Knowledge and Information Systems (2025) 67:7987–8022
https://doi.org/10.1007/s10115-025-02476-5

RESEARCH

Beyond pairwise relationships: a transformer-based hypergraph learning approach for fraud detection

Venus Haghighi[1] · Behnaz Soltani[1] · Nasrin Shabani[1] · Jia Wu[1] · Yang Zhang[2] · Lina Yao[3,4] · Jian Yang[1] · Quan Z. Sheng[1]

Received: 23 February 2025 / Revised: 19 April 2025 / Accepted: 7 May 2025 /
Published online: 28 May 2025
© The Author(s) 2025

Abstract
Fraud detection in online networks has become increasingly challenging as fraudsters adopt sophisticated camouflage tactics to evade detection, making it imperative to combat their deceptive strategies. Graph-based fraud detection has gained significant attention in recent years, reflecting its growing potential to mitigate sophisticated fraudulent activities. The main objective of graph-based fraud detection is to distinguish between fraudsters and normal entities within graphs. While real-world networks contain complex, high-order relationships, existing graph-based fraud detection methods focus solely on pairwise interactions, overlooking non-pairwise relationships and the broader dependencies among entities within fraud graphs. Thus, we highlight the importance of exploring non-pairwise relationships to build a more effective fraud detection model. In this paper, we propose TROPICAL, a novel TRansfOrmer-based hyPergraph LearnIng framework for detecting CAmouflaged maLicious actors in online social networks. To capture comprehensive high-order relations, we construct a hypergraph from the original input graph. However, constructing the hypergraph can be computationally intensive. TROPICAL addresses this challenge by carefully selecting moderate hyperparameters, creating a balance between computational efficiency and effectively capturing high-order relationships. TROPICAL learns node representations by processing multiple hyperedge groups and incorporates positional encodings into the aggregated information to enhance their distinctiveness. The aggregated sequential information is then passed through a transformer encoder, enabling the model to generate rich, high-order representations to detect camouflaged fraudsters. Extensive experiments on two real-world datasets demonstrate TROPICAL's superior performance compared to the state-of-the-art fraud detection models. The source codes and the datasets of our work are available at https://github.com/VenusHaghighi/TROPICAL.

Keywords Hypergraph learning · Camouflage · Fraudster detection · Discriminative embedding

1 Introduction

The rapid expansion of the Internet and online services has led to a substantial increase in fraudulent activities. Fraudsters adopt increasingly sophisticated tactics, becoming more

Extended author information available on the last page of the article

deceptive and causing destructive impacts on businesses and online platforms, making the identification process more challenging [4, 8, 17, 37]. Social media, blogs, trading networks, and communication networks offer a vast space for fraudulent activities, further intensifying the challenge of detecting malicious behaviors [28, 29]. Recently, graph-based fraud detection approaches [1, 5, 21, 39], particularly graph neural network (GNN)-based ones [24, 25, 41], have gained significant attention in both academia and industry due to their ability to capture rich behavioral patterns among entities within graph-structured data. However, existing graph-based approaches neglect high-order relationships among entities, which refer to interactions involving more than two nodes or entities simultaneously, such as multiple users reviewing the same product or engaging in coordinated fraudulent behavior. These methods typically rely solely on pairwise connections, which limits their effectiveness in fully capturing the underlying correlations among entities within fraud networks [7, 35, 46].

There are two main challenges in the current GNN-based fraud detection models [42, 49]. Firstly, these models are vulnerable to the camouflage behavior of fraudsters, wherein the fraudsters deliberately establish many connections to normal entities [17, 18]. For example, spammers often use benign accounts to post their spam reviews or add links to well-liked items in a product review system. Fig 1 illustrates a typical attack model within an online product review network launched by camouflaged fraudsters. The online product review network consists of two different types of relations. Relation 1: Follower-Followee, where one user follows another user, and Relation 2: User-Product-User, where users who have reviewed the same products are linked. In this attack model, fraudsters create relational and feature inconsistencies to appear legitimate and trustworthy. Fraudster "U_5" posts false-positive reviews to imitate the behavior of legitimate user "U_1", creating feature inconsistencies despite having different labels. Fraudsters "U_5", "U_6", and "U_7" strategically form connections with normal users under the Follower-Followee relation while avoiding links with other fraudsters to conceal their activity, resulting in relation inconsistencies within the network. Under the User-Product-User relation, fraudsters establish connections with other fraudsters and normal users, creating relation inconsistencies within the network to make their activities appear more legitimate. This behavior alleviates the fraudsters' suspiciousness and eventually bypasses detection systems. In particular, in the GNN aggregation process, low-pass filtering is employed to retain the commonality of node features in the same neighborhoods. This aggregation strategy causes neighboring nodes to become increasingly similar, even if they belong to different class labels, i.e., fraudulent or benign. As a result, the model faces difficulty in distinguishing malicious nodes, significantly hindering the performance of fraud detection [13, 14, 48]. Secondly, beyond pairwise interactions, nodes within graphs often demonstrate complex higher-order interactions and dependencies. Existing fraud detection methods largely overlook these non-pairwise dependencies among entities in graphs. Incorporating these high-order dependencies into the graph representation learning process is crucial to attaining more comprehensive, expressive, and accurate node representations, ultimately improving the model's ability to detect fraudsters effectively.

In response to the challenges posed by camouflaged fraudsters within fraud graphs, several enhanced GNN-based fraud detection models have been proposed [7, 22, 35, 46, 47], which mainly focus on improving the aggregation procedure of GNN models by either employing the sampling strategies or by enhancing the filtering process during the aggregation phase. The sampling techniques aim to fulfill the homophily assumption, a fundamental aspect in designing GNN models, ensuring that representations of connected nodes are proximate to each other in the embedding space [49]. Consequently, nodes with dissimilar features are excluded from the aggregation process. The second group of approaches have discovered that the existence of anomalies leads to a '*right-shift*' phenomenon in the spectral energy distribu-

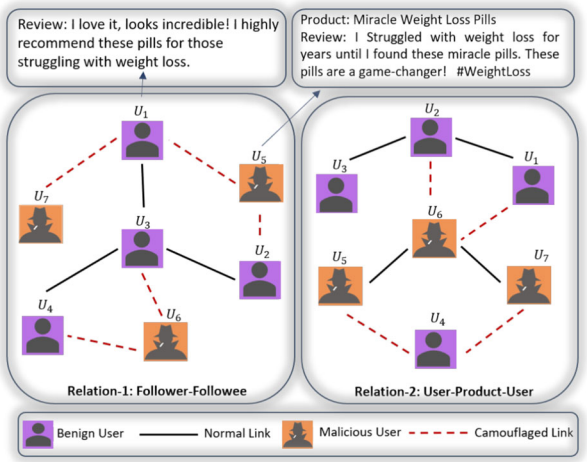


Fig. 1 A toy example of an attack model induced by camouflaged malicious actors in a multi-relational online product review network. The online product review network consists of two different types of relations, i.e., Follower-Followee and User-Product-User. Users and their comments are represented as nodes and node features in the online product review network

tion, where less concentration is on low frequencies and more on high frequencies [11, 12, 35, 40]. Hence, these approaches propose to extract different frequencies of information, including both low-frequency and high-frequency information, during the aggregation process to achieve discriminative embeddings [3]. Despite their success in detecting fraudulent entities, these proposed models fail to accurately characterize high-order interactions, thereby constraining their ability to capture and learn from complex and intricate relationships presented in fraud graphs.

Moreover, existing HyperGNN models, such as HGNN [9], HyperGCN [43], and HyperSAGE [2], rely on message passing mechanisms that aggregate features across hyperedges to learn node representations. However, their aggregation processes often lack the ability to generate discriminative embeddings for fraudsters and benign entities. Specifically, these models blend features of all connected nodes within hyperedges, causing fraudulent nodes to appear similar to normal nodes and hindering effective fraud detection. For instance, HGNN employs hypergraph Laplacian-based convolution, which smooths features indiscriminately across hyperedges, while HyperGCN reduces hyperedges to pairwise connections, losing critical high-order information. HyperSAGE's aggregation process, while scalable, fails to adequately separate fraudsters from benign nodes when their features are mixed. This lack of discriminative power is a fundamental limitation of existing HyperGNN models. To the best of our knowledge, no existing hypergraph-based models have been explicitly designed to detect camouflaged fraudsters in online networks. While various HyperGNN models have been proposed, such as HGNN, HyperGCN, and HyperSAGE, these models are generally designed for applications involving node classification and recommendation systems. They lack the ability to effectively detect camouflaged fraudsters who strategically create connections to blend with benign entities. This limitation motivates the development of a hypergraph-based fraud detection model capable of generating discriminative embeddings that effectively capture high-order relationships.

We delve into this problem and argue that learning from such high-order relationships is crucial but under-explored. To bridge this gap, we intend to answer the following two questions in this paper:

- How can we construct a hypergraph from the original input graph to effectively encode high-order data correlations and maximize the information for identifying fraudsters?

- How can we comprehensively learn from the constructed hypergraph to effectively detect camouflaged fraudsters?

To answer the first question, we employ a twofold approach to capture the structural and semantic relationships and patterns, both local and global, among entities in graphs. We begin by utilizing local structures to capture the immediate interactions. Next, we extend our focus to the global level, examining the broader structure and semantic patterns to uncover fraudulent relationships. Thus, a n-hop-based neighborhood and k-nearest neighbors strategies are applied to generate different hyperedge groups from the original simple graph. To address the second question, we propose TROPICAL, a novel TRansfOrmer-based hyPergraph learnIng framework for detecting CAmouflaged maLicious actors. TROPICAL aims to learn from the constructed hypergraph, generating low-dimensional sequential input embedding information. Subsequently, we employ a transformer encoder to produce more comprehensive final representations from sequential embedding information. In a nutshell, the main contributions of our work are summarized as follows:

- To effectively encode high-order information for fraudster detection, we employ a twofold strategy for hypergraph construction via capturing both local and global structural and semantic information among entities in graphs.
- We develop a novel transformer-based hypergraph learning framework TROPICAL for fraudster detection, which learns from the constructed hypergraph, captures high-order relations among entities, and generates rich representations for effective camouflaged fraudster detection.
- We extensively evaluate TROPICAL on two real-world fraud datasets to validate its performance. The experimental results demonstrate that TROPICAL outperforms the state-of-the-art methods.

The work reported in this paper is an extension of our previous work published at ICDM 2024 [14], with a more in-depth analysis of the methodology, detailed explanations of the model's components, and further visualization of the final representations to verify its effectiveness in detecting fraud within complex networks. The remainder of the paper is organized as follows. Section 2 discusses the relevant work in the literature. Sect. 3 describes our problem and its related definitions, and Sect. 4 discusses the motivation of our research. The technical details of our new proposed framework and the experimental results are presented in Sect. 5 and 6, respectively. Finally, Section 7 offers some concluding remarks.

2 Related work

In this section, we provide a summary of related studies on hypergraph neural networks and further explore research focused on graph-based fraud detection.

2.1 Hypergraph neural networks

The widespread success of various GNN models [41] across diverse applications, such as node classification, has demonstrated their strong ability to capture and reveal structural relationships within graph-structured data [25]. These models leverage message passing and aggregation mechanisms to learn node representations based on information exchange with neighboring nodes, effectively modeling local and pairwise structural dependencies within graphs. While this approach effectively captures local structural dependencies, it becomes

inadequate and imposes significant limitations in scenarios where dealing with complex interactions involving multiple entities simultaneously, rather than just two at a time. Consequently, GNNs struggle to fully capture and model high-order relationships, which are crucial for representing real-world complex networks, such as biological networks, financial transactions, multiuser collaborations, and fraud detection networks, where interactions often occur among groups of nodes rather than isolated pairs. This constraint ultimately limits the learned node representations, leading to suboptimal performance in tasks requiring a deeper understanding of intricate dependencies across multiple entities. To overcome this limitation, hypergraph learning has emerged as a powerful paradigm that generalizes traditional graph-based models by utilizing hyperedges, which connect multiple nodes within a single relationship. This approach enables the effective representation of intricate high-order relationships, making it particularly suitable for various domains, including genetic medicine [23], where biological interactions involve multi-way dependencies, and social recommendation systems [16], where user behaviors are influenced by collective group dynamics rather than just individual connections. As a result, hypergraph-based learning has gained significant attention as a solution to enhance learned representations in complex graph-based problems [44].

Hypergraph neural network (HGNN) [9] is the first work that extends GNNs to hypergraphs by designing a hyperedge convolution operation. Hyperedge convolution allows feature propagation across high-order relationships through hyperedge. The convolution operation in HGNN is formulated using a hypergraph Laplacian, where the node degree matrix is $\mathbf{D}_v$ and the hyperedge degree matrix is $\mathbf{D}_e$. The hypergraph convolution operation is defined as:

$$X' = \sigma\left(\mathbf{D}_v^{-1/2} H \mathbf{D}_e^{-1} H^T \mathbf{D}_v^{-1/2} X W\right), \tag{1}$$

where $X \in \mathbb{R}^{|V| \times d}$ represents the input node features, W is a learnable weight matrix, and σ is a nonlinear activation function suc as ReLU.

The convolution operation consists of three key steps: (1) propagating node features to their associated hyperedges via H^T, (2) aggregating information within each hyperedge using $\mathbf{D}_e^{-1}$, and (3) redistributing the updated information back to nodes through H, followed by normalization using $D_v^{-1/2}$. By leveraging higher-order dependencies, hyperedge convolution enables HGNN to effectively learn richer node representations, capturing intricate relationships beyond traditional pairwise connections in graphs. HGNN$^+$ [10] is the extended version of HGNN to handle and learn multi-type data correlations by directly concatenating the hypergraphs constructed from each single individual modality. HyperGCN [43] introduces a novel approach designed to extend graph convolutional networks on hypergraphs, thereby capturing complex high-order relationships inherent in many real-world networks. HyperGCN approximates each hyperedge with a set of pairwise edges, connecting the most disparate node within the hyperedge. This strategy, i.e., linear approximation, reduces the complexity of the hypergraph to a standard graph structure. To further enhance efficiency, FastHyperGCN [43], a variant that precomputes the hypergraph Laplacian matrix reducing training overhead and making the method scalable to large hypergraphs. HyperSAGE [2] presents an inductive learning framework to efficiently propagate information through hypergraphs. HyperSAGE employs a two-level message passing mechanism, enabling both intra-hyperedge and inter-hyperedge information flow. This approach allows for generalization to unseen nodes and scalability to large hypergraphs, making it particularly effective for dynamic and partially observed networks. Hypergraph neural networks have found a vast range of applications in multiple domains, such as genetic medicine [23] and social recommendation [16]. Nevertheless, its application in the domain of fraud detection remains

limited and relatively unexplored. To the best of our knowledge, this is the first work that leverages hypergraph learning to identify camouflaged malicious actors or fraudsters within online social networks.

2.2 Graph-based fraud detection

Considering the camouflage behavior of fraudsters in networks, where they strategically disguise their malicious activities to blend in with legitimate entities, there is an increasing focus on designing GNN-based models that exhibit robustness and resilience against these camouflaged fraudulent activities. In this section, we focus on previous GNN-based fraud detection models that have achieved some success in handling these camouflaged behaviors. Two general approaches have been proposed to enhance GNN-based fraud detection models against camouflaged fraudsters. We discuss each of these approaches in detail in the sequel.

2.2.1 Homophily-aware GNN-based fraud detection models

This group of approaches attempts to modify the graph structure during the aggregation process to meet the *homophily assumption*, a fundamental principle in the design of classic GNN models. Their objective is to generate more informative representations by transforming heterophilic local neighborhoods into homophilic ones. This is achieved through techniques such as sampling, which selectively eliminates heterophilic connections to refine the neighborhood aggregation process.

GraphConsis [22] is the first work that explores the graph inconsistencies problem induced by camouflaged fraudsters. To boost the aggregation process in GNN models, GraphConsis filters dissimilar neighbors for a given central node. In particular, GraphConsis is built on three techniques designed to simultaneously tackle three types of inconsistency problems, including context inconsistency, feature inconsistency, and relation inconsistency within a fraud graph. Firstly, to handle the context inconsistency of neighbors, GraphConsis assigns each node a trainable context embedding to represent the local structure of a given node, which helps distinguish the fraudulent nodes. Secondly, GraphConsis aggregates consistent neighbor embeddings based on a new metric that measures the embedding consistency between nodes. Incorporating the embedding consistency score into the aggregation process generates the sampling probability and ignores the neighbors with a low consistency score. Lastly, to alleviate the relation inconsistency, GraphConsis learns relation attention weights with neighbors and includes them in the aggregation process.

CARE-GNN [7] is another study inspired by homophily assumption to enhance the performance of GNN-based fraud detection models. CARE-GNN refines the graph structure before the aggregation process by filtering dissimilar neighbors of a given central node. The filtering process involves adjusting a threshold learned by the reinforcement learning technique. Additionally, CARE-GNN combines neighborhood information from different relations to obtain the final node representation. RioGNN [31] extends the CARE-GNN model by proposing a novel recursive and scalable reinforcement learning framework. This framework optimizes the filtering threshold for each relation and conducts the GNN training process in a general manner.

PC-GNN [20] designs a label-balanced sampler for selecting nodes and edges for training. The probability assigned to each node is inversely proportional to its label frequency, ensuring that nodes of the minority class are more likely to be selected. Consequently, the induced subgraph of the selected nodes will have a balanced label distribution. Next, they propose a

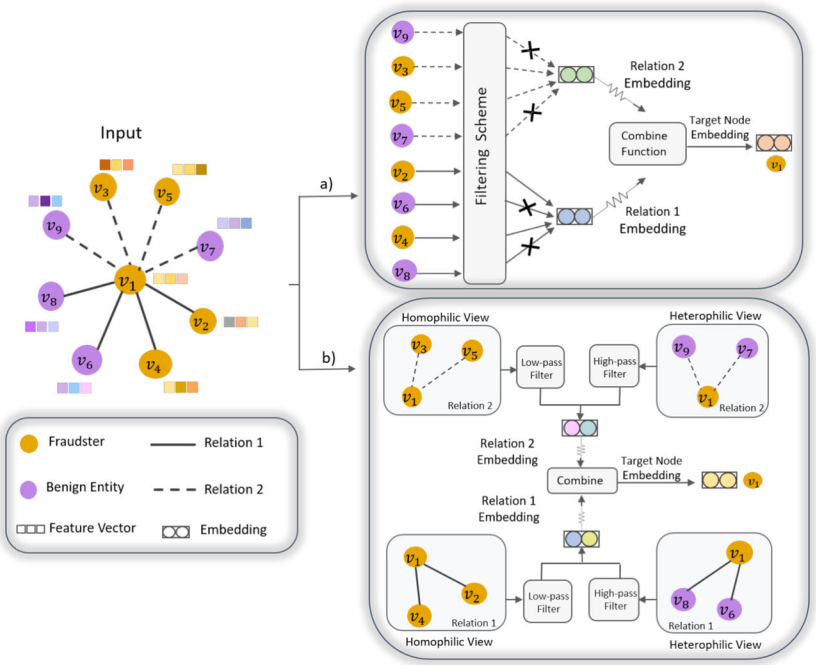


Fig. 2 Graphical illustration of two categories of previous methods for generating a comprehensive representation of a given target node, v_1 in a multi-relational fraud graph. **a)** The first category, homophily-aware GNN-based fraud detection models, lies in the filtering approach to follow the homophily assumption. Dissimilar neighbors are eliminated from the aggregation process through the filtering phase. **b)** The second category, heterophily-aware GNN-based fraud detection methods, utilizes different filtering schemes to create discriminate embedding for a target node. A high-pass filter is applied to heterophilic connections, and in contrast, a low-pass filter is adopted for homophilic connections

neighborhood sampler to choose neighbors using a learnable parameterized distance function. For the fraud target node, redundant links can be filtered by selecting neighbors that are distant from the target node, measured by distance, and removing them from the neighbor set. Necessary links that are beneficial for fraud prediction are created by selecting similar nodes of the fraud class and considering them as neighbors. Another study [34] conducts a comparative study of graph-based anomaly detection techniques in digital markets, focusing on both node statistics and graph embeddings. The paper proposes a lightweight method based on node and neighborhood features, which achieves competitive performance with lower computational cost. It also highlights scalability and interpretability as key challenges in embedding-based approaches.

2.2.2 Heterophily-aware GNN-based fraud detection models

Relying solely on the homophily assumption in GNN-based fraud detection can lead to a loss of discriminative information, which is crucial for detecting camouflaged fraudsters. As disassortativity (heterophilic connections) increases in a network, high-frequency information becomes essential to generate more discriminative embeddings [3]. To address this, recent

techniques integrate both low-frequency and high-frequency information into the neighborhood aggregation process, capturing both homophilic and heterophilic interactions.

FRAUDRE [47] is a fraud detection framework that addresses graph inconsistencies caused by camouflaged fraudsters. FRAUDRE enhances detection by aggregating neighborhood information and differences while integrating intermediate representations across neural network layers. Additionally, it introduces an imbalance-oriented classification module to handle the unequal distribution of normal users and fraudsters. H^2-FDetector [33] introduces a new information aggregation strategy to ensure that homophilic connections propagate similar information, while heterophilic connections propagate different information. It employs a GAT-based self-attention mechanism for homophilic connections to propagate similar features, while heterophilic connections leverage opposite representations for self-attention. This ensures that nodes with different labels (benign and fraudulent) remain distinct, while those sharing the same label become more similar.

BWGNN [35] analyzes graph anomalies through graph spectrum analysis and incorporates tailored spectral filters into GNN models to enhance fraud detection. It identifies the "right-shift" phenomenon in spectral energy distribution, where anomalies shift energy from low to high frequencies. This method observes that, in the presence of fraudulent or abnormal behavior, the spectral energy distribution of the graph signal undergoes a "right-shift" phenomenon. In a typical, well-behaved graph, most of the energy of the signal (e.g., node features) is concentrated in the low-frequency spectrum, indicating smoothness across connected nodes. However, the presence of anomalies, such as fraudulent nodes that behave differently from their neighbors, introducing sharp variations in the signal, causing the energy to shift toward higher frequencies. This shift reflects reduced homophily and smoothness in the graph structure. Spectral-based anomaly detection methods exploit this insight by designing filters that emphasize energy concentration in the high-frequency range, making them effective at identifying abnormal patterns. To address this, BWGNN leverages Hammond's graph wavelet theory to design a band-pass GNN architecture, effectively mitigating the impact of spectral shifts and improving the performance of fraud detection. COFRAUD [46] is a correlation-aware GNN-based fraud detection model that captures multi-relation interactions within fraud graphs to address the camouflage problem. It leverages two key statistical metrics, i.e., alienation and marginalization, to characterize fraudulent behaviors. Alienation refers to fraudsters isolating their interactions across multiple relations to evade detection, while marginalization reflects their tendency to place themselves in marginal positions in the network. By effectively modeling multi-relation correlation information, COFRAUD enhances fraud detection by distinguishing fraudulent nodes from normal users based on their structural and relational behaviors.

Despite the promising success achieved by previous studies, they primarily focus on pairwise relationships, overlooking the rich *non-pairwise* interactions that exist within complex graph structures. However, real-world fraud networks often involve high-order dependencies, where fraudulent behaviors are not limited to direct connections but extend across multiple interacting entities. As a result, there is an urgent need to effectively capture and learn these higher-order relations to generate more comprehensive and discriminative node representations. Moreover, directly applying existing HyperGNN models often fails to capture the camouflaged and deceptive behaviors inherent in fraud detection due to their limitations in aggregation mechanisms. These models typically suffer from a lack of generating discriminative representations due to feature aggregation processes that assimilate node characteristics within hyperedges without effectively distinguishing between fraudulent and benign entities. This indiscriminate aggregation process makes it harder to distinguish differences. Additionally, many HyperGNN models rely on hypergraph Laplacian-based convolution or simplify

hyperedges into pairwise relationships, which diminishes their capacity to capture intricate, high-order dependencies essential for identifying sophisticated fraudulent patterns. Our study stands as a pioneering effort in this direction by exploring and leveraging both structural and semantic non-pairwise interactions within graphs. By incorporating higher-order correlations, our approach enables a deeper understanding of fraud patterns, leading to more accurate and robust fraud detection.

Figure 2 provides a detailed overview of the two main categories of fraud detection models, i.e., homophily-aware and heterophily-aware approaches, highlighting their key distinctions and methodologies.

3 Preliminaries

In this section, we give the necessary definitions and related concepts, followed by a formal formulation of the research problem to be addressed.

3.1 Definitions

Multi-relation graph. Given a multi-relational graph, which can be denoted as $G = \{V, X, \{E_r|_{r=1}^{R}\}, Y\}$, where V is the set of nodes $\{v_1, v_2, ..., v_m\}$, and each node is associated with a d-dimensional feature vector $x_i \in R^d$. $X = \{x_1, x_2, ..., x_m\}$ is feature vector of all nodes. $E = \{E_1, E_2, ..., E_r\}$ is the edge set under R relations. $y_i \in Y$ is a binary label for a subset of labeled nodes, where $y_i = 0$ indicates benign nodes and in contrast $y_i = 1$ corresponds to fraudulent ones.

Relation-specific subgraph. We create distinct subgraphs for each relation type within the multi-relation graph. Each of these subgraphs, denoted by $G_{r_i}|_{i=1}^{r}$, contains nodes and relations of only one type.

Hypergraph. In contrast to a simple graph, an edge within a hypergraph connects multiple nodes rather than just two nodes. A hypergraph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V}$ and X are the node set and the corresponding node attributes, respectively. The edge degree and node degree matrices of $\mathcal{G}$ are indicated as $\mathbf{D}_e$ and $\mathbf{D}_v$. $\mathcal{G}$ can be denoted by an incidence matrix $\mathbf{H} = |\mathcal{V}| \times |\mathcal{E}|$, where $h(v, e) = 1$ if the node v belongs to hyperedge e, otherwise $h(v, e) = 0$. An illustrative comparison between a simple graph and a hypergraph is shown in Fig. 3.

3.2 Problem statement

Graph-based fraud detection. Given a multi-relational graph G, distinguishing fraudsters from normal nodes with the maximum accuracy is the main objective of the graph-based fraud detector. Fraud detection is defined as a binary node classification task as follows:

$$\begin{aligned} &f : V \rightarrow \{0, 1\}, \\ &max \sum_{v_i \in V_U} 1[f(v_i) = y_i], \end{aligned} \tag{2}$$

where f is the learnable function that predicts the labels of nodes, and $1[.]$ is the indicator function that is 1 if the condition is true and 0 otherwise. Only the labels of a subset of nodes

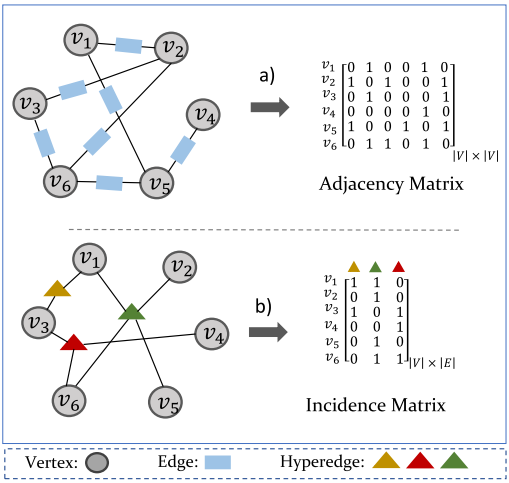


Fig. 3 Difference between simple graph and hypergraph. A simple graph represents pairwise connections using an adjacency matrix, while a hypergraph captures high-order relationships through an incidence matrix, where a single hyperedge can connect multiple nodes

are known. A fraud detector is trained based on the known labels, $v \in V_L$, and then the suspiciousness of unlabeled nodes, $v \in V_U$, can be predicted.

4 Motivation

To study the behavior patterns of fraudsters from the relation-specific subgraph structure, we leverage graph statistics on a real-world public dataset `Amazon` [27], which includes three types of relations: 1) U-P-U, 2) U-S-U, and 3) U-V-U (see Sect. 6.1). First, we utilize the following formula to measure the density of fraudsters from the structural spaces of relation-specific subgraphs.

$$\rho_n(v_i, n) = \sum_{v_i \in V \& y_i = 1} \frac{|\{v_j | v_j \in N_{hop_n}(v_i), y_j = 1\}|}{|N_{hop_n}(v_i)|} \tag{3}$$

Here $\rho_n(v_i, n)$ represents the density of fraudsters (structure-based) within the n-hop neighborhood of fraudulent node v_i, and $N_{hop_n}(v_i)$ denotes the set of nodes in the n-hop neighborhood of node v_i. We aim to reveal the relational patterns of fraudsters by expanding the n-hop neighborhood. As illustrated in Fig. 4 (a), the density of fraudsters increases as the n-hop neighborhood expands in different relation-specific subgraphs of the `Amazon` dataset. This indicates that fraudsters tend to establish connections with other distant fraudsters primarily within their 3-hop and 4-hop neighborhoods in different relation-specific subgraphs, but not beyond that. Therefore, inspired by this observation, we aim to extend our analysis beyond immediate or pairwise interactions. By creating non-pairwise relations among fraudsters (based on n-hop neighborhoods information), we can achieve a higher-order understanding of fraudsters' relational behaviors. The inclusion of more fraudsters in higher-order neighborhoods can lead to less heterophily neighborhood structure. This approach enables the generation of a more comprehensive representation paradigm for identifying fraudsters within fraud graphs.

Second, we analyze the density of fraudsters from the feature spaces on the `Amazon` fraud graph by utilizing a metric to measure the density of fraudsters based on K nearest neighbors

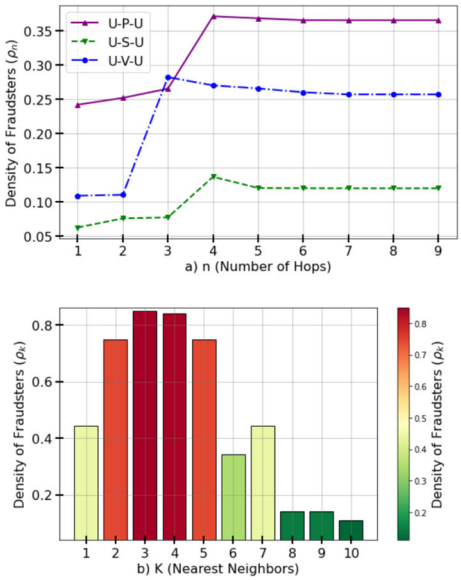


Fig. 4 **a)** The density of fraudsters versus the number of hops for different relations of the `Amazon` dataset. **b)** The density of fraudsters for various K values of nearest neighbors for the `Amazon` dataset. The plots illustrate that higher-order connections among fraudsters can be generated based on certain values of n and K, which has the potential to reveal the behavioral patterns of fraudulent activities effectively

as follows:

$$\rho_k(v_i, k) = \sum_{v_i \in V \& y_i = 1} \frac{|\{v_j | v_j \in N_{knn_k}(v_i), y_j = 1\}|}{|N_{knn_k}(v_i)|}, \tag{4}$$

where $N_{knn_k}(v_i)$ represents the set of nodes in the k-nearest neighbors of a fraudulent node v_i. As shown in Fig. 4 (b), we calculate the density of fraudsters for different values of k. It can be observed that $\rho_k(v_i, k)$ increases for certain values of k, indicating that extending beyond pairwise relationships by identifying k-nearest neighbors can effectively link more fraudsters. Consequently, this approach can be highly beneficial for generating more comprehensive node representations.

5 Methodology

In this section, we first give a brief overview of the TROPICAL framework and then describe the technical details of each component.

5.1 Framework overview

As depicted in Fig. 5, TROPICAL consists of several main components, each playing a crucial role in capturing complex fraud patterns and improving the detection process. Specifically, the hypergraph generation module constructs different hyperedge groups from the original multi-relational input graph by leveraging n-hop neighborhood and k-nearest neighbors information. This step enables the model to capture both local and global semantic and structural dependencies, ensuring that high-order relationships among entities are effectively represented. The hyperedge groups' aggregation module aggregates information from different constructed hyperedge groups to generate enriched sequential input features that encapsulate structural and relational patterns. To further enhance the expressiveness of these

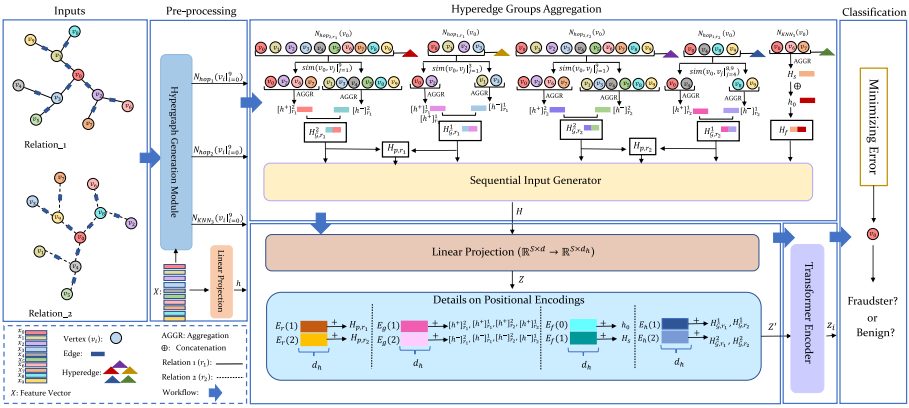


Fig. 5 The overview of the TROPICAL framework. TROPICAL takes a multi-relational graph as input. Firstly, the hypergraph generation module constructs different groups of hyperedges for each relation-specific subgraph. Secondly, the hyperedge groups' aggregation module aggregates information from different constructed hyperedge groups to generate rich sequential input features. Thirdly, the positional encodings module incorporates positional encodings into the sequential input features. Finally, the transformer encoder module learns the representations and the classification module aims at predicting the class of target node v_0 by minimizing the error of the loss function

features, the positional encodings module injects different types of positional encodings into the sequential input features, making the structural, relational, and feature-based characteristics more distinct. These enriched inputs are then fed into the transformer encoder module, which utilizes a self-attention mechanism to learn deep dependencies and generate final node representations for our model. Finally, the classification module leverages these learned representations to distinguish between fraudulent and normal nodes.

5.2 Hypergraph generation module

The construction of a hypergraph is the initial step in hypergraph learning, enabling the model to capture intricate high-order relationships among entities. To achieve this, we employ a twofold strategy to generate different hyperedge groups from the original multi-relational input graph. The first strategy focuses on structural correlations within graph data, where hyperedges are formed based on n-hop neighborhood connectivity. This ensures that entities with close structural relationships are effectively connected. The second strategy leverages the feature space of nodes, forming hyperedge groups based on similarity in node attributes using k-nearest neighbors. By incorporating both graph structure and feature similarity, this approach captures local and global interactions, allowing the model to learn rich and meaningful high-order dependencies among entities.

Hyperedge group using n-hop neighbors ($\mathcal{E}_{hop_n}$). The n-hop neighborhood information enables the model to effectively capture high-order structural relationships within the graph. The hyperedge group $\mathcal{E}_{hop_n}$ is constructed by identifying hyper vertices associated with a given central node v, based on n-hop reachable positions in the graph structure. Specifically, the n-hop neighborhood of a vetrix v_i is defined as:

$$N_{hop_n}(v_i) = \{v_j | A^n_{ij} \neq 0, v_j \in N(v_i)\} \tag{5}$$

where $N(v_i)$ is the neighbor set of v_i, and A^n is the n^{th} adjacency matrix of input graph G. The completed set of hyperedge group $\mathcal{E}_{hop_n}$ is then formulated as:

$$\mathcal{E}_{hop_n} = \{N_{hop_n}(v_i)|v_i \in V\}. \tag{6}$$

By expanding the search radius to include nodes up to n-hop away, $\mathcal{E}_{hop_n}$ enables the model to form hyperedges that preserve local and long-range structural dependencies, leading to more informative representations. To preserve low-order correlations, we consider a special case where $n = 1$, denoted as $\mathcal{E}_{hop_1}$. This ensures that each node retains its ego network, capturing its immediate neighbors and maintaining essential local structural correlations as fundamental information.

Hyperedge group using features ($\mathcal{E}_{knn_k}$). To leverage the feature similarity among nodes, we construct a second group of hyperedges, denoted as $\mathcal{E}_{knn_k}$, by identifying the K nearest neighbors of each node within the feature space. Unlike structural-based hyperedges, which rely on graph topology, this method focuses on attribute-based relationships, ensuring that nodes with similar feature representations are grouped together. For a given central node v_i, its K nearest neighbors in the feature space are linked through a hyperedge, forming a group of nodes that share similarity within feature space. The set of hyperedges $\mathcal{E}_{knn_k}$ is defined as follows:

$$\mathcal{E}_{knn_k} = \{N_{knn_k}(v_i)|v_i \in V\}. \tag{7}$$

where $N_{knn_k}(v_i)$ represents the set of K nearest neighbors of node v_i based on feature similarity. This approach groups nodes with similar feature characteristics into the same hyperedge, enabling the model to capture semantic relationships beyond just structural links.

Combination of hyperedge groups. The next step is to combine different hyperedge groups generated through the described strategies to construct the final hypergraph $\mathcal{G}$. Each hyperedge group captures distinct aspects of the data. $\mathcal{E}_{hop_1}$ preserve local neighborhood information, $\mathcal{E}_{hop_n}$ extend this to higher-order structural relationships, and $\mathcal{E}_{knn_k}$ incorporates feature-based similarities. To fully utilize this complementary information, we merge them into a unified hypergraph representation. This integration is achieved by concatenating the incidence matrices corresponding to each hyperedge set. We can directly concatenate the incidence matrices corresponding for each constructed hyperedge denoted as $\mathbf{H}_{hop_1}$, $\mathbf{H}_{hop_n}$, and $\mathbf{H}_{knn_k}$, to form the incidence matrix for the final hypergraph $\mathcal{G}$ as follows:

$$\mathbf{H}^{\mathcal{G}} = \mathbf{H}_{hop_1}||\mathbf{H}_{hop_n}||\mathbf{H}_{knn_k}. \tag{8}$$

Figure 6 provides an example illustrating the hypergraph generation module in detail, showing how it constructs different groups of hyperedges.

5.3 Hyperedge groups aggregation module

The neighborhood aggregation function of traditional GNN models such as GCN [19], GraphSAGE [15], and GAT [38] primarily collects information from local neighbors, which limits their ability to capture and generalize distant information across graphs. Such an aggregation strategy fails to identify fraudsters who employ camouflage techniques in fraud graphs. To overcome this limitation, we devise a hyperedge group aggregation strategy that enables access to distant neighbors and long-range dependencies, effectively capturing rich high-order information within fraud graphs.

Hyperedge group aggregation for $\mathcal{E}_{hop_n}$. We propose a discriminative enhanced aggregation strategy for the generated hyperedge group $\mathcal{E}_{hop_n}$ to highlight the distinct information of nodes

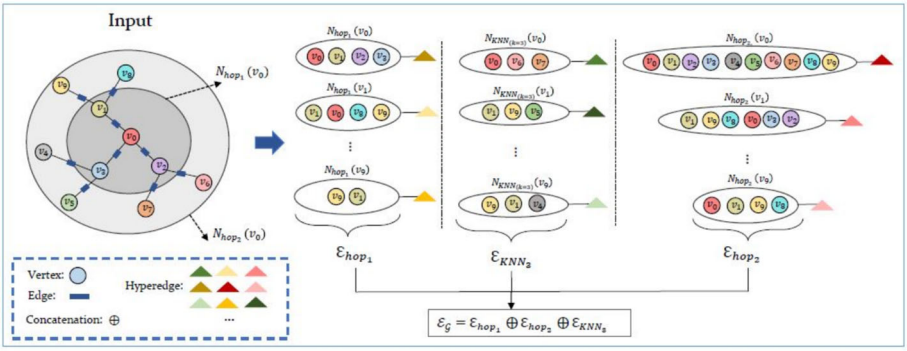


Fig. 6 An example illustrating how the hypergraph generation module constructs different groups of hyperedges to extract complementary information from both structural relationships and feature-based similarities

from different classes, i.e., fraudulent and benign. Its primary objective is to generate highly discriminative embeddings by maximizing the distance between the fraudulent and normal nodes, enabling our model to effectively distinguish between normal and fraudulent entities. Given a target node, v_i belongs to a hyperedge. We categorize hypernodes with similar feature space into one group and those with dissimilar features into another group. This allows performing separate aggregation within each group. We employ Cosine similarity as a pivotal metric for measuring the similarities between feature vectors of nodes in n-dimensional space as follows:

$$sim(v_i, v_j) = \{\frac{x_i . x_j}{\|x_i\| . \|x_j\|}, |v_j \in N_{hop_n}(v_i)\}, \tag{9}$$

where v_j is a neighbor of target node v_i and belongs to hyperedge $N_{hop_n}(v_i)$. x_i and x_j are feature vectors associated to nodes v_i and v_j, respectively.

Concretely, for a target node v_i and its n-hop neighbors under relation r, the nodes are divided into P groups as $V = \{V^+, V^-\}$. V^+ and V^- include similar and dissimilar hypernodes, respectively.

$$\mathbf{H_g^n} = [\mathbf{h}^+, \mathbf{h}^-]^{\mathbf{n}}, \tag{10}$$

where $\mathbf{H_g^n}$, named as *Group Encoding*, is the sequence of $P = 2$ discriminative embeddings generated by $[h^+]^n$ and $[h^-]^n$ as follows:

$$\begin{aligned} [\mathbf{h}^+]^{\mathbf{n}} &= F_{agg} = \frac{1}{|V^+|} \sum_{u \in V^+, i=1}^{2} x_u, \\ [\mathbf{h}^-]^{\mathbf{n}} &= F_{agg} = \frac{1}{|V^-|} \sum_{u \in V^-, i=1}^{2} x_u, \end{aligned} \tag{11}$$

where F_{agg} can be any permutation invariance aggregation function such as sum or mean. We adopt the mean function as F_{agg}.

We adopt the process of Hyperedge Group Aggregation (HGA) for $\mathcal{E}_{hop_1}$. Hence, for each target node v_i, we perform HGA for each individually generated $\mathcal{E}_{hop_1}$ and $\mathcal{E}_{hop_n}$ under relation r to produce its final representation as follows:

$$\mathbf{H_p} = [\mathbf{h_g^1} \oplus \mathbf{h_g^n}], \tag{12}$$

Algorithm 1: Hyperedge Group Aggregation in TROPICAL

Input: Feature matrix $X \in \mathbb{R}^{|V| \times d}$; Hyperedge sets $\mathcal{E}_{hop_n}$ and $\mathcal{E}_{knn_k}$; Similarity threshold θ.
Output: Sequential input H for transformer encoder.

Step 1: Hop Encoding (for $\mathcal{E}_{hop_n}$);
foreach *node* $v_i \in V$ **do**
  foreach *relation* r **do**
    $V^+ \leftarrow \{\}$; $V^- \leftarrow \{\}$;
    foreach $v_j \in N_{hop_n}(v_i)$ *under relation* r **do**
      if $sim(x_i, x_j) \geq \theta$ **then**
        $V^+ \leftarrow V^+ \cup \{x_j\}$;
      else
        $V^- \leftarrow V^- \cup \{x_j\}$;
    $h^+ \leftarrow \text{mean}(V^+)$; $h^- \leftarrow \text{mean}(V^-)$;
    $h_g^{(n)} \leftarrow [h^+ \| h^-]$;
    $h_p^{(r)} \leftarrow h_g^{(n)} \| h_g^{(1)}$;
  $H_p[i] \leftarrow \|_r h_p^{(r)}$;

Step 2: Feature Encoding (for $\mathcal{E}_{knn_k}$);
foreach *node* $v_i \in V$ **do**
  $N_k \leftarrow N_{knn_k}(v_i)$;
  $H_s \leftarrow \text{mean}(\{x_j \mid v_j \in N_k\})$;
  $h_i \leftarrow \sigma(W x_i)$;
  $H_f[i] \leftarrow [h_i \| H_s]$;

Step 3: Sequential Input Feature Generator;
foreach *node* $v_i \in V$ **do**
  $H[i] \leftarrow H_p[i] \| H_f[i]$;
return H

where $\oplus$ is the concatenation operation, and $\mathbf{H_p}$ is the *Hop Encoding* of target node v_i under relation r.

Hyperedge group aggregation for $\mathcal{E}_{knn_k}$. We introduce *Feature Encoding* for a given target node v_i belongs to the hyperedge group $\mathcal{E}_{knn_k}$ as follows:

$$\mathbf{H_s} = \sum_{u \in N_{knn_k}(v_i)} x_u, \tag{13}$$

where u is a neighbor of the target node v_i belongs to the K nearest neighbor set $N_{knn_k}(v_i)$ and x_u is the feature vector associated with node u. To further enhance the representation of the target node v_i, we incorporate self-attention by combining the raw feature vector of target node x_i with the aggregated neighbor features $\mathbf{H_s}$ to generate final *Feature Encoding* as follows:

$$h_i = \sigma(x_i W_i), \tag{14}$$

$$\mathbf{H_f} = [h_i] \oplus \mathbf{H_s}, \tag{15}$$

where h_i is the linear feature projection of target node v_i, obtained through a learnable weight matrix W_i and an activation function σ. Finally, we combine h_i with the aggregated neighbor features $\mathbf{H_s}$ using the concatenation operation to produce the final *Feature Encoding* $\mathbf{H_f}$. This

encoding preserves the target node's unique features while also integrating information from its K nearest neighbors, creating a richer representation in the hypergraph.

Sequential input feature generator. To construct the final input feature sequence, we combine all the encoded representations obtained from different processing steps. This ensures that the sequential input feature captures both structural and feature-based information from the generated hypergraph. The final input feature sequence is formulated as:

$$\mathbf{H} = \mathbf{H_p} \oplus \mathbf{H_f}. \tag{16}$$

where $\mathbf{H_p}$ captures the structural encodings derived from hyperedge groups $\mathcal{E}_{hop_1}$ and $\mathcal{E}_{hop_n}$, while $\mathbf{H_f}$ represents the feature-based encodings obtained from hyperedge group $\mathcal{E}_{knn_k}$. By combining these two encodings, we effectively integrate high-order structural relationships with sematic similarities, ensuring a more comprehensive and informative node representations.

The length of the input feature sequence is $S = R \times (P \times N) + 1 + 1$, where R is the number of relations in the multi-relational input graph, P is the number of classes for group aggregation, and N is the number of hops we adopt for $\mathcal{E}_{hop_n}$. It should be noted that the first one (i.e., 1) in the formula accounts for feature aggregation based on $\mathcal{E}_{knn_k}$, and the second one is for self-attention aggregation of the target node. To enhance clarity and reproducibility, we provide Algorithm 1, which details the step-by-step process of the hyperedge groups aggregation used in TROPICAL.

5.4 Positional encoding module

The generated sequential input consists of multiple segments derived from different relations, hop distances, node groups, and feature perspectives. However, these segments do not inherently encode positional information, making it difficult for the transformer to distinguish their structural and contextual roles. To enhance the distinctiveness of structural, relational, and feature-based information in the generated sequential input data ($\mathbf{H}$), we incorporate four types of positional encodings: i) hop positional encoding ($E_h(.)$), ii) relation positional encoding ($E_r(.)$), iii) group positional encoding ($E_g(.)$), and iv) feature positional encoding ($E_f(.)$). Each encoding plays a unique role in guiding the self-attention mechanism to learn more meaningful and discriminative representations. We first apply a linear transformation to the sequential feature data, ensuring that it is projected into a suitable representation space for further processing. This transformation is defined as follows:

$$\mathbf{Z} = \sigma(\varphi(\mathbf{H})), \tag{17}$$

where function $\varphi : \mathbb{R}^{S\times d} \rightarrow \mathbb{R}^{S\times d_h}$, σ is the activation function, and d_h is the final dimension of encoding.

Next, we introduce hop positional encoding to enable the encoder to capture and recognize multi-hop structural dependencies within the graph. This ensures that the model can distinguish nodes based on their hierarchical distance, allowing the model to effectively learn from both local and distant neighborhood interactions. Additionally, we incorporate relation positional encoding, which enhances the model's ability to differentiate between various types of relations within the fraud graph. Since fraud graphs often contain multiple relationship types, such as user-product interactions and user-user connections, encoding relational distinctions helps the model understand the role of each relationship. Furthermore, group positional encoding is applied to separate the aggregation outputs of different node groups, thereby preserving the unique characteristics of each group. Finally, we introduce

feature positional encoding to enhance the model's ability to distinguish between K nearest neighbor features and central node features.

The details of the positional encoding module can be found in Fig. 5. We employ learnable embeddings of different positional encodings as a lookup matrix. Each lookup matrix holds the learnable embeddings for different indices. When an index i is provided to the embedding layer, it retrieves the corresponding row $E(i)$ from the embedding matrix. Upon incorporating positional encodings, the final sequential input data takes the following form for target node v_i:

$$\begin{aligned}\mathbf{Z}' = [(\mathbf{H_{p,r_1}} + E_r(1))||...||(\mathbf{H_{p,r_n}} + E_r(n))] \\ \oplus [\mathbf{H_s} + E_f(1)] \oplus [h_i + E_f(0)]\end{aligned} \tag{18}$$

Hop positional encoding for relation r_i can be incorporated to Eq. 18 as follows:

$$\mathbf{H_{p,r_i}} = [(\mathbf{h^1_{g,r_i}} + E_h(1)) \oplus (\mathbf{h^n_{g,r_i}} + E_h(2))] + E_r(i). \tag{19}$$

Moreover, group positional encoding can be added to Eq. 19 as follows:

$$\begin{aligned}\mathbf{h_{g,r_i}} = [([\mathbf{h}^+ + E_g(1), \mathbf{h}^- + E_g(2)]^1_{r_i} + E_h(1)) \\ \oplus ([\mathbf{h}^+ + E_g(1), \mathbf{h}^- + E_g(2)]^n_{r_i} + E_h(2))] + E_r(i),\end{aligned} \tag{20}$$

5.5 Transformer encoder

We employ a transformer-based encoder to effectively learn representations from the final sequential input data, $\mathbf{Z}'$, as introduced in Sect. 5.4. This encoder enables the model to capture complex dependencies and contextual interactions within the hypergraph structure. Inspired by Graph Transformer Networks [45] and the original Transformer architecture [36], we incorporate multi-head attention mechanisms within the encoder to perform deep interactions among input vectors. The updating process of z_i for position i in the input data, $\mathbf{Z}'$, for layer $l+1$ is as follows:

$$z_i^{l+1} = Function(head_1, ..., head_h)W^l, \tag{21}$$

where $Function$ is the concatenation function, $z_i^0 = z_i$, h is the number of attention heads, and W^l is a projection to match the dimensions between adjacent layers. Each attention head independently captures relationships between nodes, allowing the model to assign different levels of importance to neighboring nodes. The attention mechanism operates by computing weighted interactions using query $Q^{m,l}$, key $K^{m,l}$, and value $V^{m,l}$ matrices, which are learnable parameters for each head at layer l. The formulation of an individual attention head is as follows:

$$\begin{aligned}head_h &= Attention(Q^{m,l}z_i^l, K^{m,l}z_j^l, V^{m,l}z_j^l), \\ &= \sum_{j \in S} w_{ij}(V^{m,l}z_j^l), \\ w_{i,j} &= softmax\left(\frac{Q^{m,l}z_i^l . K^{m,l}z_j^l}{\sqrt{d_h}}\right),\end{aligned} \tag{22}$$

where $Q^{m,l}$, $K^{m,l}$, and $V^{m,l}$ are the learnable weights of the h^{th} attention head, and $w_{i,j}$ is the attention scores using dot products followed by softmax, which determines the importance of node j in influencing the updated representation of node i. By aggregating information from multiple attention heads, the model effectively captures long-range dependencies and multi-hop relationships in the fraud graph. The concatenated outputs from all heads are then

projected using W^l to maintain dimensional consistency across layers. This multi-head self-attention mechanism allows the transformer-based encoder to model complex interactions and enhance the discriminative power of the learned representations.

5.6 Classification module

In TROPICAL, we adopt a weighted cross entropy loss function to train our model:

$$L_{CE} = -\sum_{v_i \in V} [\gamma y_i log(p_i) + (1 - y_i) log(1 - p_i)], \tag{23}$$

where γ is the training weight, which indicates the ratio of fraud labels to normal labels, and y_i is the label of node v_i. Eventually, we use the softmax function to compute the abnormal probability as:

$$p_i = softmax(z_i), \tag{24}$$

where p_i is the probability of the last layer embedding of target node v_i.

5.7 Computational complexity

In TROPICAL, we compute n-hop neighborhoods of nodes in a graph as $O(|V|c^n d)$, where c is the average degree of the graph, and $|V|$ is the number of nodes in the graph. We also compute k-nearest neighbors of nodes with $O(|v|kd)$, where d is the embedding dimension. We consider the dimension of query, key, and values as d_Q=d_K=d_V=d for the transformer encoder module. Therefore, the complexity of the self-attention layer is $O(S^2 d + Sd^2)$, where S is the length of the sequential input feature. The overall complexity of TROPICAL is $O(|V|c^n d + |v|kd + eh(S^2 d + Sd^2))$, where h is the number of attention layers, and e is the number of training epochs.

6 Experiments

To evaluate the effectiveness of TROPICAL and its potential benefits, we design a series of experiments to address the following research questions:

- **RQ1:** Is camouflage behavior prevalent among fraudsters in fraud datasets?
- **RQ2:** What is the performance of TROPICAL?
- **RQ3:** Does the TROPICAL model outperform the state-of-the-art baselines?
- **RQ4:** What is the performance of the TROPICAL model with respect to different hyper-parameters?
- **RQ5:** How does each component of TROPICAL contribute to the overall performance?
- **RQ6:** Is TROPICAL efficient in running time compared to up-to-date baselines?
- **RQ7:** Can TROPICAL effectively shorten the distance between fraudsters while extending the distance between fraudsters and normal entities?

Table 1 The statistical details of the datasets

Dataset	#Nodes (Fraud%)	#Features	Class	#Class	Relation	#Edges
YelpChi	45,954 (14.53%)	32	Positive	6,677	R-U-R	98,630
					R-T-R	1,147,232
			Negative	39,277	R-S-R	6,805,486
					All (homo)	8,051,348
Amazon	11,944 (6.87%)	25	Positive	821	U-P-U	351,216
			Negative	7,818	U-S-U	7,132,958
			Unlabeled	3,305	U-V-U	2,073,474
					All (homo)	9,557,648

6.1 Experimental setup

6.1.1 Datasets

We conduct a set of experiments on two real-world datasets, `Amazon` [27] and `YelpChi` [32] to validate the effectiveness of DEFEND. The statistical properties of these two datasets are shown in Table 1.

The `YelpChi` dataset includes spam and legitimate reviews on hotels and restaurants recommended by Yelp. In this dataset, Yelp's proprietary filtering algorithm is used to flag potentially fraudulent reviews, which are hidden from public views. Users are labeled as fraudulent if at least one of their reviews was filtered by Yelp, while users whose reviews were never filtered are labeled as legitimate. The `YelpChi` dataset includes two types of classes, i.e., positive and negative, which are counted for 6,677 and 39,277 samples, respectively. Nodes in the YelpChi dataset are reviews and the dataset consists of three relations as follows:

- R-U-R: It connects reviews posted by the same user.
- R-T-R: It connects two reviews under the same product in the same month.
- R-S-R: It links reviews under the same product with the same star rating (1-5 stars).

The `Amazon` dataset includes users and their reviews on musical instruments. In the `Amazon` dataset, users with less than 20% helpful votes are labeled as fraudsters. Therefore, we have three types of classes which include positive, negative, and unlabeled. These three types of classes are counted for 821, 7,818, and 3,305 samples, respectively. Users are represented as nodes and there are three kinds of connections among them as follows:

- U-P-U: It connects users who have reviewed one or more same products.
- U-S-U: It connects users who have issued at least one rating that is the same within 7 days.
- U-V-U: It connects users who have reviewed ranked in the top 5% in terms of TF-IDF similarity.

6.1.2 Implementation details

We implement our model using PyTorch [30] and implement the baseline models using the official codes provided by the authors and reproducing their results within our environment.

All experiments are conducted in Python 3.9.13, running on an NVIDIA Tesla P100 GPU with 480GB RAM.

For optimization, we utilize the Adam optimizer, setting the learning rate to 0.001 and the weight decay to 0.0001. The transformer encoder in our model consists of two layers for the `Amazon` dataset and three layers for the `YelpChi` dataset. Additionally, we configure the multi-head attention mechanism with four attention heads and apply a dropout rate of 0.1 to prevent overfitting.

6.1.3 Evaluation metrics

We employ AUC and Macro-F1 as evaluation metrics to assess the performance of all models. AUC represents the probability that a randomly selected positive sample (fraudulent node) is ranked higher than a randomly selected negative sample (benign node), effectively measuring the model's ability to distinguish between the two classes. Macro-F1, which computes the unweighted mean of the F1-score across all classes, is particularly valuable for evaluating model performance in scenarios where data imbalance is present. The F1-score is the harmonic mean of precision and recall, where precision measures the proportion of correctly identified fraudulent nodes among all predicted fraud cases, and recall indicates the proportion of actual fraudulent nodes correctly detected by the model. A higher Macro-F1-score reflects a better trade-off between precision and recall.

6.1.4 Baselines

We compare our proposed TROPICAL model with the state-of-the-art models to verify its effectiveness. Additionally, we follow the hyperparameters settings recommended in the respective papers to ensure a fair reproduction of the results. We categorize baseline models into traditional GNN models, HyperGNN models, and GNN-based fraud detection models.

The following models, which fall into the first category, serve as classic GNN models:

- GCN [19]: GCN is a spectral graph convolution model that defines efficient propagation rules using a first-order approximation of the Chebyshev polynomial [6]. It operates directly on graph data, leveraging message passing and sum aggregation to generate node representations.
- GAT [38]: GAT is a neural network model for graph-structured data that employs masked self-attention to assign adaptive weights to neighboring nodes, generating representations based on their importance.
- GraphSAGE [15]: GraphSAGE is an inductive GNN model that efficiently generates node embeddings for unseen data by sampling and aggregating features from local neighborhoods, enhancing scalability and flexibility.
- FAGCN [3]: FAGCN is a frequency-adaptive GNN with a self-gating mechanism that dynamically integrates low- and high-frequency signals during message passing for node representation learning.

The second category includes hypergraph neural network models, such as HGNN [9] and HyperGCN [43], which are designed for binary node classification on hypergraphs:

- HGNN [9]: HGNN introduces a hypergraph convolution operation that propagates node features through hyperedges, capturing high-order relationships. It leverages the hypergraph Laplacian and spectral-based convolution to generate node representations.

- HyperGCN [43]. HyperGCN approximates each hyperedge with a set of pairwise edges, effectively capturing complex relationships. It introduces a medoid-based aggregation strategy to efficiently propagate node features while reducing computational complexity, making it appropriate for large-scale hypergraph learning.

The last category consists of GNN-based fraud detection models, which are specifically designed to identify fraudulent entities in low-homophily settings:

- GraphConsis [22]: GraphConsis integrates a consistency score into the aggregation process to filter low consistency neighbors and determine sampling probabilities. Additionally, it learns attention weights for neighbors to mitigate relation inconsistency.
- CARE-GNN [7]: CARE-GNN is a fraud detection model that filters dissimilar neighbors during the aggregation process by a learnable threshold. It leverages reinforcement learning (RL) to adaptively find the optimal neighbor selection threshold.
- PC-GNN [20]: PC-GNN is a GNN-based imbalance learning model that proposes a label-balanced sampler to pick nodes and edges to train so that nodes of the minority class are more likely to be picked. It integrates two stages of graph sampling and neighbor selection into general GNN frameworks to boost the performance of fraud detection.
- FRAUDRE [47]: FRAUDRE is a fraud detection model that is dual resistant to the camouflage and imbalance nature of fraudsters. It proposes a fraud-aware graph convolution model to embed the graph topologies to represent the dissimilarity of neighboring users.
- H^2-FDetector [33]: H^2-FDetector is a fraud detection model that leverages both homophilic and heterophilic connections to enhance node embedding generation. It introduces a novel aggregation strategy, ensuring that homophilic connections propagate similar information, while heterophilic connections emphasize differences.
- BWGNN [35]: The BWGNN model proposes beta wavelet graph neural network based on Hammond's graph Wavelet theory to tackle the right-shift phenomenon of graph anomalies. BWGNN uses different wavelet kernels in parallel and then aggregates the corresponding filtering results.
- COFRAUD [46]: The COFRAUD model is a correlation-aware fraud detection model that incorporates synergistic camouflage into fraud detection. It proposes two distinctive features of fraudsters, i.e., alienation and marginalization, to capture the correlation among multi-relation fraudulent behaviors.

It is worth mentioning that we follow these papers to choose hyperparameters for the regeneration of results.

6.2 Camouflage evidence (RQ1)

We analyze the camouflage behavior of fraudsters by introducing two metrics to measure both local and global heterophily rates in a given multi-relational graph G. Fraudsters often attempt to integrate into benign communities by forming heterophilic connections, making them harder to detect using traditional GNN-based methods. To quantify this behavior, we define the global heterophily rate, $H^G_{edge}(G_r)$, which calculates the proportion of edges that connect nodes from different class labels, i.e., fraudulent and benign nodes, under a specific relation r. A higher value of $H^G_{edge}(G_r)$ indicates a greater tendency for fraudsters to camouflage themselves within normal users, making fraud detection more challenging. This metric is defined as:

$$H^G_{edge}(G_r) = \frac{|\{e_{ij}|e_{ij} \in E_r, y_i \neq y_j\}|}{|E_r|} \tag{25}$$

Table 2 Global and local heterophily ratio statistics under different relations

Dataset	Relation	#Edges	$H^G_{edge}(G_r)$	$H^L_{node}(G_r)$
YelpChi	R-U-R	98,630	0.10	0.09
	R-T-R	1,147,232	0.95	0.82
	R-S-R	6,805,486	0.95	0.81
	All (homo)	8,051,348	0.93	0.81
Amazon	U-P-U	351,216	0.81	0.83
	U-S-U	7,132,958	0.96	0.94
	U-V-U	2,073,474	0.97	0.95
	All (homo)	9,557,648	0.95	0.92

where E_r is the edge set under relation r in the multi-relational graph G, y_i is the node label for node v_i, and e_{ij} is the edge between node v_i and v_j. To confirm the existence of camouflage behavior, we compute $H^G_{edge}(G_r)$ for both the `YelpChi` and `Amazon` datasets. A high score of $H^G_{edge}(G_r)$, $(\rightarrow 1)$, indicates that fraudsters largely connect to benign nodes, successfully blending into normal communities and evading detection.

In addition to global heterophily, we introduce a local heterophily metric, $H^L_{node}(G_r)$, which evaluates the extent of heterophilic connections for an individual fraudulent node v_i within its local neighborhood under a specific relation r. This metric provides a node-level perspective on how fraudsters position themselves in the graph, revealing structural patterns that contribute to their deceptive behavior. It is defined as follows:

$$H^L_{node}(G_r) = \frac{1}{|V|} \sum_{v_i \in V \& y_i = 1} \frac{|\{v_j | v_j \in N_{v_i}, y_i \neq y_j\}|}{d_{v_i}} \tag{26}$$

Here, $H^L_{node}(G_r)$ calculates the average number of heterophilic connections in local neighborhoods of fraudulent nodes in the graph G. A high value indicates that fraudsters maintain a significant number of connections to benign users, further employing their camouflage strategy. Table 2 summarizes these statistics for both the `YelpChi` and `Amazon` datasets. Our findings reveal that fraudulent nodes form over 80% heterophilic connections, except in the R-U-R relation of `YelpChi`. This confirms that camouflage behavior is a widespread tactic among fraudsters in fraud graphs, allowing them to seamlessly blend into normal communities.

6.3 Performance comparison (RQ2, RQ3)

We compare the performance of TROPICAL with the baseline methods. The results in table demonstrate that TROPICAL surpasses all other GNN models. It is worth highlighting that GCN, GAT, GraphSAGE, and FAGCN are designed for single relation input graphs, whereas other models are specifically developed to handle multi-relational fraud graphs. We observe that directly applying traditional GNN models for fraud detection fails to achieve promising results. Traditional GNN models function as low-pass filters during the aggregation process, retaining commonality among nodes and reducing the distinction between fraudsters and benign entities. As a result, they perform poorly in low-homophily settings, such as fraud detection.

While FAGCN outperforms traditional GNN models by incorporating both high-frequency and low-frequency channels in the aggregation process, its effectiveness remains limited in low-homophily settings like fraud detection. There is still significant room for improvement to further enhance its ability to capture complex relational structures and better differentiate fraudulent entities in fraud graphs. We employ HGNN and HyperGCN to perform learning on constructed hypergraphs from original graphs for `Amazon` and `YelpChi`. Both HGNN and HyperGCN models outperform traditional GNN models, indicating the significance of learning on high-order graph structures. GraphConsis, CARE-GNN, and PC-GNN achieve promising results in low-homophily settings by filtering dissimilar neighbors from the aggregation process to better align with the homophily assumption. These models effectively transform a low-homophily setting into a high homophily one by prioritizing similar neighbors, which allows the model to capture more informative node representations and enhances the learning process. FRAUDRE, H^2-FDetector, COFRAUD, and BWGNN enhance the aggregation process by integrating high-frequency information and stacking multiple modules to obtain cross-correlation information in multi-relational fraud graphs. However, their effectiveness remains limited to pairwise relationships, restricting their ability to fully capture higher-order interactions in complex fraud graphs. Our proposed model, TROPICAL, surpasses GNN-based fraud detection models in `YelpChi`. For `Amazon`, TROPICAL achieves better results in terms of Macro-F1 and demonstrates a comparable performance to BWGNN in terms of AUC. The results show that TROPICAL is capable to extract and learn rich high-order information among entities, thereby successfully overcoming the camouflage issue Table 3.

6.4 Parameter analysis (RQ4)

Impact of embedding sizes and several transformer encoder layers. We investigate the sensitivity and performance of TROPICAL under different values of hyperparameters. We adjust the embedding sizes of all learnable embeddings from 16, 32, 64, and 128 to explore the degree of TROPICAL sensitivity. These values are commonly used in the graph learning and offer a practical range for studying the trade-off between expressiveness and computational cost. This range enables a comprehensive evaluation of the model's sensitivity to representation capacity. We evaluated the stability of TROPICAL across different embedding sizes (16, 32, 64, 128) by measuring the standard deviation of AUC and Macro-F1-scores. The results show that the model exhibits strong stability on YelpChi and acceptable stability on Amazon, with standard deviations remaining within a practical threshold of $1 - 2\%$. These findings indicate that TROPICAL performs consistently across embedding configurations.

Moreover, we set different numbers of transformer encoder layers for TROPICAL. The results shown in Fig. 7 indicate that TROPICAL maintains acceptable stability with regard to different hyperparameters for both `YelpChi` and `Amazon` datasets. Moreover, we conducted grid search experiments by varying the number of encoder layers and attention heads. Our results show that using three layers and four attention heads for `YelpChi` and two layers with four heads for `Amazon` achieves the best performance. These configurations yield superior results in terms of AUC and Macro-F1. First, deeper transformer architectures tend to overfit due to the inherent sparsity, noise, and heterophily present in fraud graphs, especially when input sequences are relatively short. Using two or three layers provides sufficient capacity to capture complex high-order dependencies without introducing excessive complexity. Second, we have found that four attention heads provide a good balance between expressiveness (the learned representations) and stability, allowing the model to simultaneously attend to different

Table 3 Performance comparison on YelpChi and Amazon datasets under 10% and 40% training data. Results are reported as mean ± std

Method	YelpChi (10%)		YelpChi (40%)		Amazon (10%)		Amazon (40%)	
	AUC	Macro-F1	AUC	Macro-F1	AUC	Macro-F1	AUC	Macro-F1
GCN	0.5080 ± 0.000	0.4608 ± 0.000	0.5338 ± 0.000	0.4608 ± 0.000	0.7736 ± 0.002	0.4751 ± 0.000	0.7791 ± 0.002	0.4751 ± 0.000
GAT	0.5013 ± 0.000	0.4608 ± 0.000	0.4993 ± 0.000	0.4608 ± 0.000	0.7743 ± 0.002	0.4751 ± 0.000	0.7785 ± 0.002	0.4751 ± 0.000
GraphSAGE	0.5639 ± 0.000	0.4608 ± 0.000	0.5812 ± 0.000	0.4608 ± 0.000	0.6968 ± 0.002	0.4751 ± 0.000	0.7134 ± 0.002	0.4751 ± 0.000
FAGCN	0.7430 ± 0.003	0.5589 ± 0.006	0.7683 ± 0.004	0.5601 ± 0.006	0.8621 ± 0.004	0.8176 ± 0.008	0.8705 ± 0.003	0.8236 ± 0.007
HGNN	0.7554 ± 0.002	0.6509 ± 0.005	0.7654 ± 0.003	0.6669 ± 0.005	0.8356 ± 0.003	0.7189 ± 0.006	0.8438 ± 0.003	0.7196 ± 0.006
HyperGCN	0.7657 ± 0.002	0.6998 ± 0.004	0.7883 ± 0.003	0.7028 ± 0.004	0.8241 ± 0.003	0.7250 ± 0.005	0.8358 ± 0.003	0.7344 ± 0.005
GraphConsis	0.6401 ± 0.002	0.6113 ± 0.004	0.6156 ± 0.003	0.6289 ± 0.004	0.8205 ± 0.003	0.7546 ± 0.007	0.8515 ± 0.003	0.7765 ± 0.005
CARE-GNN	0.7234 ± 0.003	0.5867 ± 0.005	0.7356 ± 0.004	0.6081 ± 0.006	0.8816 ± 0.004	0.8821 ± 0.006	0.8736 ± 0.004	0.8836 ± 0.005
PC-GNN	0.7804 ± 0.002	0.6591 ± 0.006	0.7943 ± 0.003	0.6721 ± 0.005	0.9231 ± 0.003	0.8177 ± 0.009	0.9305 ± 0.003	0.8621 ± 0.006
FRAUDRE	0.7147 ± 0.003	0.5958 ± 0.004	0.7393 ± 0.004	0.6165 ± 0.005	0.8921 ± 0.004	0.9098 ± 0.006	0.8998 ± 0.004	0.9102 ± 0.004
H^2-FDetector	0.8312 ± 0.003	0.6904 ± 0.005	0.8467 ± 0.003	0.6988 ± 0.005	0.9288 ± 0.002	0.8312 ± 0.008	0.9519 ± 0.002	0.8396 ± 0.006
BWGNN	0.8488 ± 0.002	0.7167 ± 0.005	0.8889 ± 0.002	0.7598 ± 0.004	0.9363 ± 0.002	0.8372 ± 0.007	**0.9624 ± 0.001**	0.9126 ± 0.003
COFRAUD	0.8314 ± 0.003	0.7065 ± 0.004	0.8869 ± 0.003	0.7702 ± 0.004	0.9258 ± 0.003	0.8932 ± 0.005	0.9503 ± 0.002	0.9096 ± 0.004
TROPICAL	**0.8925 ± 0.006**	**0.7567 ± 0.008**	**0.9026 ± 0.006**	**0.7764 ± 0.008**	**0.9456 ± 0.011**	**0.9164 ± 0.022**	0.9526 ± 0.002	**0.9264 ± 0.002**

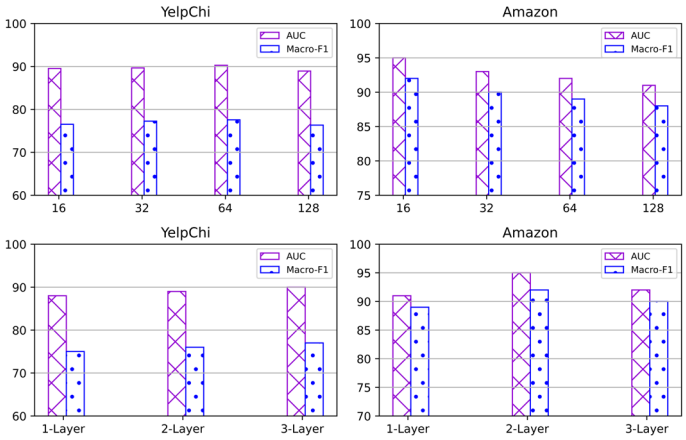


Fig. 7 Parameter Analysis. The performance of TROPICAL on different embedding sizes (Row 1) and different numbers of transformer encoder layers (Row 2)

perspectives, including structural hops, relation types, group similarity, and feature-based distinctions. Adding more heads introduces redundancy and sometimes unstable attention patterns, while fewer heads limits the model's ability to leverage diverse information sources. Finally, this configuration maintains computational efficiency, making the model scalable to large datasets, and is aligned with common studies in transformer-based models for graph and sequence learning.

Impact of parameters n and k for generating $\mathcal{E}_{hop_n}$ and $\mathcal{E}_{knn_k}$. The hyperparameters n and k are crucial in the hypergraph generation module, serving as a preprocessing step for the entire framework. To determine the optimal values for these parameters, we employ a grid search strategy, varying both n and k from 2 to 7. This range is selected based on the motivation of capturing both local and global interactions within the graph, as fraudsters tend to establish distant connections that blend them with benign entities (see Sect. 4). Increasing the hop size n allows the model to explore deeper connections within the graph, capturing complex relationships beyond immediate neighbors. Similarly, increasing k helps identify additional fraudulent nodes that are semantically similar, further enriching the constructed hyperedges. This dual mechanism captures both structural and semantic relationships, which is critical for accurately distinguishing fraudsters from benign nodes. Therefore, the grid search approach provides a thorough exploration of the hyperparameter space by evaluating all possible combinations within a predefined range. This method aims to achieve an optimal balance between effectively capturing higher-order dependencies and maintaining computational efficiency.

Figure 8 presents the AUC and Macro-F1-scores of TROPICAL for both fraud datasets. Both parameters n and k are varied from 2 to 7 to investigate which combination leads to better AUC and Macro-F1. We observe that for `YelpChi`, the combination $n = 3$ and $k = 4$ leads to better outcomes, while for `Amazon`, the combination $n = 4$ and $k = 5$ produces better results. The proposed TROPICAL therefore adopts these specific values for the hypergraph generation module in `YelpChi` and `Amazon`.

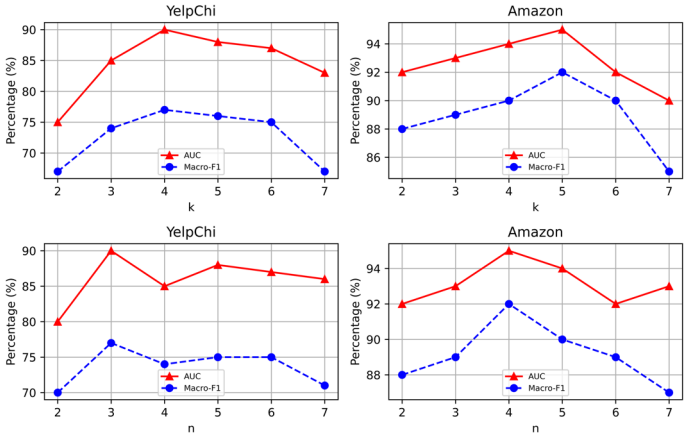


Fig. 8 Parameter Analysis. The performance of TROPICAL for different values of parameters k (Row 1) and n (Row 2)

Table 4 Ablation analysis of encoding modules in TROPICAL

Encoding Module	AUC		F1	
	YelpChi	Amazon	YelpChi	Amazon
$\mathbf{H_g}, \mathbf{H_p}, \mathbf{H_f}$	**0.9026**	**0.9526**	**0.7764**	**0.9264**
$\mathbf{H_g}, \mathbf{H_p}$	0.8333	0.9114	0.7365	0.8914
$\mathbf{H_g}, \mathbf{H_f}$	0.7941	0.9080	0.7308	0.8847
$\mathbf{H_p}, \mathbf{H_f}$	0.8412	0.9236	0.7454	0.8890
$\mathbf{H_g}$	0.7612	0.8894	0.7105	0.8629
$\mathbf{H_p}$	0.7768	0.8801	0.7036	0.8569
$\mathbf{H_f}$	0.7815	0.8962	0.7110	0.8654
-	0.7516	0.8469	0.6883	0.7248

6.5 Ablation study (RQ5)

We conduct a set of ablation studies to demonstrate the effectiveness of each individual component in TROPICAL. There are six variants: 1) w/o GE: omits the Group Encoding strategy, and instead, aggregates all nodes corresponding to a specific hop as a single group using a mean function; 2) w/o HE: excludes n-hop encoding information; therefore, we exclusively utilize 1-hop neighborhood information in the Hop Encoding process; 3) w/o FE: eliminates k-nearest neighbors Feature Encoding in the input sequence so we only consider the central node feature in the Feature Encoding process; 4) w/o PE: removes the Positional Encoding module so we employ sequential data without any positional encoding as the input of the transformer module; 5) w/o TE: replaces the Transformer Encoder module with other encoders such as LSTM module, and finally 6) TROPICAL: the primary version that stacks all the components together.

As illustrated in Fig. 9, TROPICAL outperforms all other variants, demonstrating the significance of each model component in contributing to overall performance. The training ratio is set to 0.4, ensuring a fair evaluation across datasets. Among all ablation variants, the model without the Transformer Encoder (w/o TE) exhibits the worst performance, empha-

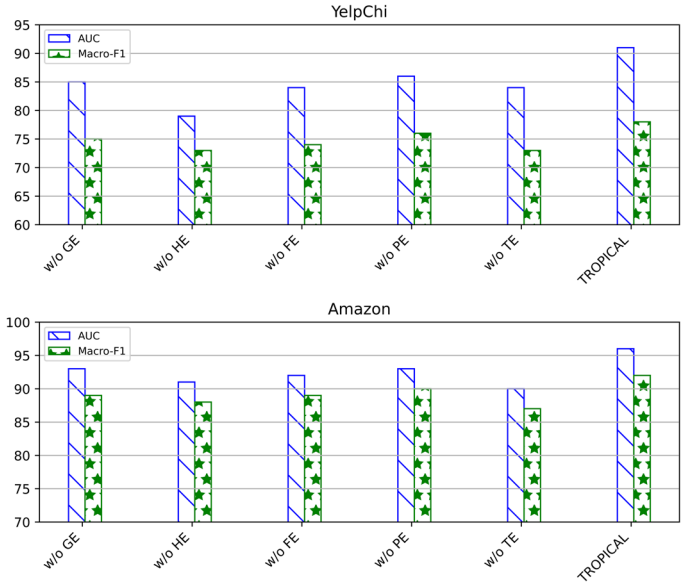


Fig. 9 The ablation analysis of the TROPICAL approach on `YelpChi` and `Amazon`

sizing the critical role of the transformer encoder in learning effective node representations. Specifically, removing the transformer encoder leads to a notable drop of 6.43% and 5.87% in AUC for `YelpChi` and `Amazon`, respectively. Similarly, the Macro-F1-score declines by 5.21% on `YelpChi` and 4.87% on `Amazon`, indicating the importance of transformer-based learning in fraud detection. TROPICAL integrates a multi-head attention mechanism within the transformer encoder, allowing the model to adaptively reweigh group vectors based on their relevance. This mechanism significantly enhances the capture of semantic information and strengthens the model's ability to differentiate between fraudulent and benign entities.

Furthermore, the results from the ablation study prove that Hop Encoding and Feature Encoding are effective in capturing distant and non-pairwise information within fraud graphs. For `YelpChi`, Hop Encoding contributes 10.76% and 5.43% improvements in AUC and Macro-F1, respectively. The w/o FE variant demonstrates the effectiveness of Feature Encoding, as the AUC decreases by 6.93% and 4.12% for the `YelpChi` and `Amazon` datasets, respectively. Additionally, the performance of w/o PE drops by 3.08% and 4.19% in AUC for the `Amazon` and `YelpChi` datasets, indicating the capability of the Positional Encoding module to enhance the distinctiveness of the generated input features, leading to better final results.

We further evaluate the effectiveness of different encoding modules, including Group Encoding (GE), Hop Encoding (HE), and Feature Encoding (FE), by incorporating various combinations of these encoding modules into TROPICAL. Table 4 shows that TROPICAL achieves the best performance with all the encoding modules (Row 1). The variant without the complete set of encoding modules (i.e., the last row) performs worse compared to other variants, demonstrating the effectiveness of all learnable encoding modules in TROPICAL.

Table 5 Training time cost of TROPICAL and the state-of-the-art fraud detection models in seconds

YelpChi	Training percentage			
	10%	20%	30%	40%
FRAUDRE	11.02	14.53	17.85	18.47
CARE-GNN	3.43	4.93	6.71	8.26
BWGNN	2.75	3.16	3.37	3.39
COFRAUD	2.83	3.01	3.27	3.29
TROPICAL	3.16	4.12	4.59	5.17
Amazon	Training Percentage			
	10%	20%	30%	40%
FRAUDRE	5.03	9.62	12.81	15.34
CARE-GNN	0.58	0.76	0.81	0.89
BWGNN	0.31	0.36	0.37	0.39
COFRAUD	0.29	0.28	0.26	0.21
TROPICAL	0.45	0.56	0.61	0.68

6.6 Time efficiency (RQ6)

This experiment focuses on time efficiency by comparing TROPICAL with four state-of-the-art fraud detection models, CARE-GNN [7], FRAUDRE [47], BWGNN [35], and COFRAUD [46]. We calculate the average training time per epoch while adjusting the training ratio from 10% to 40%. Furthermore, we set the embedding size of the hidden layer to 16 and 64 for all methods on the `Amazon` and `YelpChi` datasets, respectively. As depicted in Table 5, TROPICAL runs significantly faster than CARE-GNN and FRAUDRE on both datasets. Compared to the recent high-performing models, i.e., BWGNN and COFRAUD, TROPICAL demonstrates a comparable running efficiency, while achieving better performance (see Sect. 6.3).

6.7 Visualization (RQ7)

We perform a visualization analysis to assess how effectively fraud detection models separate fraudulent and benign nodes, i.e., positive and negative samples. Since fraud detection relies on learning meaningful node representations, it is crucial to evaluate how well models can distinguish fraudulent entities from legitimate ones. To explore this capability, we visualize the final node embeddings of several state-of-the-art fraud detection models, including CARE-GNN [7], FRAUDRE [47], H^2-FDetector [33], BWGNN [35], COFRAUD [46], and TROPICAL [14], using the `YelpChi` dataset.

For this visualization, we utilize t-SNE [26], a widely adopted dimensionality reduction technique, to project high-dimensional node embeddings into a 3D space. A 3-dimensional representation provides a clearer perspective on how well each fraud detection model separates different node classes within the fraud graph. To ensure a balanced and insightful visualization, we focus on the test set and maintain an equal number of positive and negative samples.

From Fig. 10, it is evident that TROPICAL achieves the most distinct separation between fraudulent nodes (positive samples) and benign nodes (negative samples), demonstrating its ability to generate highly discriminative node embeddings. In contrast, CARE-GNN, which

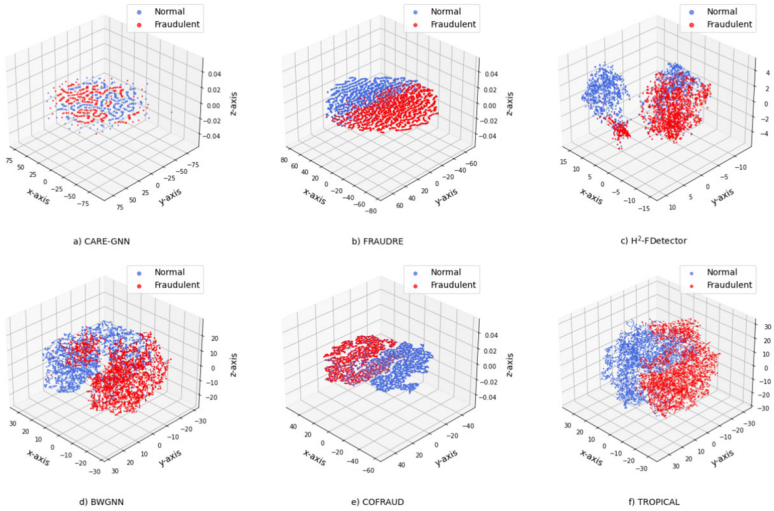


Fig. 10 Visualization of the final node embeddings for the state-of-the-art fraud detection models, including, CARE-GNN, FRAUDRE, H^2-FDetector, BWGNN, COFRAUD, and TROPICAL. The visualization results are projected into 3-dimensional space using t-SNE on the `YelpChi` dataset. Blue and Red colors represent benign and fraudulent nodes, respectively

aims to mitigate the impact of heterophilic connections during aggregation, struggles to effectively differentiate fraudulent and benign nodes, resulting in less distinct clustering. Similarly, FRAUDRE, which focuses on combining intermediate representations and imbalance-aware classification rather than producing discriminative nodes' embeddings, making it unable to effectively increase the distance between fraudulent and normal nodes in fraud graphs, eventually fails to significantly increase the distance between positive and negative samples. H^2-FDetector also exhibits limitations in producing well-separated embeddings, reducing its effectiveness in identifying fraudsters. As shown in Fig. 10, it becomes apparent that BWGNN and COFRAUD separate fraudsters from benign nodes more evidently for the `YelpChi` fraud graph. These recent models focus on integrating high-pass graph filters and cross-relation representations into the aggregation phase of classic GNN models, resulting in the creation of more discriminative embeddings and also extending the distance between fraudsters and normal nodes. Among all models, TROPICAL achieves the most effective separation by leveraging a transformer-based hypergraph learning technique, capturing complex non-pairwise relations among entities to increase the distance between fraudsters and normal entities more effectively.

7 Conclusion and future work

This study presents TROPICAL, a novel approach that leverages transformer-driven hypergraph learning to uncover fraudsters who disguise their activities within complex fraud networks. TROPICAL comprises several key components that work together to effectively identify camouflaged malicious actors in fraud graphs, including the Hypergraph Generation Module, Hyperedge Groups Aggregation Module, Positional Encoding Module, Transformer Encoder Module, and Classification Module. This research introduces a fresh perspective to

fraud detection, highlighting the importance of exploring beyond pairwise relationships to construct a more potent and robust fraud detection model. Our extensive experiments on two real-world datasets, `YelpChi` and `Amazon`, validate the superior performance of TROPICAL in fraud detection. Our future work will focus on integrating more efficient group encoding and positional encoding strategies into the TROPICAL framework. We will also explore the promising avenue of enhancing the hypergraph generation module by utilizing random walk algorithms to better capture structural and relational dependencies within fraud graphs. Furthermore, we aim to incorporate large language models (LLMs) into hypergraph learning to leverage semantic-rich representations and high-order relational reasoning, further improving fraud detection capabilities.

Despite the strong performance of TROPICAL, scalability remains a challenge when extending the model to very large graphs. The self-attention mechanism used in the transformer encoder introduces quadratic complexity with respect to the input sequence length, which increases with the number of hyperedge groups per node. Additionally, the hypergraph generation process, particularly the n-hop neighborhood expansion and pairwise KNN computation in the feature space, can become computationally intensive as the graph size grows. To address these limitations, future work could explore efficient transformer variants that reduce the cost of attention through sparse or low-rank approximations. Furthermore, scalable hypergraph construction can be achieved by incorporating approximate KNN search or hierarchical grouping strategies to limit the number of generated hyperedges. These enhancements will improve the model's applicability to large-scale fraud graphs.

Acknowledgements This work was partially supported by the Australian Research Council (ARC) Discovery Project DP230100233, and the Australia-Germany Joint Research Cooperation Scheme under grant 57702286. Venus Haghighi's work was also partially supported by a Digital Finance CRC Top-Up Scholarship and her ICDM 2024 conference attendance was partially supported by a Google Travel Grant.

Author Contributions V.H. conceptualized the study, designed the methodology, and wrote the main manuscript text. B.S. and N.S. contributed to data preprocessing, experimental setup, and implementation of the proposed model. J.W. and Y.Z. assisted with manuscript revision and provided feedback on experimental results. L.Y. and J.Y. provided theoretical insights, contributed to manuscript refinement, and supervised the study. Q.Z.S. provided overall project guidance, reviewed the manuscript, and helped refine the research direction. All authors reviewed the final manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Declarations

Competing interests This work was partially supported by the Australian Research Council (ARC) Discovery Project DP230100233, and theAustralia-Germany Joint Research Scheme under grant 57702286. Venus Haghighi's work was also partially supported by a Digital Finance CRC Top-Up Scholarship and her conference attendanceCooperation was partially supported by a Google Travel Grant.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Akoglu L, Tong H, Koutra D (2015) Graph based Anomaly Detection and Description: A Survey. Data Mining and Knowledge Discovery 29:626–688
2. Arya D, Gupta DK, Rudinac S, Worring M (2020) Hypersage: Generalizing Inductive Representation Learning on Hypergraphs. arXiv preprint arXiv:2010.04558
3. Bo D, Wang X, Shi C, Shen H (2021) Beyond Low-frequency Information in Graph Convolutional Networks. In: AAAI, pp. 3950–3957
4. Chalapathy R, Chawla S (2019) Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407
5. Cheng D, Xiang S, Shang C, Zhang Y, Yang F, Zhang L (2020) Spatio-temporal attention-based neural network for credit card fraud detection. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34, pp. 362–369
6. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems **29**
7. Dou Y, Liu Z, Sun L, Deng Y, Peng H, Yu PS (2020) Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters. In: CIKM, pp. 315–324
8. Dou Y, Ma G, Yu PS, Xie S (2020) Robust spammer detection by nash reinforcement learning. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 924–933
9. Feng Y, You H, Zhang Z, Ji R, Gao Y (2019) Hypergraph Neural Networks. In: AAAI, pp. 3558–3565
10. Gao Y, Feng Y, Ji S, Ji R (2022) HGNN+: General Hypergraph Neural Networks. IEEE Trans. on Pattern Analysis and Machine Intelligence **45**(3), 3181–3199
11. Gong Z, Wang G, Sun Y, Liu Q, Ning Y, Xiong H, Peng J (2023) Beyond homophily: Robust graph anomaly detection via neural sparsification. In: IJCAI, pp. 2104–2113
12. Haghighi V (2023) From classic gnns to hyper-gnns for detecting camouflaged malicious actors. In: WSDM, pp. 1220–1221
13. Haghighi V, Shabani N, Soltani B, Yao L, Sheng QZ, Yang J, Beheshti A (2024) Robust Graph Learning Against Camouflaged Malicious Actors. In: International Conference on Web Information Systems Engineering (WISE 2024), pp. 146–161. Springer
14. Haghighi V, Soltani B, Shabani N, Wu J, Zhang Y, Yao L, Sheng QZ, Yang J (2024) TROPICAL: Transformer-based Hypergraph Learning for Camouflaged Fraudster Detection. In: IEEE International Conference on Data Mining, (ICDM 2024). Abu Dhabi, UAE
15. Hamilton W, Ying Z, Leskovec J (2017) Inductive Representation Learning on Large Graphs. In: NeurIPS, pp. 1024–1034
16. Han J, Tao Q, Tang Y, Xia Y (2022) DH-HGCN: Dual Homogeneity Hypergraph Convolutional Network for Multiple Social Recommendations. In: SIGIR, pp. 2190–2194
17. Hooi B, Shin K, Song HA, Beutel A, Shah N, Faloutsos C (2017) Graph-based fraud detection in the face of camouflage. ACM Transactions on Knowledge Discovery from Data (TKDD) 11(4):1–26
18. Hooi B, Song HA, Beutel A, Shah N, Shin K, Faloutsos C (2016) Fraudar: Bounding Graph Fraud in the Face of Camouflage. In: SIGKDD, pp. 895–904
19. Kipf TN, Welling M (2016) Semi-supervised Classification with Graph Convolutional Networks. arXiv preprint arXiv:1609.02907
20. Liu Y, Ao X, Qin Z, Chi J, Feng J, Yang H, He Q (2021) Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection. In: WWW, pp. 3168–3177
21. Liu Z, Chen C, Yang X, Zhou J, Li X, Song L (2018) Heterogeneous graph neural networks for malicious account detection. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 2077–2085
22. Liu Z, Dou Y, Yu PS, Deng Y, Peng H (2020) Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. In: SIGIR, pp. 1569–1572
23. Luo Y (2022) SHINE: SubHypergraph Inductive Neural nEtwork. In: NeurIPS, pp. 18779–18792
24. Ma X, Liu F, Wu J, Yang J, Xue S, Sheng QZ (2025) Rethinking Unsupervised Graph Anomaly Detection With Deep Learning: Residuals and Objectives. IEEE Trans. on Knowledge and Data Engineering **37**(2), 881–895
25. Ma X, Wu J, Xue S, Yang J, Zhou C, Sheng QZ, Xiong H, Akoglu L (2023) A Comprehensive Survey on Graph Anomaly Detection with Deep Learning. IEEE Trans. on Knowledge and Data Engineering **35**(12), 12012–12038
26. Van der Maaten L, Hinton G (2008) Visualizing data using t-sne. Journal of Machine Learning Research **9**(11)

27. McAuley JJ, Leskovec J (2013) From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews. In: WWW, pp. 897–908
28. Ngai EW, Hu Y, Wong YH, Chen Y, Sun X (2011) The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision support systems 50(3):559–569
29. Ozbayoglu AM, Gudelek MU, Sezer OB (2020) Deep learning for financial applications: A survey. Applied soft computing 93:106384
30. Paszke A, Gross S, et al (2019) Pytorch: An Imperative Style, High-performance Deep Learning Library. In: NeurIPS, pp. 8024–8035
31. Peng H, Zhang R, Dou Y, Yang R, Zhang J, Yu PS (2021) Reinforced neighborhood selection guided multi-relational graph neural networks. ACM Transactions on Information Systems (TOIS) 40(4):1–46
32. Rayana S, Akoglu L (2015) Collective Opinion Spam Detection: Bridging Review Networks and Metadata. In: SIGKDD, pp. 985–994
33. Shi F, Cao Y, Shang Y, Zhou Y, Zhou C, Wu J (2022) H2-FDetector: A GNN-based Fraud Detector with Homophilic and Heterophilic Connections. In: WWW, pp. 1486–1494
34. Skorupka A (2024) Detecting anomalies in graph networks on digital markets. PloS one 19(12):e0315849
35. Tang J, Li J, Gao Z, Li J (2022) Rethinking Graph Neural Networks for Anomaly Detection. In: ICML, pp. 21076–21089
36. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is All You Need. In: NeurIPS, pp. 5998–6008
37. Velampalli S, Eberle W (2017) Novel graph based anomaly detection using background knowledge. In: FLAIRS
38. Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph Attention Networks. In: ICLR
39. Wang J, Wen R, Wu C, Huang Y, Xiong J (2019) Fdgars: Fraudster detection via graph convolutional networks in online app review system. In: Companion Proceedings of the 2019 World Wide Web Conference, pp. 310–316
40. Wang Y, Zhang J, Huang Z, Li W, Feng S, Ma Z, Sun Y, Yu D, Dong F, Jin J, et al (2023) Label information enhanced fraud detection against low homophily in graphs. In: Proceedings of the ACM Web Conference 2023, pp. 406–416
41. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2020) A Comprehensive Survey on Graph Neural Networks. IEEE Trans. on Neural Networks and Learning Systems **32**(1), 4–24
42. Xu K, Hu W, Leskovec J, Jegelka S (2019) How Powerful are Graph Neural Networks? In: ICLR
43. Yadati N, Nimishakavi M, Yadav P, Nitin V, Louis A, Talukdar P (2019) HyperGCN: A New Method for Training Graph Convolutional Networks on Hypergraphs. In: NeurIPS, pp. 1509–1520
44. Yang Z, Zhang G, Wu J, Yang J, Sheng QZ, Xue S, Zhou C, Aggarwal C, Peng H, Hu W, Hancock ER, Liò P (2025) State of the Art and Potentialities of Graph-level Learning. ACM Computing Surveys **57**(2), 28:1–28:40
45. Yun S, Jeong M, Kim R, Kang J, Kim HJ (2019) Graph Transformer Networks. In: NeurIPS, pp. 11960–11970
46. Zang Y, Hu R, Wang Z, Xu D, Wu J, Li D, Wu J, Ren L (2023) Don't Ignore Alienation and Marginalization: Correlating Fraud Detection. In: IJCAI, pp. 4959–4966
47. Zhang G, Wu J, Yang J, Beheshti A, Xue S, Zhou C, Sheng QZ (2021) Fraudre: Fraud Detection Dual-resistant to Graph Inconsistency and Imbalance. In: ICDM, pp. 867–876
48. Zheng X, Liu Y, Pan S, Zhang M, Jin D, Yu PS (2022) Graph Neural Networks for Graphs with Heterophily: A Survey. arXiv preprint arXiv:2202.07082
49. Zhu J, Yan Y, Zhao L, Heimann M, Akoglu L, Koutra D (2020) Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In: NeurIPS, pp. 7793–7804

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Venus Haghighi is a PhD candidate in the School of Computing at Macquarie University, Australia. Her research focuses on developing advanced graph-based fraud detection models to identify fraudulent and malicious behaviors in online networks, with a particular emphasis on graph neural networks, hypergraph learning, and graph transformer architectures. She has published in leading international conferences, including ICDM, IJCAI, and CIKM. Her work aims to address challenges in modeling complex, multi-relational data and uncovering camouflaged fraud patterns through robust graph learning. She is also investigating the integration of large language models (LLMs) to enhance contextual reasoning in fraud detection systems. Her research interests include graph data mining, graph neural networks, hypergraph learning, graph transformer networks, and large language models for fraud detection.



Behnaz Soltani is a Ph.D. candidate in the School of Computing at Macquarie University, Sydney, Australia. She received her Bachelor's and Master's degrees in Computer Engineering from the University of Tabriz and Sharif University of Technology, Iran, respectively. Her research interests include data mining, machine learning, and federated learning. She has published in top-tier venues such as IJCAI and CIKM.



Nasrin Shabani received her Master of Research degree in Computer Science with First Class Honours from Macquarie University, Sydney, NSW, Australia. She is currently completing her Ph.D. in Computer Science at the same institution. Her research focuses on graph mining, graph summarization, and deep learning, with an emphasis on developing algorithms to extract meaningful patterns from complex graph data structures.



Jia Wu is the Research Director for the Centre for Applied Artificial Intelligence and the Director of Higher Degree Research in the School of Computing at Macquarie University, Sydney, Australia. Dr Wu received his Ph.D. degree in computer science from the University of Technology Sydney, Australia. His current research interests include data mining and machine learning. Since 2009, he has published 200+ refereed journal and conference papers, including TPAMI, TKDE, TCYB, TKDD, TNNLS, TMM, KDD, WSDM, ICDM, CIKM, WWW, and NeurIPS. He is the Associate Editor of IEEE Transactions on Neural Networks and Learning Systems and ACM Transactions on Knowledge Discovery from Data. He is a Senior Member of the IEEE.



Yang Zhang is a Clinical Assistant Professor in the Department of Data Science at the University of North Texas. His research interests lie at the intersection of artificial intelligence and the artificial intelligence of things (AIoT), with a particular focus on natural language processing, recommender systems, and federated learning. Dr. Zhang's recent work explores the application of Large Language Models (LLMs) for advanced text mining and the development of intelligent IoT systems. His research has been published in several top-tier conferences, including ACM CIKM, IEEE ICDM, ACM UbiComp, and IEEE INFOCOM.



Lina Yao is a Senior Principal Research Scientist and Science Lead (Research Manager) @ CSIRO's Data61, Conjoint Professor @ UNSW, Honorary Professor @ Macquarie University and Adjunct Professor @ University of Technology Sydney. Her research interests include few-shot learning, zero-shot learning, deep reinforcement learning, self-supervised learning, deep generative modeling and their applications in a broad range of applications in recommender systems, computer vision, brain-computer interface, intelligent transportation system, and internet of things.



Jian Yang is a Co-Director of Data Horizons research Centre and Deputy Director of Applied Artificial Intelligence Research Centre at Macquarie University. She received her PhD in Data Integration from The Australian National University in 1995. Before she joined Macquarie University, she worked as an associate professor at Tilburg University, the Netherlands, a senior research scientist at CSIRO, Australia, and as a lecturer at The Australian Defence Force Academy, University of New South Wales. Prof Yang's current interests are graph mining; trust, fake news detection, recommendation in social media; fraud detection; data fusion, data-driven process analytics; service-oriented computing; smart city; aging and aged care.



Quan Z. Sheng is a distinguished professor and Head of School of Computing at Macquarie University, Australia. His research interests include service computing, web technologies, machine learning, and internet of things. Michael holds a PhD degree in computer science from the University of New South Wales. Distinguished Prof Sheng is the recipient of ARC (Australian Research Council) Future Fellowship, Chris Wallace Award for Outstanding Research Contribution, and Microsoft Research Fellowship. He is ranked by Microsoft Academic as one of the most impactful authors in services computing and by ScholarGPS as one of the highly ranked scholars.

Authors and Affiliations

Venus Haghighi[1] · Behnaz Soltani[1] · Nasrin Shabani[1] · Jia Wu[1] · Yang Zhang[2] · Lina Yao[3,4] · Jian Yang[1] · Quan Z. Sheng[1]

✉ Venus Haghighi
venus.haghighi@hdr.mq.edu.au

Behnaz Soltani
behnaz.soltani@hdr.mq.edu.au

Nasrin Shabani
nasrin.shabani@hdr.mq.edu.au

Jia Wu
jia.wu@mq.edu.au

Yang Zhang
yang.zhang@unt.edu

Lina Yao
lina.yao@data61.csiro.au

Jian Yang
jian.yang@mq.edu.au

Quan Z. Sheng
michael.sheng@mq.edu.au

1 School of Computing, Macquarie University, Sydney 2109, NSW, Australia

2 College of Information, University of North Texas, Denton, USA

3 School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

4 CSIRO's Data61, Eveleigh, Australia