TROPICAL: Transformer-based Hypergraph Learning for Camouflaged Fraudster Detection

Venus Haghighi*, Behnaz Soltani*, Nasrin Shabani*, Jia Wu*, Yang Zhang*,
Lina Yao†‡, Quan Z. Sheng*, Jian Yang*
*School of Computing, Macquarie University, Sydney, NSW 2109, Australia
†School of Computer Science and Engineering, University of New South Wales, NSW, Australia
‡CSIRO's Data61, Australia
Email: {venus.haghighi, behnaz.soltani, nasrin.shabani}@hdr.mq.edu.au,, lina.yao@data61.csiro.au,
{jia.wu, yang.zhang}@mq.edu.au, lina.yao@unsw.edu.au, {michael.sheng, jian.yang}@mq.edu.au

Abstract—**Graph-based fraud detection has attracted increasing attention in recent years, reflecting its growing potential in mitigating sophisticated fraudulent activities. The main objective of graph-based fraud detection is to discern between fraudsters and normal entities within graphs. As fraudsters adopt increasingly sophisticated camouflage tactics, combating them has become an urgent task. Despite the complex interactions within real-world networks involving high-order relations, existing graph-based fraud detection methods often neglect non-pairwise relationships among entities in graphs. Thus, we emphasize the significance of investigating beyond pairwise relationships for building an effective fraud detection model. In this paper, we propose constructing a hypergraph from the original input graph to encapsulate comprehensive high-order relations and present TROPICAL, a novel TRansfOrmer-based hyPergraph LearnIng for detecting CAmouflaged maLicious actors in online social networks. TROPICAL learns representations by processing different hyperedge groups and incorporates positional encodings into the aggregated information to enhance their distinctiveness. Subsequently, the model feeds the learned aggregated sequential information into the transformer encoder, achieving rich representations for effective camouflaged fraudster detection. The superiority of TROPICAL is demonstrated through experiments conducted on two real-world datasets, compared against the state-of-the-art fraud detection models. The source codes and datasets of our work are available at https://github.com/VenusHaghighi/TROPICAL.**

Index Terms—**Hypergraph Learning, Camouflage, Fraudster Detection**

I. Introduction

The proliferation of the Internet and online services has led to a substantial increase in fraudulent activities. Fraudsters have become more crafty, often causing destructive impacts and making the identification process more challenging [1]. Recently, graph-based fraud detection approaches [2], specifically graph neural network (GNN)-based ones [3], [4], have attracted enormous attention in academia and industry due to their ability to capture rich behavioral information among entities within graph-structured data. However, existing graph-based approaches overlook the high-order relationships among entities, focusing solely on pairwise connections, which is insufficient for fully inferring the correlations among entities.

There are two major challenges in current GNN-based fraud detection models [5], [6]. Firstly, they are vulnerable to the camouflage behavior of fraudsters, wherein the fraudsters deliberately establish many connections to normal entities [7]. For instance, spammers often use benign accounts to post their spam reviews or add links to well-liked items in a product review system. This behavior alleviates the fraudsters' suspiciousness and eventually bypasses detection systems. Particularly, in the aggregation process of GNNs, low-pass filtering is employed to retain the commonality of node features in the same neighborhoods. Such aggregation strategy leads to the assimilation of neighboring nodes, making their representations more similar even if they belong to different class labels, i.e., fraudulent or benign. Consequently, distinguishing malicious nodes becomes challenging, which significantly hinders the performance of fraud detection [8]. Secondly, in addition to 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 and accurate node representations.

In response to the challenges posed by camouflaged fraudsters within fraud graphs, several enhanced GNN-based fraud detection models have been proposed [9], [10], [11], [12], [13], [14], 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 [6]. 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 distribution, where less concentration is on low frequencies and more on high frequencies [11]. 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 [15].

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. 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 two-fold 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 two-fold 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 rest of the paper is organized as follows. Section II briefly introduces the relevant work in the literature. Section III describes our problem and its related definitions, and Section IV discusses the motivation of our research. The technical details of our new proposed framework and the experimental results are presented in Section V and Section VI, respectively. Finally, Section VII offers some concluding remarks.

II. Related Work

In this section, we give an overview of the relevant research on hypergraph neural networks and further discuss the work on graph-based fraud detection.

A. Hypergraph Neural Networks

The success of various graph neural network (GNN) models [3] across different tasks, such as node classification, has demonstrated their capability in revealing structural relationships within graph-structured data [4]. However, GNN models primarily focus on pairwise connections for generating node representations, limiting their ability to effectively represent complex high-order relations that extend beyond pairwise associations. To bridge this gap, hypergraph learning has recently emerged as a promising technique to capture and model intricate high-order relations among entities in graph data, finding a vast range of applications in multiple domains, such as genetic medicine [16] and social recommendation [17]. Hypergraph Neural Network (HGNN) [18] is the first work that extends GNNs to hypergraphs by designing a hyperedge convolution operation. HGNN^{+} [19] 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 [20] introduces a novel way of training a GCN on hypergraphs based on tools from spectral theory. HyperSAGE [21] presents an inductive learning framework to propagate information through hypergraphs efficiently. Hypergraph neural networks have found a vast range of applications in multiple domains, such as genetic medicine [16] and social recommendation [17]. 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 for identifying camouflaged malicious actors or fraudsters within online social networks.

B. Graph-based Fraud Detection

Considering the camouflage behavior of fraudsters in networks, there is a growing interest in designing GNN-based models that exhibit robustness against these camouflaged fraudulent activities. Two general approaches exist to enhance GNN-based fraud detection models against camouflaged fraudsters. The first approach modifies the neighborhood structure within the aggregation process of GNN. GraphConsis [12] is the first work that explores the graph inconsistencies problem derived by camouflaged fraudsters. To boost the aggregation process in GNN models, GraphConsis filters dissimilar neighbors for a given central node based on a pre-defined threshold. CARE-GNN [10] is another study that adopts reinforcement learning to adjust a threshold and refine the graph structure before the aggregation process. PC-GNN [22] designs a label-balanced sampling technique for node and edge selection in sub-graph training. Following this, it employs a sampler to over-sampling neighbors for the minority class while under-sampling neighbors for the majority class.

The second approach focuses on improving the aggregation process of GNN models by devising an adaptive aggregation strategy. FRAUDRE [13] proposes a GNN-based fraud detection model that is dual-resistant to graph inconsistencies and the imbalanced nature imposed by fraudsters. H^{2}-FDetector [23] introduces a new information aggregation strategy to make the homophilic (resp., heterophilic) connections propagate similar (resp., different) information. BWGNN [11] introduces a novel graph neural network architecture based on the band-pass Hammond's graph wavelet theory, enhancing the distinction of anomalies within graphs. COFRAUD [9] is a correlation-aware fraud detection model, that explores the correlation among multi-relation interactions within fraud graphs and introduces a fraud detection model based on two statistical metrics, i.e., alienation and marginalization.

Despite the promising success attained by the previous studies, they largely overlook *non-pairwise* relationships among nodes within graphs. Hence, there is an urgent need to learn the high-order relations among nodes, aiming to generate more comprehensive representations. Our study stands as a pioneering effort, exploring and leveraging rich non-pairwise structural and semantic information among nodes within graphs to construct a more effective fraud detection model.

III. Preliminaries

A. 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.

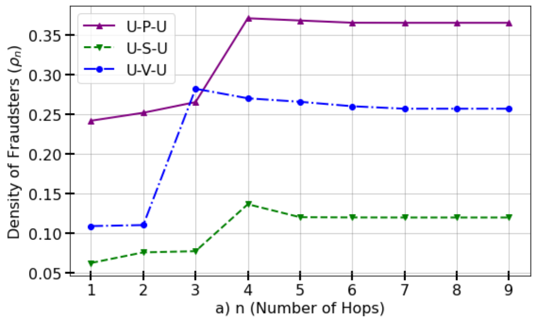
Hypergraph. In contrast to a simple graph, an edge within a hypergraph connects multiple nodes rather than just two nodes. A hypergraph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V}$ and X are the node set and the corresponding node attributes, respectively. The edge 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 $h(v, e) = 0$.

B. 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:

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

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 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.



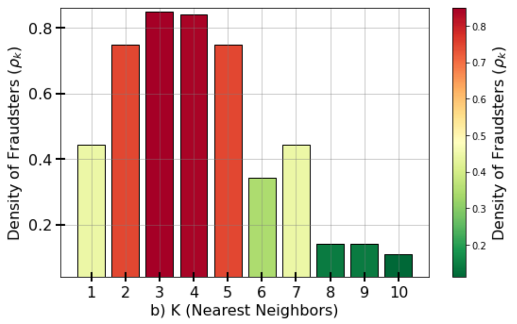


Fig. 1: a) The density of fraudsters vs. the number of hops for different relations of the `Amazon` dataset. b) The density of fraudsters for various K values of nearest neighbors for the `Amazon` dataset. The plots illustrate that higher-order connections among fraudsters can be generated based on certain values of n and K, which has the potential to reveal the behavioral patterns of fraudulent activities effectively.

IV. 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` [24], which includes three types of relations: 1) U-P-U, 2) U-S-U, and 3) U-V-U (see Section VI-A). First, we utilize the following formula to measure the density of fraudsters from the structural spaces of relation-specific subgraphs.

$$\rho_n(v_i, n) = \sum_{v_i \in V \& y_i = 1} \frac{|\{v_j | v_j \in N_{hop_n}(v_i), y_j = 1\}|}{|N_{hop_n}(v_i)|} \tag{2}$$

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. 1 (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 as follows:

$$\rho_k(v_i, k) = \sum_{v_i \in V \& y_i = 1} \frac{|\{v_j | v_j \in N_{knn_k}(v_i), y_j = 1\}|}{|N_{knn_k}(v_i)|}, \quad (3)$$

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. 1 (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.

V. Methodology

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

A. Framework Overview

As depicted in Fig. 2, TROPICAL consists of several main components. Specifically, the hypergraph generation module constructs different hyperedge groups from the original multi-relational input graph based on n-hop neighborhood and k-nearest neighbors information. The hyperedge groups aggregation module aggregates different constructed hyperedge groups' information to generate rich sequential input features. The positional encodings module incorporates positional encodings into the sequential input features, making it more distinct, and the transformer encoder module learns the representations for our model. Finally, the classification module distinguishes normal and fraudulent nodes based on the learned model representations.

B. Hypergraph Generation Module

Hypergraph construction is the initial step in hypergraph learning. To capture rich high-order relations among entities, we employ a two-fold strategy to construct different groups of hyperedges from the original multi-relational input graph. The first focuses on leveraging data correlation with the graph structure and the second involves utilizing the feature space of nodes to generate hyperedge groups.

Hyperedge group using n-hop neighbors ($\mathcal{E}_{hop_n}$). The n-hop neighborhood information is utilized for hyperedge group generation from the data with graph structure. $\mathcal{E}_{hop_n}$ finds the related hyper vertices for a central node v through the n-hop reachable positions in the graph structure. Hence, the n-hop neighborhood of a vertex v_i is defined as: $N_{hop_n}(v_i) = \{v_j | A^n_{ij} \neq 0, v_j \in N(v_i)\}$, 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 hyperedge group $\mathcal{E}_{hop_n}$ can be formulated as:

$$\mathcal{E}_{hop_n} = \{N_{hop_n}(v_i) | v_i \in V\}. \quad (4)$$

$\mathcal{E}_{hop_n}$ extends the search radius to the n-hop neighborhood within the graph structure, leading to groups of vertices for the hyperedges to provide more comprehensive correlation information. We also consider $\mathcal{E}_{hop_1}$ to preserve the ego network and low-order correlation of a given central node in the original graph structure as basic needed information.

Hyperedge group using features ($\mathcal{E}_{knn_k}$). Considering the feature for each vertex, the second group of hyperedges, i.e., $\mathcal{E}_{knn_k}$, is generated by finding K nearest neighbors of that vertex within the feature space. Given a central node v_i, its K-nearest neighbors in the feature space are connected by a hyperedge. $\mathcal{E}_{knn_k}$ is formulated as follows:

$$\mathcal{E}_{knn_k} = \{N_{knn_k}(v_i) | v_i \in V\}. \quad (5)$$

This group of hyperedges aims at exploring the correlation among entities within their features.

Combination of hyperedge groups. Different groups of hyperedges are generated utilizing the above strategies. We need to further combine them to generate the ultimate hypergraph. Using the incidence matrices corresponding for each constructed hyperedge denoted as $\mathbf{H}_{hop_1}$, $\mathbf{H}_{hop_n}$, and $\mathbf{H}_{knn_k}$, we can concatenate them directly to form the incidence matrix for the final hypergraph $\mathcal{G}$ as follows:

$$\mathbf{H}^{\mathcal{G}} = \mathbf{H}_{hop_1} || \mathbf{H}_{hop_n} || \mathbf{H}_{knn_k}. \quad (6)$$

C. Hyperedge Groups Aggregation Module

The neighborhood aggregation function of traditional GNN models such as GCN [25], GraphSAGE [26], and GAT [27] primarily collects information from local neighbors, limiting their ability to capture and generalize distant information across graphs. Such an aggregation strategy fails to identify fraudsters who perform camouflage in fraud graphs. Motivated to solve the above limitation, we devise a hyperedge groups aggregation strategy to access distant neighbors and capture rich high-order information within fraud graphs.

Hyperedge Group Aggregation for $\mathcal{E}_{hop_n}$. We devise a Discriminator Aggregation strategy for constructed $\mathcal{E}_{hop_n}$ to highlight the distinct information of nodes from different classes. Its primary objective is to generate discriminative embeddings by increasing the distance between 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

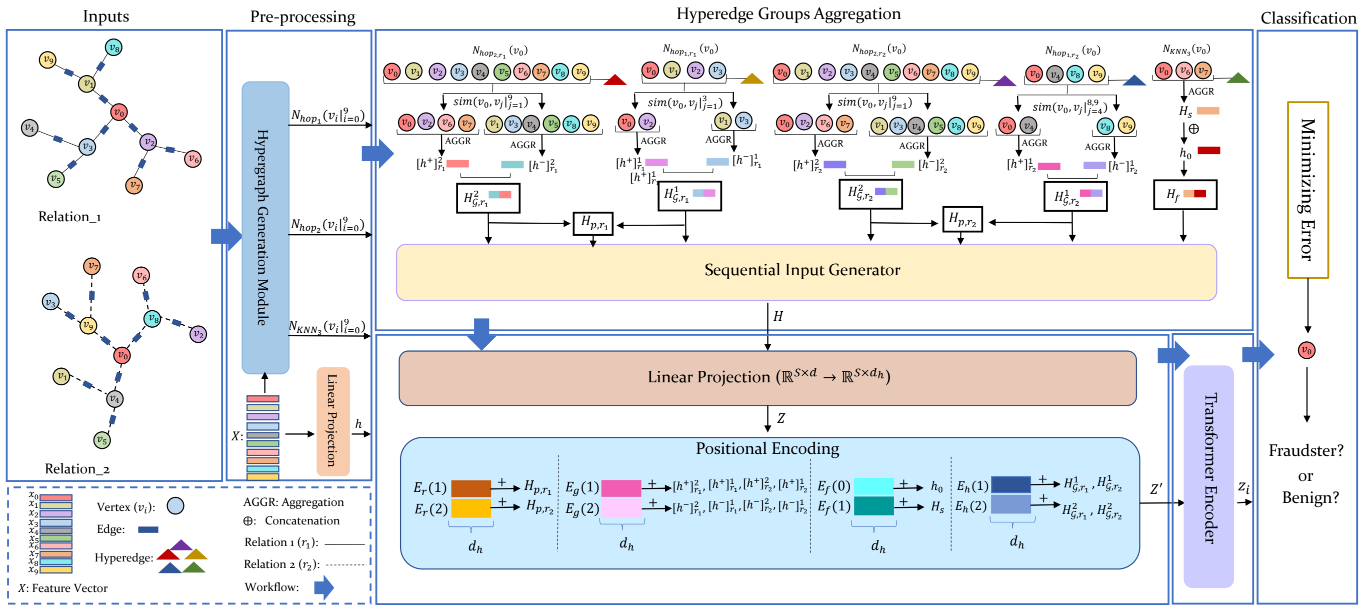


Fig. 2: The overview of the TROPICAL framework. TROPICAL takes a multi-relational graph as input. Firstly, the hypergraph generation module constructs different groups of hyperedges for each relation-specific subgraph. Secondly, the hyperedge groups aggregation module aggregates information from different 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.

as a pivotal metric for measuring the similarities between feature vectors of nodes in n-dimensional space as follows:

$$sim(v_i, v_j) = \{\frac{x_i.x_j}{\|x_i\|.\|x_j\|}, |v_j \in N_{hop_n}(v_i)\}, \tag{7}$$

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.

$$\mathbf{H_g^n} = [\mathbf{h^+}, \mathbf{h^-}]^\mathbf{n}, \tag{8}$$

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:

$$[\mathbf{h^+}]^\mathbf{n} = F_{agg} = \frac{1}{|V^+|}\sum_{u\in V^+, i=1}^{2} x_u,$$
$$[\mathbf{h^-}]^\mathbf{n} = F_{agg} = \frac{1}{|V^-|}\sum_{u\in V^-, i=1}^{2} x_u, \tag{9}$$

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:

$$\mathbf{H_p} = [\mathbf{h_g^1} \oplus \mathbf{h_g^n}], \tag{10}$$

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 $\mathcal{E}_{knn_k}$. Given a target node v_i, its *Feature Encoding* for $\mathcal{E}_{knn_k}$ can be constructed as follows:

$$\mathbf{H_s} = \sum_{u\in N_{knn_k}(v_i)} x_u. \tag{11}$$

Moreover, to apply self-attention on the target node, we combine the raw feature vector of the target node v_i, i.e., x_i, with $\mathbf{H_s}$ to generate final *Feature Encoding* as follows:

$$h_i = \sigma(x_i W_i), \tag{12}$$

$$\mathbf{H_f} = [h_i] \oplus \mathbf{H_s}, \tag{13}$$

where h_i is the linear feature projection of target node v_i, and σ is the linear activation function.

Sequential Input Feature Generator. We combine all the encodings to generate the final input feature sequence as follows:

$$\mathbf{H} = \mathbf{H_p} \oplus \mathbf{H_f}. \tag{14}$$

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, 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.

D. Positional Encoding Module

To make the structural, relational, and feature information more distinct in the generated sequential input data, $\mathbf{H}$, four positional encodings are added, which include: i) hop positional encoding ($E_h(.)$), ii) relation positional encoding ($E_r(.)$), iii) group positional encoding ($E_g(.)$), and iv) feature positional encoding ($E_f(.)$). We first perform the linear transformation of sequential input data as follows:

$$\mathbf{Z} = \sigma(\varphi(\mathbf{H})), \tag{15}$$

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.

Then, we add hop positional encoding to make the encoder aware of the multi-hop structural information. Furthermore, incorporating relation positional encoding empowers the model to differentiate between various relations within the fraud graph. In addition, employing group positional encoding becomes imperative to differentiate between the aggregation outputs of different groups of nodes. Moreover, we adopt feature positional encoding to make k-nearest neighbor features and central node features more distinguished.

The details of the positional encoding module can be found in Fig. 2. 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:

$$\begin{aligned}\mathbf{Z}' = [(\mathbf{H}_{\mathbf{p},\mathbf{r_1}} + E_r(1))||...||(\mathbf{H}_{\mathbf{p},\mathbf{r_n}} + E_r(n))] \\ \oplus [\mathbf{H_s} + E_f(1)] \oplus [h_i + E_f(0)]\end{aligned} \tag{16}$$

Hop positional encoding for relation r_i can be incorporated to Eq. 16 as follows:

$$\mathbf{H}_{\mathbf{p},\mathbf{r_i}} = [(\mathbf{h}^{\mathbf{1}}_{\mathbf{g},\mathbf{r_i}} + E_h(1)) \oplus (\mathbf{h}^{\mathbf{n}}_{\mathbf{g},\mathbf{r_i}} + E_h(2))] + E_r(i). \tag{17}$$

Moreover, group positional encoding can be added to Eq. 17 as follows:

$$\begin{aligned}\mathbf{h}_{\mathbf{g},\mathbf{r_i}} = [([\mathbf{h}^{+} + E_g(1), \mathbf{h}^{-} + E_g(2)]^{1}_{r_i} + E_h(1)) \\ \oplus ([\mathbf{h}^{+} + E_g(1), \mathbf{h}^{-} + E_g(2)]^{n}_{r_i} + E_h(2))] + E_r(i),\end{aligned} \tag{18}$$

E. Transformer Encoder

We employ a transformer-based encoder to learn representations from the final sequential input data, $\mathbf{Z}'$, generating in Section V-D. Inspired by [28] and [29], in a transformer encoder, the multi-head attention modules are utilized to perform deep interactions among vectors. The updating process of z_i for position i in input data, $\mathbf{Z}'$, for layer $l+1$ is as follows:

$$z_i^{l+1} = Function(head_1, ..., head_h)W^l, \tag{19}$$

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 is defined as:

$$\begin{aligned} head_h &= Attention(Q^{m,l}z_i^l, K^{m,l}z_j^l, V^{m,l}z_j^l), \\ &= \sum_{j\in S} w_{ij}(V^{m,l}z_j^l), \\ w_{i,j} &= softmax\left(\frac{Q^{m,l}z_i^l . K^{m,l}z_j^l}{\sqrt{d_h}}\right), \end{aligned} \tag{20}$$

where $Q^{m,l}$, $K^{m,l}$, $V^{m,l}$ are the learnable weights of the h^{th} attention head, and $w_{i,j}$ is the attention scores using dot products followed by softmax.

F. Classification Module

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

$$L_{CE} = -\sum_{v_i\in V}[\gamma y_i log(p_i) + (1-y_i)log(1-p_i)], \tag{21}$$

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:

$$p_i = softmax(z_i), \tag{22}$$

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

G. 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 $O(S^2d + Sd^2)$, where S is the length of the sequential input feature. The overall complexity of TROPICAL is $O(|V|c^n d + |v|kd + eh(S^2d + Sd^2))$, where h is the number of attention layers and e is the number of training epochs.

VI. Experiments

A. Experimental Setup

Datasets. Our experiments involve two real-world datasets: `Amazon` [24], derived from musical instrument comments on *Amazon.com*, and `YelpChi` [30], consisting of spam reviews related to restaurants and hotels. In `Amazon`, three relationships are explored: U-P-U (users reviewing at least one common product), U-S-U (users sharing at least one same star rating within a week), and U-V-U (users exhibiting top 5% mutual review similarities). `YelpChi` also includes three relationships: R-U-R (reviews posted by the same user), R-S-R (reviews under the same product with matching star ratings),

TABLE I: The statistical details of the datasets.

Dataset	#Nodes (Fraud%)	#Features	Class	#Class	Relation	#Edges	$H^G_{edge}(G_r)$	$H^L_{node}(G_r)$
YelpChi	45,954 (14.53%)	32	Positive	6,677	R-U-R	98,630	0.10	0.09
					R-T-R	1,147,232	0.95	0.82
			Negative	39,277	R-S-R	6,805,486	0.95	0.81
					All (homo)	8,051,348	0.93	0.81
Amazon	11,944 (6.87%)	25	Positive	821	U-P-U	351,216	0.81	0.83
			Negative	7,818	U-S-U	7,132,958	0.96	0.94
			Unlabeled	3,305	U-V-U	2,073,474	0.97	0.95
					All (homo)	9,557,648	0.95	0.92

and R-T-R (reviews under the same product posted in the same month). All (homo) includes all types of relations. We analyze the camouflage behavior of fraudsters by adopting two metrics for measuring the local and global heterophily rates of a given multi-relational graph G. Global heterophily rate, $H^G_{edge}(G_r)$, calculates the proportion of edges that link two nodes from different class labels, i.e., fraudulent and benign nodes, under relation r. $H^G_{edge}(G_r)$ is defined as follows:

$$H^G_{edge}(G_r) = \frac{|\{e_{ij}|e_{ij} \in E_r, y_i \neq y_j\}|}{|E_r|} \tag{23}$$

We calculate the global metric of $H^G_{edge}(G_r)$ for both `YelpChi` and `Amazon` datasets to prove the existence of camouflage behavior among fraudsters. A high-score of $H^G_{edge}(G_r)$ $(\rightarrow 1)$ implies that fraudsters largely succeed to camouflage themselves. We also utilize another metric, i.e., local heterophily value for a given fraudulent node v_i of graph G under relation r, which is defined as follows:

$$H^L_{node}(G_r) = \frac{1}{|V|} \sum_{v_i \in V \& y_i = 1} \frac{|\{v_j|v_j \in N_{v_i}, y_i \neq y_j\}|}{d_{v_i}} \tag{24}$$

$H^L_{node}(G_r)$ calculates the average number of heterophilic connections in local neighborhoods of fraudulent nodes in graph G. Table I collects the statistical information of these two datasets. According to $H^L_{node}(G_r)$, we observe that fraudulent nodes create over 80% of heterophilic connections in the datasets, except the R-U-R relation of `YelpChi`. As a result, camouflage is a prevalent action among fraudsters in fraud graphs.

Baselines. We categorize the state-of-the-art models for overall comparison. First, we select traditional GNN models including GCN [25], GAT [27], and GraphSAGE [26]. FAGCN [15] is an enhanced version of traditional GNN models designed for low homophily settings. Second, hypergraph neural network models such as HGNN [18] and HyperGCN [20] are utilized to perform binary node classification on hypergraphs. The third category focuses on the state-of-the-art GNN-based fraud detection models for comparison. These fraud detection models include GraphConsis [12], CARE-GNN [10], PC-GNN [22], FRAUDRE [13], H^2-FDetector [23], BWGNN [11], and COFRAUD [9]. It is worth mentioning that we follow these papers to choose hyperparameters for the regeneration of results.

Evaluation Metrics. We adopt AUC and Macro-F1 as evaluation metrics to evaluate all models. AUC is the probability that the model ranks a randomly selected positive sample (fraudulent node) higher than a randomly selected negative sample (benign node). Macro-F1 is the unweighted mean of the F1-score that indicates the performance of the model particularly when imbalanced data are present.

Implementation Details. We implement our model using Pytorch [31]. We also implement the baseline models and regenerate their results in our environment based on the codes provided by the authors. All models are run on Python 3.9.13, 1 NVIDIA Tesla P100 GPU, and 480GB RAM. For our model, we select Adam as the optimizer. The learning rate and weight decay are set to 0.001 and 0.0001. The number of transformer encoder layers is set to 2 and 3 for `Amazon` and `Yelpchi`, respectively. Moreover, the number of attention heads is set to 4, and the dropout rate is set to 0.1.

B. Performance Comparison

We compare the performance of TROPICAL with the baseline methods. The results in Table IV show that TROPICAL outperforms all other GNN models. It is worth noting that GCN, GAT, GraphSAGE, and FAGCN are designed for single relation input graphs, while others exploit and run for multi-relational fraud graphs. We observe that directly applying traditional GNN models for fraud detection fails to achieve promising results. Traditional GNN models act as low-pass filters during the aggregation process, which works poorly for low homophily problems such as fraud detection.

While FAGCN demonstrates better performance compared to traditional GNN models by incorporating both high-frequency and low-frequency channels in the aggregation procedure, there remains room for further improvement. 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. FRAUDRE, H^2-FDetector, COFRAUD, and BWGNN improve the aggregation operation by incorporating high-frequency information and stacking multiple modules, yet their effectiveness is limited to pair-wise relationships. Our

TABLE II: Performance comparison under different percentages of training data.

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

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.

C. Parameter Analysis

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. Moreover, we set different numbers of transformer encoder layers for TROPICAL. The results shown in Fig. 3 indicate that TROPICAL maintains acceptable stability with regard to different hyperparameters for both `YelpChi` and `Amazon` datasets. We also observe that deploying two layers and three layers of transformer encoder for `Amazon` and `YelpChi`, respectively, yields superior results.

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. Fig. 4 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`.

D. Ablation Study

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 shown in Fig. 5, TROPICAL performs better compared to other variants, indicating that each component plays an important role in the model. The training ratio is set as 0.4. In both datasets, w/o TE exhibits the worst results, which indicates the importance of the transformer encoder. We observe a significant drop by 6.43% and 5.87% in terms of AUC for the w/o TE variant in both `YelpChi` and `Amazon`, respectively. Moreover, the performance drops by 5.21% and 4.87% for w/o TE in terms of Macro-F1 for `YelpChi` and `Amazon`, respectively. TROPICAL incorporates multi-head attention in the transformer encoder to adaptively reweigh group vectors, enhancing the capture of semantic information.

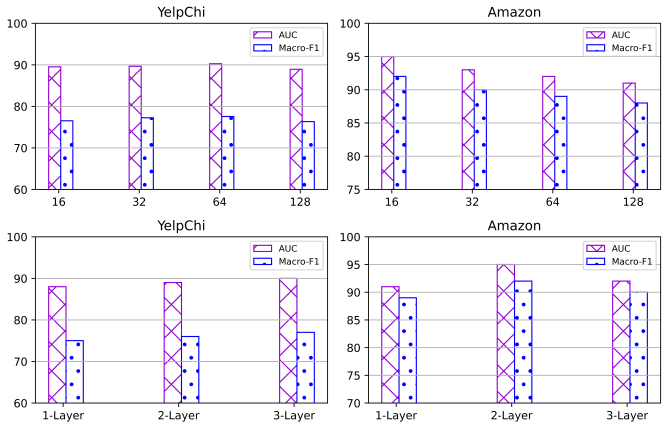


Fig. 3: Parameter Analysis. The performance of TROPICAL on different embedding sizes (Row 1), and different numbers of transformer encoder layers (Row 2).

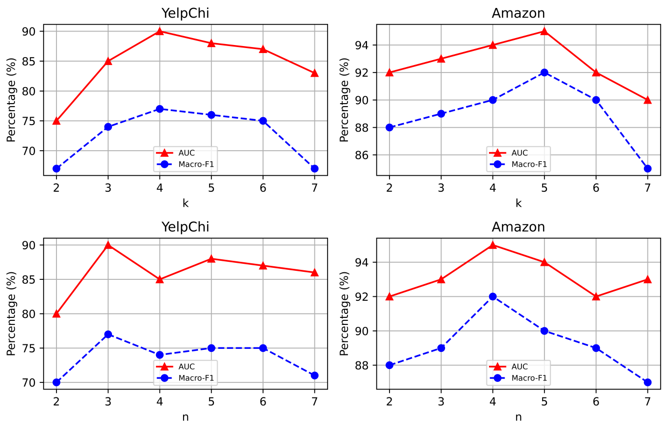


Fig. 4: Parameter Analysis. The performance of TROPICAL for different values of parameters k (Row 1) and n (Row 2).

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 III 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.

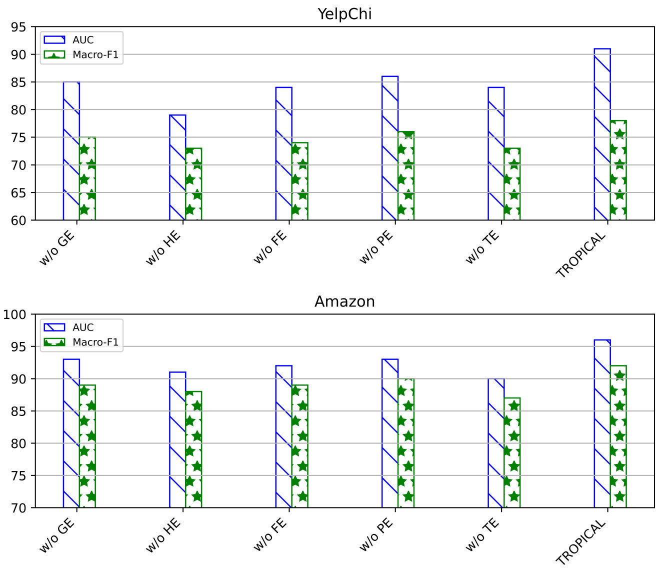


Fig. 5: The ablation analysis of the TROPICAL approach.

TABLE III: The ablation analysis to verify the effectiveness of different 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

E. Time Efficiency

This experiment focuses on time efficiency and we compare TROPICAL with four state-of-the-art fraud detection models, CARE-GNN [10], FRAUDRE [13], BWGNN [11], and COFRAUD [9]. 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 IV, 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 Section VI-B).

TABLE IV: 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

VII. Conclusion and Future Work

In this study, we have introduced TROPICAL, an innovative hypergraph learning model based on transformers that is designed to identify camouflaged fraudsters within complex fraud graphs. TROPICAL comprises several key components that form a robust framework for effectively capturing camouflaged malicious actors in fraud graphs. This research introduces a fresh perspective to fraud detection, highlighting the importance of exploring beyond pairwise relationships to construct a more potent fraud detection model. Our extensive experiments on two real-world datasets, `YelpChi` and `Amazon`, demonstrate the superior performance of TROPICAL. 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 through the utilization of random walk algorithms.

Acknowledgments

This work was partially supported by the Australian Research Council (ARC) Discovery Project DP230100233. Venus Haghighi's work was also partially supported by a Digital Finance CRC Top-Up Scholarship and her conference attendance was partially supported by a Google Travel Grant.

References

[1] S. Velampalli and W. Eberle, "Novel graph based anomaly detection using background knowledge," in *FLAIRS*, 2017.

[2] L. Akoglu, H. Tong, and D. Koutra, "Graph based Anomaly Detection and Description: A Survey," *Data Mining and Knowledge Discovery*, vol. 29, pp. 626–688, 2015.

[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[4] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A Comprehensive Survey on Graph Anomaly Detection with Deep Learning," *IEEE Trans. on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 012–12 038, 2023.

[5] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" in *ICLR*, 2019.

[6] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs," in *NeurIPS*, 2020, pp. 7793–7804.

[7] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding Graph Fraud in the Face of Camouflage," in *SIGKDD*, 2016, pp. 895–904.

[8] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, "Graph Neural Networks for Graphs with Heterophily: A Survey," *arXiv preprint arXiv:2202.07082*, 2022.

[9] Y. Zang, R. Hu, Z. Wang, D. Xu, J. Wu, D. Li, J. Wu, and L. Ren, "Don't Ignore Alienation and Marginalization: Correlating Fraud Detection," in *IJCAI*, 2023, pp. 4959–4966.

[10] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters," in *CIKM*, 2020, pp. 315–324.

[11] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking Graph Neural Networks for Anomaly Detection," in *ICML*, 2022, pp. 21 076–21 089.

[12] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng, "Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection," in *SIGIR*, 2020, pp. 1569–1572.

[13] G. Zhang, J. Wu, J. Yang, A. Beheshti, S. Xue, C. Zhou, and Q. Z. Sheng, "Fraudre: Fraud Detection Dual-resistant to Graph Inconsistency and Imbalance," in *ICDM*, 2021, pp. 867–876.

[14] V. Haghighi, "From classic gnns to hyper-gnns for detecting camouflaged malicious actors," in *WSDM*, 2023, pp. 1220–1221.

[15] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond Low-frequency Information in Graph Convolutional Networks," in *AAAI*, 2021, pp. 3950–3957.

[16] Y. Luo, "SHINE: SubHypergraph Inductive Neural nEtwork," in *NeurIPS*, 2022, pp. 18 779–18 792.

[17] J. Han, Q. Tao, Y. Tang, and Y. Xia, "DH-HGCN: Dual Homogeneity Hypergraph Convolutional Network for Multiple Social Recommendations," in *SIGIR*, 2022, pp. 2190–2194.

[18] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph Neural Networks," in *AAAI*, 2019, pp. 3558–3565.

[19] Y. Gao, Y. Feng, S. Ji, and R. Ji, "HGNN+: General Hypergraph Neural Networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3181–3199, 2022.

[20] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "HyperGCN: A New Method for Training Graph Convolutional Networks on Hypergraphs," in *NeurIPS*, 2019, pp. 1509–1520.

[21] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring, "Hypersage: Generalizing Inductive Representation Learning on Hypergraphs," *arXiv preprint arXiv:2010.04558*, 2020.

[22] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He, "Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection," in *WWW*, 2021, pp. 3168–3177.

[23] F. Shi, Y. Cao, Y. Shang, Y. Zhou, C. Zhou, and J. Wu, "H2-FDetector: A GNN-based Fraud Detector with Homophilic and Heterophilic Connections," in *WWW*, 2022, pp. 1486–1494.

[24] J. J. McAuley and J. Leskovec, "From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews," in *WWW*, 2013, pp. 897–908.

[25] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," *arXiv preprint arXiv:1609.02907*, 2016.

[26] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *NeurIPS*, 2017, pp. 1024–1034.

[27] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *ICLR*, 2018.

[28] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph Transformer Networks," in *NeurIPS*, 2019, pp. 11 960–11 970.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," in *NeurIPS*, 2017, pp. 5998–6008.

[30] S. Rayana and L. Akoglu, "Collective Opinion Spam Detection: Bridging Review Networks and Metadata," in *SIGKDD*, 2015, pp. 985–994.

[31] A. Paszke, S. Gross *et al.*, "Pytorch: An Imperative Style, High-performance Deep Learning Library," in *NeurIPS*, 2019, pp. 8024–8035.