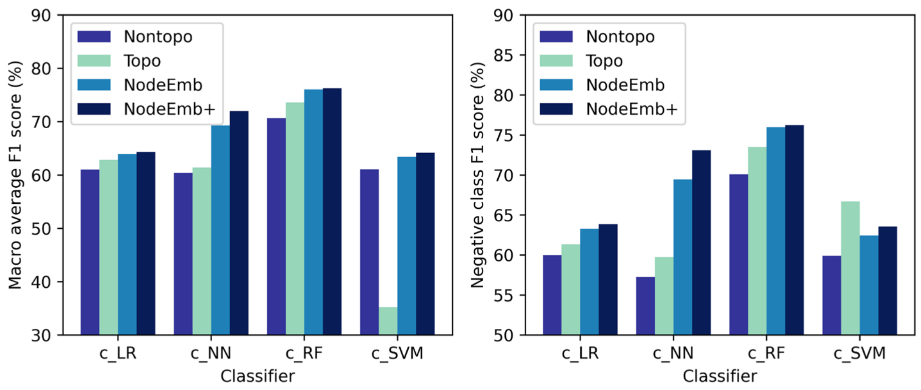

Figure 5 Comparison of F1 scores across different classifiers on a balanced dataset

Table 4 Performance comparison of different feature strategies on imbalanced datasets

Minority Ratio	Feature	Acc(%)	Macro average				Minority class			
			Pre(%)	Rec(%)	F1(%)	Δ F1	Pre(%)	Rec(%)	F1(%)	Δ F1
0.3	Nontopo	76.87	73.72	66.87	68.41	7.92%	68.75	41.9	52.07	17.55%
	Topo	78.57	75.76	69.89	71.56	3.17%	71.02	48.21	57.43	6.58%
	NodeEmb	79.99	77.43	72.26	73.92	-0.12%	72.91	52.94	61.34	-0.21%
	NodeEmb+	79.91	77.31	72.18	73.83	0.00%	72.70	52.85	61.21	0.00%
0.1	Nontopo	89.41	67.70	59.16	61.49	2.32%	43.63	21.38	28.69	9.60%
	Topo	89.56	68.58	59.82	62.28	1.03%	45.26	22.67	30.21	4.08%
	NodeEmb	89.67	69.22	60.34	62.90	0.02%	46.45	23.71	31.39	0.17%
	NodeEmb+	89.64	69.07	60.38	62.92	0.00%	46.13	23.85	31.44	0.00%
0.0159	Nontopo	98.01	53.29	51.08	51.48	1.03%	8.1	2.63	3.97	25.92%
	Topo	97.98	54.24	51.55	52.15	-0.26%	9.99	3.62	5.32	-6.03%
	NodeEmb	98.07	53.75	51.42	51.95	0.12%	8.94	3.35	4.88	2.48%
	NodeEmb+	98.07	53.85	51.46	52.01	0.00%	9.14	3.44	5.00	0.00%

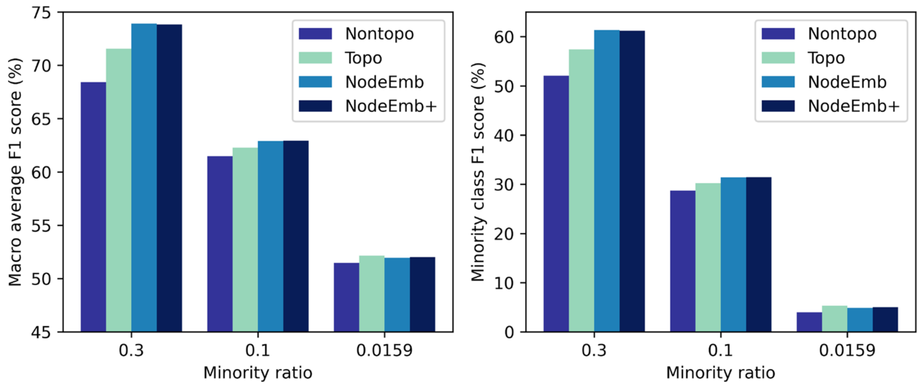

Figure 6 Comparison of F1 scores across various minority ratios

Nontopo and Topo features, while NodeEmb and NodeEmb+ perform comparably. This superiority can be attributed to several factors listed below.

- **Semantic representation**: NodeEmb captures the semantic representations of users and resources through node embeddings learned from the trained HGNN model. Unlike Nontopo, which relies solely on manually extracted non-topological features from users and resources, NodeEmb leverages the underlying structure and relationships within the constructed access control knowledge graph, resulting in more comprehensive and informative representations.
- **Heterogeneous graph-based information**: NodeEmb heterogeneously incorporates graph-based information, inherently encoded within the knowledge graph's topology. Consequently, it can capture complex relational patterns and dependencies among users and resources simultaneously, surpassing the purely homogeneous extraction of topological features in Topo.
- **Dimensionality reduction**: NodeEmb effectively reduces users' and resources' original high-dimensional feature space into lower-dimensional embeddings, preserving essential information while mitigating the curse of dimensionality. This allows NodeEmb to capture relevant patterns and relationships more efficiently than Nontopo and Topo, which may suffer from high-dimensional feature spaces and potential sparsity issues.
- **Adaptive learning**: Through iterative learning, NodeEmb refines its embeddings adaptively based on the graph's topology and structure, potentially capturing subtle but meaningful relationships that may not be explicitly represented in Nontopo or Topo.

Overall, NodeEmb's ability to capture semantic representations, leverage graph-based information, reduce dimensionality, and adaptively learn from the knowledge graph contributes to its superior performance compared to Nontopo and Topo in the context of the link prediction task.

4.5 ACHG graph structure ablation study

As depicted in Figure 3, the constructed ACHG comprises four edge types: has_department, has_title, has_manager, and has_potential_access. Among these, has_potential_access is selected for the link prediction task in self-supervised node embedding, as the edge head and tail correspond to access requests. Therefore, this relationship is always retained in ACHG during this graph structure ablation study.

The experiments in this section investigate the impact of graph structure on the final access control decision-making performance through ablation studies. The results are reported in Table 5, where the second column shows five different ACHG graph structures. Specifically, ACHG_All refers to the entire constructed ACHG, as illustrated in Figure 3; ACHG-N indicates that only the has_potential_access edges are retained, with none of the other types of edges included; ACHG-M, ACHG-D, and ACHG-T represent ACHG-N supplemented with has_manager, has_department, and has_title edges, respectively.

Surprisingly, despite the intuitive notion that a more complex graph would contain more information and lead to better performance, the results show that different ACHG structures perform equivalently across all negative ratio settings. Specifically, when examining negative class F1 scores, ACHG-T performs best at 76.51% and 31.93% for negative ratios of 0.5 and 0.1, respectively. For a negative ratio of 0.3, ACHG-M achieves the highest score at 61.38%, while ACHG-All performs best at 5% for a negative ratio of 0.0159. These findings align with observations from other studies suggesting that explicit edge-type information does not significantly enhance downstream applications [42]. One hypothesis to explain this phenomenon is that the current HGNN message passing and information fusion mechanisms

Table 5 Performance comparison of various ACHG graph structures

Negative ratio	ACHG structure	Acc(%)	Macro average			Negative class		
			Pre(%)	Rec(%)	F1(%)	Pre(%)	Rec(%)	F1(%)
0.5	ACHG-All	76.25	76.27	76.27	76.25	77.40	75.10	76.23
	ACHG-M	76.44	76.46	76.46	76.44	77.72	75.04	76.36
	ACHG-D	76.18	76.19	76.19	76.18	77.25	75.16	76.19
	ACHG-T	76.62	76.66	76.65	76.62	78.00	75.07	76.51
	ACHG-N	76.47	76.48	76.48	76.47	77.61	75.33	76.45
0.3	ACHG-All	79.91	77.31	72.18	73.83	72.70	52.85	61.21
	ACHG-M	80.02	77.49	72.28	73.95	73.02	52.94	61.38
	ACHG-D	79.46	76.81	71.40	73.07	72.20	51.25	59.95
	ACHG-T	79.77	77.10	72.02	73.65	72.37	52.65	60.96
	ACHG-N	79.83	77.24	72.01	73.68	72.68	52.48	60.95
0.1	ACHG-All	89.64	69.07	60.38	62.92	46.13	23.85	31.44
	ACHG-M	89.63	68.98	60.19	62.71	45.99	23.42	31.03
	ACHG-D	89.68	69.25	60.39	62.96	46.49	23.82	31.50
	ACHG-T	89.69	69.37	60.59	63.18	46.70	24.25	31.93
	ACHG-N	89.65	69.11	60.33	62.87	46.24	23.71	31.34
0.0159	ACHG-All	98.07	53.85	51.46	52.01	9.14	3.44	5.00
	ACHG-M	98.07	53.78	51.43	51.97	8.99	3.38	4.92
	ACHG-D	98.06	53.77	51.45	51.98	8.97	3.41	4.94
	ACHG-T	98.07	53.78	51.42	51.96	8.99	3.35	4.89
	ACHG-N	98.08	53.78	51.41	51.94	8.99	3.33	4.85

may be insufficient to discern differences. Another possibility is that the simplest graph version already possesses adequate strength to learn all necessary embeddings for the relevant downstream applications. However, this remains an open question, necessitating further investigation in future research endeavors.

Figure 7 compares the self-supervised node embedding training time and training epochs across different ACHG graph structures. The embedding length is set to 64, and the batch size is fixed at 1024 for all graphs. Among the various graph structures, ACHG-All, the most complex structure, exhibits the longest training time at 5607.93 seconds and the highest number of training epochs at 121. ACHG-T, which involves 4,979 title nodes, ranks second in terms of embedding training time, recording 4410.25 seconds.

4.6 Impact of the embedding length hyper-parameter

This subsection presents the comparison results of hyper-parameter embedding lengths on performance, embedding training time, and embedding training epochs, as shown in Table 6 and Figure 8. The graph used in these experiments is ACHG-N, and the epoch size is 1024.

As depicted in Table 6, in general, shorter embedding lengths generally outperform longer ones on the ACHG-N graph constructed in our study. For example, an embedding length of 32 yields the highest accuracy, macro average F1 score, and negative class F1 score when the negative ratio is 0.5. Similarly, for negative ratios of 0.3 and 0.1, embedding lengths of 16 and 32 perform nearly equally well and outperform other lengths. In the case of a negative ratio of 0.0159, an embedding length of 16 achieves the best performance in terms of both the macro average and negative class F1 scores.

Regarding the embedding training time and epochs, a consistent trend emerges: shorter embedding lengths correlate with longer training times and larger training epochs, as shown in Figure 8. We hypothesize that this phenomenon occurs because shorter embedding lengths result in a finer-grained training process, thereby necessitating more time and epochs for convergence but ultimately leading to improved performance. However, it's worth noting that the impact of embedding length may vary across different graphs and downstream applications. Furthermore, the mechanism by which embedding length influences the self-supervised node embedding process remains inadequately understood in the existing literature.

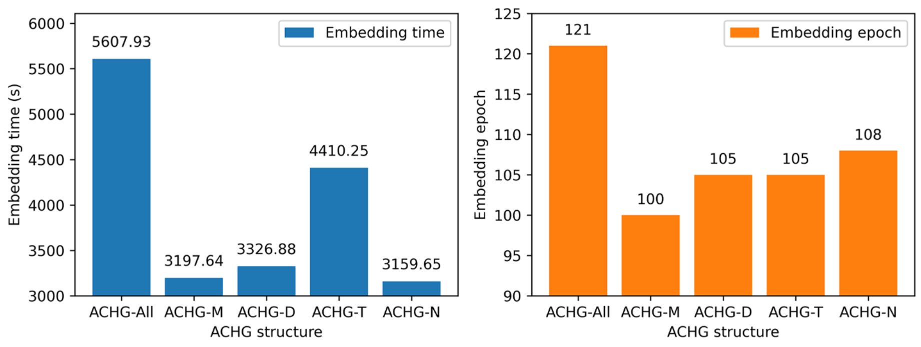

Figure 7 Comparison of node embedding training time and training epochs across different ACHG graph structures

Table 6 Comparison of hyper-parameter embedding lengths on performance

Negative ratio	Embedding length	Acc(%)	Macro average			Negative class		
			Pre(%)	Rec(%)	F1(%)	Pre(%)	Rec(%)	F1(%)
0.5	16	75.81	75.81	75.81	75.81	76.64	75.21	75.92
	32	76.61	76.62	76.62	76.61	77.64	75.66	76.64
	64	76.39	76.39	76.40	76.39	77.02	76.17	76.60
	128	75.98	75.99	75.99	75.98	77.00	75.04	76.01
	256	75.65	75.68	75.67	75.65	77.00	74.11	75.53
0.3	16	80.08	77.42	72.59	74.20	72.63	53.89	61.87
	32	79.98	77.19	72.61	74.16	72.11	54.20	61.89
	64	79.73	76.98	72.06	73.66	72.07	52.91	61.02
	128	79.61	77.05	71.54	73.24	72.61	51.39	60.18
	256	79.41	76.66	71.46	73.09	71.82	51.59	60.05
0.1	16	89.75	69.68	60.83	63.46	47.28	24.71	32.46
	32	89.71	69.51	60.90	63.49	46.91	24.91	32.54
	64	89.65	69.17	60.50	63.05	46.32	24.08	31.69
	128	89.60	68.86	60.26	62.76	45.74	23.62	31.15
	256	89.60	68.85	60.13	62.64	45.74	23.33	30.90
0.0159	16	98.07	54.03	51.55	52.13	9.49	3.61	5.23
	32	98.07	53.71	51.41	51.93	8.85	3.33	4.84
	64	98.07	53.89	51.49	52.05	9.22	3.50	5.07
	128	98.07	53.66	51.37	51.88	8.75	3.24	4.73
	256	98.08	53.82	51.40	51.93	9.07	3.30	4.84

5 Conclusion

Access control stands as a fundamental pillar for safeguarding the security and integrity of modern information systems. This paper introduces an innovative approach, a heterogeneous graph-based semi-supervised learning framework, for constructing access control

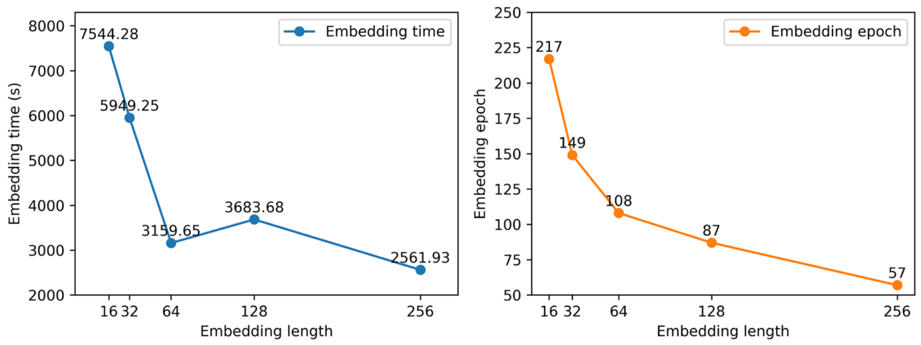

Figure 8 The relationships of embedding length vs embedding time and embedding epoch

decision-making models. Through rigorous experimentation on an open-sourced Amazon access control dataset, the efficacy of this framework is demonstrated in enhancing access control performance across balanced datasets and various imbalance settings. Moreover, the exploration of the influence of heterogeneous graph structure and node embedding length on access control performance offers invaluable insights applicable to a broad spectrum of heterogeneous graph-based applications. Therefore, this research not only provides a pioneering solution for access control but also presents a promising methodology for enhancing downstream applications based on heterogeneous graphs and HGNNs.

Author Contributions
•Jiao Yin contributes to conceptualization, methodology, software, and writing - original draft.
•Guihong Chen contributes to conceptualization and writing - original draft.
•Wei Hong contributes to software, validation, and writing - review & editing.
•Jinli Cao contributes to supervision, formal analysis, and writing - review & editing.
•Hua Wang contributes to supervision, project administration, and writing - review & editing.
•Yuan Miao contributes to supervision and writing - review & editing.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. Not applicable.

Availability of data and materials The original data utilized in this paper is available at: https://archive.ics.uci.edu/dataset/216/amazon+access+samples. The prepossessed data and codes are available on GitHub: https://github.com/happyResearcher/ACHG-SSLF.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval Not applicable.

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Hong, W., Yin, J., You, M., Wang, H., Cao, J., Li, J., Liu, M., Man, C.: A graph empowered insider threat detection framework based on daily activities. ISA Trans. **141**, 84–92 (2023). https://doi.org/10.1016/j.isatra.2023.06.030
2. Manoharan, P., Yin, J., Wang, H., Zhang, Y., Ye, W.: Insider threat detection using supervised machine learning algorithms. Telecommunication Systems. 1–17 (2023). https://doi.org/10.1007/s11235-023-01085-3
3. Sun, X., Wang, H., Plank, A.: An efficient hash-based algorithm for minimal k-anonymity. Proc Thirty-First Aust Conf Comp Sci. **74**, 101–107 (2008). https://doi.org/10.1145/1378279.1378297
4. Kabir, M.E., Mahmood, A.N., Wang, H., Mustafa, A.K.: Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing. IEEE Transactions on Cloud Computing. **8**(2), 408–417 (2020). https://doi.org/10.1109/TCC.2015.2469649

5. Wang, H., Sun, L.: Trust-involved access control in collaborative open social networks. In: 2010 Fourth International Conference on Network and System Security, pp. 239–246 (2010). https://doi.org/10.1109/NSS.2010.13. IEEE
6. You, M., Yin, J., Wang, H., Cao, J., Miao, Y.: A minority class boosted framework for adaptive access control decision-making. In: International Conference on Web Information Systems Engineering, pp. 143–157 (2021). https://doi.org/10.1007/978-3-030-90888-1_12. Springer
7. Wang, H., Zhang, Y., Cao, J., Varadharajan, V.: Achieving secure and flexible m-services through tickets. IEEE Transactions on Systems, Man, and Cyberne-Part A: Systems and Humans. **33**(6), 697–708 (2003). https://doi.org/10.1109/TSMCA.2003.819917
8. Wang, H., Zhang, Y., Cao, J.: Effective collaboration with information sharing in virtual universities. IEEE Trans. Knowl. Data Eng. **21**(6), 840–853 (2009). https://doi.org/10.1109/TKDE.2008.132
9. Yin, J., Chen, G., Hong, W., Wang, H., Cao, J., Miao, Y.: Empowering vulnerability prioritization: A heterogeneous graph-driven framework for exploitability prediction. In: International Conference on Web Information Systems Engineering, pp. 289–299 (2023). https://doi.org/10.1007/978-981-99-7254-8_23. Springer
10. Wang, Y., Shen, Y., Wang, H., Cao, J., Jiang, X.: Mtmr: Ensuring mapreduce computation integrity with merkle tree-based verifications. IEEE Transactions on Big Data. **4**(3), 418–431 (2018). https://doi.org/10.1109/TBDATA.2016.2599928
11. Ge, Y.-F., Bertino, E., Wang, H., Cao, J., Zhang, Y.: Distributed cooperative coevolution of data publishing privacy and transparency. ACM Transactions on Knowledge Discovery from Data 18 (2023). https://doi.org/10.1145/3613962
12. Bertino, E., Bonatti, P.A., Ferrari, E.: Trbac: A temporal role-based access control model. In: Proceedings of the Fifth ACM Workshop on Role-based Access Control, pp. 21–30 (2000). https://doi.org/10.1145/344287.344298
13. Wang, H., Cao, J., Zhang, Y.: A flexible payment scheme and its role-based access control. IEEE Trans. Knowl. Data Eng. **17**(3), 425–436 (2005). https://doi.org/10.1109/TKDE.2005.35
14. Servos, D., Osborn, S.L.: Current research and open problems in attribute-based access control. ACM Computing Surveys (CSUR). **49**(4), 1–45 (2017). https://doi.org/10.1145/3007204
15. Wang, H., Sun, L., Bertino, E.: Building access control policy model for privacy preserving and testing policy conflicting problems. J. Comput. Syst. Sci. **80**(8), 1493–1503 (2014). https://doi.org/10.1016/j.jcss.2014.04.017
16. Wang, H., Cao, J., Zhang, Y.: Ticket-based service access scheme for mobile users. Australian Computer Science Communications. 285–292 (2002). https://doi.org/10.1145/563857.563834
17. Shu, J., Jia, X., Yang, K., Wang, H.: Privacy-preserving task recommendation services for crowdsourcing. IEEE Trans. Serv. Comput. **14**(1), 235–247 (2021). https://doi.org/10.1109/TSC.2018.2791601
18. Cheng, K., Wang, L., Shen, Y., Wang, H., Wang, Y., Jiang, X., Zhong, H.: Secure kk-nn query on encrypted cloud data with multiple keys. IEEE Transactions on Big Data. **7**(4), 689–702 (2021). https://doi.org/10.1109/TBDATA.2017.2707552
19. Ge, Y.-F., Wang, H., Bertino, E., Zhan, Z.-H., Cao, J., Zhang, Y., Zhang, J.: Evolutionary dynamic database partitioning optimization for privacy and utility. IEEE Transactions on Dependable and Secure Computing, 1–17 (2023). https://doi.org/10.1109/TDSC.2023.3302284
20. Shi, W., Chen, W.-N., Kwong, S., Zhang, J., Wang, H., Gu, T., Yuan, H., Zhang, J.: A coevolutionary estimation of distribution algorithm for group insurance portfolio. IEEE Transactions on Systems, Man, and Cybernetics: Systems. **52**(11), 6714–6728 (2022). https://doi.org/10.1109/TSMC.2021.3096013
21. Yang, J.-Q., Yang, Q.-T., Du, K.-J., Chen, C.-H., Wang, H., Jeon, S.-W., Zhang, J., Zhan, Z.-H.: Bi-directional feature fixation-based particle swarm optimization for large-scale feature selection. IEEE Transactions on Big Data. **9**(3), 1004–1017 (2023). https://doi.org/10.1109/TBDATA.2022.3232761
22. Wang, C., Sun, B., Du, K.-J., Li, J.-Y., Zhan, Z.-H., Jeon, S.-W., Wang, H., Zhang, J.: A novel evolutionary algorithm with column and sub-block local search for sudoku puzzles. IEEE Transactions on Games. **16**(1), 162–172 (2024). https://doi.org/10.1109/TG.2023.3236490
23. Tawhid, M.N.A., Siuly, S., Wang, K., Wang, H.: Automatic and efficient framework for identifying multiple neurological disorders from eeg signals. IEEE Transactions on Technology and Society. **4**(1), 76–86 (2023). https://doi.org/10.1109/TTS.2023.3239526
24. Siuly, S., Alçin, Ö.F., Wang, H., Li, Y., Wen, P.: Exploring rhythms and channels-based eeg biomarkers for early detection of alzheimer's disease. IEEE Transactions on Emerging Topics in Computational Intelligence. **8**(2), 1609–1623 (2024). https://doi.org/10.1109/TETCI.2024.3353610
25. Alvi, A.M., Siuly, S., Wang, H.: A long short-term memory based framework for early detection of mild cognitive impairment from eeg signals. IEEE Transactions on Emerging Topics in Computational Intelligence. **7**(2), 375–388 (2023). https://doi.org/10.1109/TETCI.2022.3186180

26. Hong, W., Yin, J., You, M., Wang, H., Cao, J., Li, J., Liu, M.: Graph intelligence enhanced bi-channel insider threat detection. In: International Conference on Network and System Security, pp. 86–102 (2022). https://doi.org/10.1007/978-3-031-23020-2_5. Springer
27. Morgado, C., Busichia Baioco, G., Basso, T., Moraes, R.: A security model for access control in graph-oriented databases. In: 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS), pp. 135–142 (2018). https://doi.org/10.1109/QRS.2018.00027
28. Shan, D., Du, X., Wang, W., Wang, N., Liu, A.: Kpi-hgnn: Key provenance identification based on a heterogeneous graph neural network for big data access control. Inf. Sci. **659**, 120059 (2024). https://doi.org/10.1016/j.ins.2023.120059
29. You, M., Yin, J., Wang, H., Cao, J., Wang, K., Miao, Y., Bertino, E.: A knowledge graph empowered online learning framework for access control decision-making. World Wide Web. **26**(2), 827–848 (2023). https://doi.org/10.1007/s11280-022-01076-5
30. Yin, J., Tang, M., Cao, J., You, M., Wang, H.: Cybersecurity applications in software: data-driven software vulnerability assessment and management. In: Emerging Trends in Cybersecurity Applications, pp. 371–389. Springer, Berlin (2022). https://doi.org/10.1007/978-3-031-09640-2_17
31. Huang, T., Gong, Y.-J., Kwong, S., Wang, H., Zhang, J.: A niching memetic algorithm for multi-solution traveling salesman problem. IEEE Trans. Evol. Comput. **24**(3), 508–522 (2020). https://doi.org/10.1109/TEVC.2019.2936440
32. Li, J., Zhan, Z., Wang, H., Zhang, J.: Data-driven evolutionary algorithm with perturbation-based ensemble surrogates. IEEE Transactions on Cybernetics. **51**(8), 3925–3937 (2021). https://doi.org/10.1109/TCYB.2020.3008280
33. Yin, J., Tang, M., Cao, J., You, M., Wang, H., Alazab, M.: Knowledge-driven cybersecurity intelligence: Software vulnerability coexploitation behavior discovery. IEEE Trans. Industr. Inf. **19**(4), 5593–5601 (2023). https://doi.org/10.1109/TII.2022.3192027
34. Li, J.-Y., Du, K.-J., Zhan, Z.-H., Wang, H., Zhang, J.: Distributed differential evolution with adaptive resource allocation. IEEE Transactions on Cybernetics. **53**(5), 2791–2804 (2023). https://doi.org/10.1109/TCYB.2022.3153964
35. Ge, Y., Orlowska, M., Cao, J., Wang, H., Zhang, Y.: Mdde: multitasking distributed differential evolution for privacy-preserving database fragmentation. VLDB J. **31**, 957–975 (2022). https://doi.org/10.1007/s00778-021-00718-w
36. Peng, M., Zhu, J., Wang, H., Li, X., Zhang, Y., Zhang, X., Tian, G.: Mining event-oriented topics in microblog stream with unsupervised multi-view hierarchical embedding. ACM Trans. Knowl. Discov. Data **12**, 1–26 (2018). https://doi.org/10.1145/3173044
37. Montanez, K.: Amazon Access Samples. UCI Machine Learning Repository (2011). https://doi.org/10.24432/C5JW2K
38. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems 30 (2017). https://doi.org/10.48550/arXiv.1706.02216
39. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The Semantic Web: 15th International Conference, ESWC 2018, pp. 593–607 (2018). https://doi.org/10.1007/978-3-319-93417-4_38 . Springer
40. Li, H., Wang, Y., Wang, H., Zhou, B.: Multi-window based ensemble learning for classification of imbalanced streaming data. World Wide Web. **20**, 1–19 (2017). https://doi.org/10.1007/s11280-017-0449-x
41. Yin, J., Tang, M., Cao, J., Wang, H., You, M., Lin, Y.: Vulnerability exploitation time prediction: an integrated framework for dynamic imbalanced learning. Word Wide Web. **25**, 401–423 (2021). https://doi.org/10.1007/s11280-021-00909-z
42. Lv, Q., Ding, M., Liu, Q., Chen, Y., Feng, W., He, S., Zhou, C., Jiang, J., Dong, Y., Tang, J.: Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1150–1160 (2021). https://doi.org/10.1145/3447548.3467350

Authors and Affiliations

Jiao Yin[1,2] · Guihong Chen[3,4] · Wei Hong[2] · Jinli Cao[1] · Hua Wang[2] · Yuan Miao[2]

✉ Jinli Cao
j.cao@latrobe.edu.au

Jiao Yin
j.yin@latrobe.edu.au

Guihong Chen
chenguihong@gpnu.edu.cn

Wei Hong
wei.hong2@live.vu.edu.au

Hua Wang
hua.wang@vu.edu.au

Yuan Miao
yuan.miao@vu.edu.au

1 Department of Computer Science and Information Technology, La Trobe University, Melbourne 3086, VIC, Australia

2 Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne 3011, VIC, Australia

3 School of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, Guangdong, China

4 School of Cyber Security, Guangdong Polytechnic Normal University, Guangzhou 510000, Guangdong, China