algorithms

MDPI

Article

A Temporal Graph Network Algorithm for Detecting Fraudulent Transactions on Online Payment Platforms

Diego Saldaña-Ulloa [1,2,*], Guillermo De Ita Luna [1,*] and J. Raymundo Marcial-Romero [3]

[1] Faculty of Computer Sciences, Benemérita Universidad Autónoma de Puebla (BUAP), Puebla 72570, Mexico
[2] Moneypool, Monterrey 64650, Mexico
[3] Faculty of Engineering, Universidad Autónoma del Estado de México (UAEMEX), Toluca 50000, Mexico; jrmarcialr@uaemex.mx
* Correspondence: 022dccom0001@viep.com.mx (D.S.U.); guillermo.deita@correo.buap.mx (G.D.I.L.)

Abstract: A temporal graph network (TGN) algorithm is introduced to identify fraudulent activities within a digital platform. The central premise is that digital transactions can be modeled via a graph network where various entities interact. The data used to build an event-based temporal graph (ETG) were sourced from an online payment platform and include details such as users, cards, devices, bank accounts, and features related to all these entities. Based on these data, seven distinct graphs were created; the first three represent individual interaction events (card registration, device registration, and bank account registration), while the remaining four are combinations of these graphs (card–device, card–bank account, device–bank account, and card–device–bank account registration). This approach was adopted to determine if the graph's structure influenced the detection of fraudulent transactions. The results demonstrate that integrating more interaction events into the graph enhances the metrics, meaning graphs containing more interaction events yield superior fraud detection results than those based on individual events. In addition, the data used in this work correspond to Latin American payment transactions, which is relevant in the context of fraud detection since this region has the highest fraud rate in the world, yet few studies have focused on this issue.

Keywords: graph neural network; temporal graph network; fraud detection; event-based temporal graph

Citation: Saldaña-Ulloa, D.; De Ita Luna, G.; Marcial-Romero, J. R. A Temporal Graph Network Algorithm for Detecting Fraudulent Transactions on Online Payment Platform. *Algorithms* **2024**, *17*, 552. https://doi.org/10.3390/a17120552

Academic Editors: Massimiliano Caramia and Frank Werner

Received: 28 August 2024
Revised: 5 November 2024
Accepted: 11 November 2024
Published: 3 December 2024

1. Introduction

Transaction fraud affects different sectors of society, from individuals who use online services to companies and businesses that sometimes have to absorb the economic costs due to these illegal operations, in addition to the reputational damage they entail. This problem generates significant economic losses worldwide and has increased with the advance of the digital age. Therefore, this work focuses on digital fraud committed on online payment platforms.

This activity typically arises from the illicit use of credit and debit cards obtained unlawfully. Another contributing factor is unauthorized transactions resulting from unauthorized access aimed at stealing funds from user accounts. In light of this, fraudulent activities can be detected at transaction and user account levels. At the account level, this generally includes transactions and other actions that fraudulent users might carry out on online platforms.

Traditionally, digital fraud solutions have been tackled from various angles, including the use of machine learning rules and algorithms [1,2] that can effectively prevent some of the most common cases. Fraud detection is a complex issue that encompasses several challenges. For instance, it falls under the category of imbalanced class problems, leading some approaches to address it through class weighting [2]. The camouflage issue arises when fraudsters imitate normal behavior to execute fraudulent activities [3]. The effectiveness of detection also heavily relies on the size of the data used [4]. Moreover, it is a dynamic process where uncovering hidden patterns can significantly aid in prevention [5].

However, many machine learning algorithms fail to account for the full context in which a transaction occurs.

The context in which a fraudulent transaction occurs is crucial for identifying relevant characteristics useful for its detection. Some features are more significant than others and can play a crucial role in differentiating fraudulent transactions from legitimate transactions [6]. Users' different actions within a digital platform contribute to this context. Within these, intrinsic characteristics of the transaction can be found, such as amounts, types of card, dates [1], and even patterns related to user behavior such as the frequency and recency of transactions [6], as well as the temporality about the dynamics of the problem [7]. All these characteristics offer a broader view whose purpose will always focus on distinguishing the types of operations.

This work applies a temporal graph network (TGN) algorithm to predict fraudulent events using real data from an online payment platform. The TGN algorithm operates with an Event-Based Temporal Graph (ETG), a graph representation for a dynamic continuous approach (not a discrete approach, nor does it use temporal snapshots of a graph). This means that one of the main contributions of this work is the construction of an ETG based on real data, which serve as the input for the TGN algorithm. The data were provided by Moneypool, an online payment platform with clients worldwide but with a more significant presence in Mexico. This dataset is of utmost importance as it represents a sample of transactions in Latin America, particularly from Mexico. Latin America is the region with the highest rate of credit card fraud in the world [8], so the availability of this kind of information in a work of this type represents a contribution to the use and development of tools to attack this problem. As of the time of writing this work, only [9] uses real transaction datasets with a TGN methodology. Thus, one of the main purposes is to evaluate how adding different types of events influences the metric results.

The contributions of this work are highlighted in the following lines:

- The proposal of an algorithm with a TGN approach, incorporating explicit training, validation, and testing steps, with a focus on fraud detection. The algorithm processes graph data in streaming mode, i.e., its suitability is the dynamic continuous case rather than a discrete approach.
- The experimental validation that structural and temporal information (in a graph) is related to improving the metric results for detecting fraud when using a Graph Neural Network approach based on TGN. This was achieved by combining several ETGs and validating the notion that integrating more interaction events enhances the metric results.
- We have found experimentally that increasing the number of features (edge features related to interaction events in a graph) improves the metric results.
- Within the scope of our knowledge, we present the first work correlating the density of the graph, which models temporal digital transactions, with the F1 and AUC metrics coming from an ETG neural network. And a direct positive correlation is obtained.
- Furthermore, the repository of source data comes from a real online payment platform with information from Latin American cases, a region with the highest fraud rate [8], and where there is a lack of work focused on using real data [10].

This paper is structured as follows: Section 1 provides a general introduction. Section 2 shows work related to our algorithmic proposal. Section 3 introduces theoretical concepts and the dynamic graph neural network to be used. In Section 4, the main model to be used in the design of our algorithmic proposal is introduced. Section 5 shows characteristics for data and for the algorithm to be used, as well as the experimental results obtained. We add a section (Section 6) to present a brief discussion about the results obtained. The final section offers conclusions drawn from this work.

2. Related Work

In recent years, Graph Neural Networks (GNNs) have been applied to a wide range of problems [11]. Utilizing graphs as a data structure provides valuable insights by analyzing the interactions between its elements (vertices and edges). With the integration of neural

networks, processing these interactions can be achieved by incorporating techniques such as message passing [12], attention mechanisms [13], heterogeneity [14], or the temporal dynamics of the graph [15], among others.

The temporal dimension is crucial for capturing the interactions between various entities within a graph. Temporal graphs introduce an additional layer of information that can be examined through a discrete or continuous perspective. Graph Neural Networks designed for discrete and continuous time graphs leverage this temporal data and have been applied to address a range of challenges [16–18]. The temporal dependencies within a graph are vital for many processes [9,19], as they can enhance the accuracy of classification or prediction tasks involving vertices, edges, or entire graphs by incorporating these temporal details.

By leveraging Graph Neural Networks, the structural information of graphs can be utilized for fraud prediction on online platforms [3,5], as fraudulent activities often tend to cluster due to network homophily (the tendency for individuals to form small, homogeneous groups). Recently, there has been an increasing interest in applying these methods to real-world operational data, primarily because access to such information has traditionally been limited.

A new approach involves using Graph Neural Networks (GNNs) for fraud detection. The central premise is that fraud can be represented as a graph or network where various entities interact. Additionally, homophily can be relevant in this context, as fraud often exhibits clustering behavior.

Moreover, the increasing interest in using GNNs with dynamic graphs has led to their application in detecting fraudulent activities. For instance, [5] explores dynamic homogeneous graphs within a transactional graph segmented into temporal snapshots. This approach employs a GNN message-passing module, attention mechanisms, and convolutional aggregation within each temporal snapshot. Processing data as a temporal sequence of graphs enables classifying fraudulent transactions.

Other studies have similarly focused on utilizing temporal snapshots to identify fraudulent accounts and transactions [7]. This approach involves creating temporal and structural graphs processed through multiple layers of a Graph Convolutional Network (GCN). The timestamp information, which indicates when an event occurs, is combined with the vertex features, and the representation of two connected vertices is derived from the product of their embeddings (a transformation that maps from a high-dimensional space, like a graph, to a lower-dimensional space, like a vectorial space). Likewise, [20] adopts a structural and temporal approach, generating embeddings for connected vertices and their temporal relationships by chaining convolutional layers of both the structural and temporal graphs. This method is applied to detect large-scale fraudulent account registrations.

In [21], a heterogeneous dynamic graph detects fraudulent transactions. It uses three types of graphs (static, instantaneous temporal, and two-stage directed) obtained from transaction logs. The static and instantaneous graphs construct the two-stage directed graph used in the training and inference. The training combines a GCN (Graph Convolutional Network) layer for the batch graph (historical part of the two-stage directed graph) and a GCN layer for the real-time graph (current entities and transactions). The contribution of [21] is that the results of the batch graph are stored in a database. In this way, training and inference reduce the latency times of traditional methods.

The previous works have primarily concentrated on discrete temporal graph applications, with limited attention given to the continuous aspect. In [9], the temporal graph network (TGN) is employed for classifying fraudulent transactions and events, along with a methodology for asynchronous training, which is particularly useful for handling large-scale graphs. TGN involves using a GNN in continuous-time scenarios, where vertices and edges are linked to interaction events via timestamps. The TGN also incorporates a memory module, which stores information about past interactions of each vertex with its neighbors. Additionally, the TGN integrates other modules, such as attention mechanisms, to enhance performance.

In addition, previous works, such as [22], have only addressed fraud detection in online payment platforms by incorporating a limited number of events and static graphs, without

adding the temporal aspect or using an ETG (Event-Based Temporal Graph). Other works that have approached fraud detection from the dynamic point of view do so using a discrete dynamic perspective; the graph is divided into snapshots in discrete time intervals [5,7,20,21,23]. Meanwhile, the current work uses an ETG, which is a dynamic graph for the continuous case, rather than a discrete/snapshot-based graph. Furthermore, these works consider the temporal aspect as an extra parameter of the algorithm, for example, combining different features with timestamps of the graph or encoding the temporal parameter produced by a neural network, as the temporal aspect is seen as recurrent layers of information.

3. Dynamic Graph Neural Networks

Graph Neural Networks (GNNs) are neural network frameworks that process data structured as graphs. Their primary objective is to produce effective embedding representations for vertices, edges, or entire graphs, enabling them to address key tasks such as vertex/edge classification, edge prediction, and graph generation. GNNs achieve this through the widely recognized message-passing method, which involves exchanging information between neighboring vertices within a local subgraph. This concept is extended to include the time dimension when applied to temporal graphs.

When graphs are used to model a problem, their dynamic structure provides a broader view of the phenomenon's characteristics. Depending on the context, the properties of edges and vertices could evolve. Therefore, Dynamic Graph Neural Networks, particularly temporal graph networks, play an essential role in studying dynamic problems [19,24]. Many real-world scenarios, such as social networks, financial transactions, weather forecasting, or even recommendation systems, exhibit temporal dependencies where the interactions are not static. In those scenarios, past events contain valuable information that helps to describe the temporal dependencies, i.e., the dynamic context is helpful to obtain better graph representations and, consequently, better predictions.

Incorporating dynamic information can enhance the performance of machine learning models, and for GNN, this is associated with the expressive power of the architecture. In graph machine learning, expressive power means how accurate the model is in capturing the intrinsic patterns within a graph, the ability to distinguish between different graph structures, or the generalization to unseen data, to mention a few cases. This notion of expressive power is also related to how accurate the embeddings generated by the algorithm would be. The following subsections help to clarify how this process is performed. Table 1 shows some definitions used in this work.

Table 1. Math definitions used in this work.

Symbol	Meaning
$\mathbb{R}^{d_E}$	The set and dimension of vector features
$\mathbb{R}^{d_V}$	The set and dimension of edge features
V_T	The set of temporal vertices defined by a tuple
E_T	The set of temporal edges defined by a tuple
$\mathcal{N}_T(n)$	The temporal neighborhoods of vertex n
V_j	The set of vertices associated to a snapshot
E_j	The set of edges associated to a snapshot
$\varepsilon_V^{\pm}$	Vertex insertion/deletion event
$\varepsilon_E^{\pm}$	Edge insertion/deletion event
$h_v^k(t)$	Embedding of vertex v at time t and iteration k
$\mathcal{M}_v(t)$	Message aggregation of vertex v at time t
$s_v(t^+)$	Updated state of vertex v due to events at time t
$z_v(t)$	The embedding of vertex v after a process of an GNN

3.1. Temporal Graph Networks

To review more theoretical information about what is presented in this section, see [25]. A static graph G is defined as the tuple $(V, E, \mathcal{V}, \mathcal{E})$, where V is the set of vertices, $E \subseteq V \times V$ the set of edges; notice that directed graphs are being considered. Each vertex $v \in V$ has

$x_v \in \mathcal{V} : \mathcal{V} \in \mathbb{R}^{d_\mathcal{V}}$ features and each edge $(u, v) \in E$ has $e_{uv} \in \mathcal{E} : \mathcal{E} \in \mathbb{R}^{d_E}$ features. In addition, $\mathcal{N}(u) := \{v \in V | (u, v) \in E\}$ is the set of neighborhoods of u in G.

Temporal Graphs: A temporal graph G_τ is defined as the tuple (V, E, V_τ, E_τ), where V and E are the set of vertices and edges, respectively, that may appear at any time. Some temporal vertices and edges have time-dependent features such that $V_\tau := \{(v, x_v, t) \mid v \in V, x_v \in \mathcal{V}, t \in \tau\}$, $E_\tau := \{(e, x_e, t) \mid e \in E, x_e \in \mathcal{E}, t \in \tau\}$, where $\tau = \{t_1, t_2, ..., t_f\}$ is a set of timestamps. The temporal neighborhood of u is given by

$$\mathcal{N}_\tau(u) := \{v \in V \mid \exists\, (e, x_e, t) \in E_\tau, e = (u, v)\}. \tag{1}$$

If the timestamps on a temporal graph (TG) follow a fixed time step, the TG becomes a Discrete-Time Temporal Graph.

Discrete-Time Temporal Graph (DTTG): Let $\Delta\tau > 0$ be a fixed time-step and let $t_1 < t_2 < ... < t_n$ be timestamps with $t_{k+1} = t_k + \Delta\tau$. A DTTG G_{DT} is a TG where for each $(v, x_v, t) \in V_\tau$ or $(e, x_e, t) \in E_\tau$, the timestamps are taken from the set of fixed timestamps.

3.2. Dynamic Graph Neural Networks, Discrete and Continuous Case

Temporal graphs can be represented as a stream of static graphs or vertex/edge events. The former is referred to as a Snapshot-based Temporal Graph, while the latter is known as an Event-based Temporal Graph. This work focuses on designing an algorithm based on Event-based Temporal Graphs, i.e., a continuous dynamic approach.

Snapshot-based Temporal Graph (STG): Let $t_1 < t_2 < ... < t_n$ be an ordered set of timestamps in G_τ. Let $V_j := \{(v, x_v, t_i, t_f) \mid v \in V_\tau, t_i \leq t_j \leq t_f\}$, $E_j := \{(e, x_e, t_i, t_f) \mid e \in E_\tau, t_i \leq t_j \leq t_f\}$ be the sets of vertices and edges that exist between an initial time t_i and a final time t_f and $G_j = (V_l, E_l)$, $l = 1, 2, ..., n$ be the graph snapshots. A Snapshot-based Temporal Graph is the sequence

$$G_\tau^S = \{(G_l, t_l) \mid l = 1, 2, ..., n\}. \tag{2}$$

The literature indicates STGs are related to DTTGs because the snapshots are at periodic fixed time intervals.

Another form of graph representation occurs when the data model a continuous process, i.e., a dynamic continuous temporal graph. However, this approach is better modeled as a stream of events, so it receives the name of an Event-Based Temporal Graph.

Event-based Temporal Graph (ETG): Let G_τ be a temporal graph and $\mathcal{E}$ one of the following events:

- Vertex insertion $\mathcal{E}_V^+ := (v, t)$: There is an addition of the vertex v to G_τ at time t.
- Vertex deletion $\mathcal{E}_V^- := (v, t)$: There is a removal of the vertex v from G_τ at time t.
- Edge insertion $\mathcal{E}_E^+ := (e, t)$: There is an addition of the edge e to G_τ at time t.
- Edge deletion $\mathcal{E}_E^- := (e, t)$: There is a removal of the edge e from G_τ at time t.

An ETG of a temporal graph is a sequence of events

$$G_\tau^E = \{\mathcal{E} | \mathcal{E} \in \{\mathcal{E}_V^+, \mathcal{E}_V^-, \mathcal{E}_E^+, \mathcal{E}_E^-\}\}. \tag{3}$$

Message-Passing TGNN—MPTGNN: Let G_τ be a temporal graph with parameters V_τ and E_τ. At each iteration of the MPTGNN, an embedding $h_v^k(t)$ is updated with neighborhood information $\mathcal{N}_{\tau(v)}$. According to [24], the message passing can be expressed as

$$h_v^{(k+1)}(t) = \text{UPDATE}^k\Big(h_v^{(k)}(t), \text{AGG}^k\Big(\{h_u^{(k)}(t), \forall u \in \mathcal{N}_{\tau(v)}\}\Big)\Big) \tag{4}$$

$$= \text{UPDATE}^k\Big(h_v^{(k)}(t), m_{\mathcal{N}(u)}^{(k)}(t)\Big), \tag{5}$$

where UPDATE is a differentiable function (a neural network); AGG could be some permutation-invariant operation such as mean, max-pooling, sum, etc.; $m_{\mathcal{N}(u)}(t)$ is the

message that aggregates temporal neighborhood information about the vertex n and $\mathbf{h}_u^{(0)}(t) = s_u(t)$ is the state of v at time t. The TGNN algorithm proposed uses the concept of memory as a set of vectors that summarizes the history of a vertex and is updated whenever an event occurs. Given a set of events $\mathcal{E}$, the state of v is updated based on $\mathcal{E}$ as

$$\mathcal{M}_v(t) = \text{MEMAGG}\Big(\{(s_u(t), s_v(t), t - t_v, \mathcal{E}_{uv}(t)) \mid (u, v, t) \in \mathcal{E}\}\Big) \tag{6}$$

$$s_v(t^+) = \text{MEMUPDATE}\Big(s_v(t), \mathcal{M}_v(t)\Big), \tag{7}$$

where $s_v(0) = x_v$. According to [25], and given the previous definitions, the embeddings in a TGNN can be expressed as

$$h_v^{(k+1)}(t) = \text{UPDATE}^k\Big(h_v^{(k)}(t), \text{AGG}\{\mathcal{M}_\mathcal{E}, \mathcal{E} = \mathcal{E}_E^{\pm} = (u, t') \text{ with } u \in \mathcal{N}_{\mathcal{T}(v)}\}\Big), \tag{8}$$

where $\mathcal{E}_E^{\pm} = (u, t')$ with $u \in \mathcal{N}_t(v)$ is the addition or deletion of a temporal neighbor of v. One thing to clarify is that the MPTGNN belongs to the TGN approach; from now on, this terminology will be used.

4. Using Learning Tasks to Model Online Transactions

Deep learning represents the next evolutionary step in machine learning techniques, which allow computational models to learn by examples. As a mature research area, deep learning has received much interest and experimentation in online fraud detection. Diverse variants of deep learning models exist, including but not limited to Convolutional Neural Networks, Recurrent Neural Networks, adversarial Neural Networks, and deep autoencoders, each of which is frequently applied to the online fraud detection research domain [10]. In this work, dynamic graph neural networks are used to model online transactions and detect digital fraud.

Some learning mechanisms can be identified within machine learning, and they can be differentiated depending on whether training and test sets are present during training and testing. An inductive learning approach means the algorithm is trained exclusively with the training set, and the inference is applied to unseen data. A transductive learning approach means that the training occurs with the train set, some information about the test set is available (for example, neighborhood vertices overlapping with the train set), and the inference is performed on the test set.

Learning tasks in a supervised approach include classification, regression, or prediction. This work focuses on vertex classification tasks applying an MPTGNN algorithm based on an Event-Based Temporal Graph (ETG).

Temporal Vertex Classification. Let $G_\mathcal{T} = (V, E, V_\mathcal{T}, E_\mathcal{T})$ be a temporal graph. The vertex classification consists in learning the function $f_{VC} : V \times \mathbf{R}^+ \to \mathcal{C}$, which maps each vertex class to a class $\mathcal{C} = \{1, ..., L\}$ such that $\mathcal{C} \in \mathbb{Z}^+$ at time $t \in \mathbb{R}^+$. The function f_{VC} results from the whole (machine learning) training process.

During each iteration of a message-passing framework (as previously seen), an embedding $\mathbf{h}_v^{(k)}(t)$ is computed; after k iterations of an algorithm, the output of the final layer represents the embedding of the vertex v such that

$$z_v(t) = h_v^{(k)}(t), \ \forall v \in V. \tag{9}$$

With this information, tasks such as vertex classification can be performed. To predict the label of a vertex v, the Softmax function is used to map input vectors and normalize them to a probability distribution, as in the next equation:

$$\hat{p} = \text{Softmax}(z_v)_i = \frac{e^{z_i}}{\sum_{c=1}^{L} e^{z_c}}, \tag{10}$$

where $i \in \{1, ..., L\}$, $\hat{p} : \mathbb{R}^L \to (0,1)^L$, $\hat{p}$ represents the probability of vertex v to belonging to the L-th class of y_v and $\mathbf{z}_v = (z_{v,1}, ..., z_{v,L}) \in \mathbb{R}^L$. The complete process to obtain a vertex classification consists of minimizing a loss function, such as the Cross-Entropy Loss (CE) used for multiclass classification. CE measures the difference between the predicted class probabilities and the true class labels:

$$\text{CE} = \sum_{i=1}^{L} y_{v,i} log(\hat{p}_i), \tag{11}$$

where $y_{v,i}$ is the actual label (ground truth) of the vertex v under the class i. $\hat{p}_i$ is the Softmax probability for the ith class, and L is the total number of classes. The minimization process is performed at each iteration of the training and involves the backpropagation algorithm [26] to modify the weights of the neural networks mentioned in the previous section.

On the other hand, when a classification process involves an imbalance of the label classes, techniques must be adopted so that this imbalance does not have negative consequences on the classification, for example, an impact on the proportion of classes predicted with a specific label due to a bias of the majority classes. To solve this problem, one of the commonly used techniques is class weighting [27]. This refers to an assignment of weights that moderates the importance of the classes during each iteration of the algorithm (training). The weight assignment is entered as a weighting factor in the loss function (for example, cross-entropy loss). In such a manner, the minority classes are weighted with higher weights, and the algorithm increases the penalty errors, causing the prediction to improve in each iteration due to the minimization process of the loss function. The weights can be calculated as an inverse of the class frequency

$$w_L = \frac{N_{\text{Total}}}{N_L}, \tag{12}$$

where w_L is the weight of the L-th class, N_{Total} is the total number of samples, and N_L is the number of samples of the L-th class.

Modelling Fraud Detection Using Temporal Graph Networks

A Message-Passing Temporal Graph Neural Network (MPTGNN) algorithm will be employed to capture the characteristics of digital fraud. Graphs provide a way to represent information about the structural context, such as details about the entities linked to a particular user. This representation was adequate for the estimation of various statistical measures. A basic method involves applying traditional machine learning algorithms (like decision trees or neural networks) to extract and combine graph features with transaction-related features. However, while this method uses direct information from the graph, it overlooks the structural neighborhood context of vertices and, specifically, the types of vertices or edges that are connected.

A database of transactions and events on a digital platform can be modeled using either a homogeneous or heterogeneous graph, depending on the amount of information and the chosen approach, in a static or dynamic context. The model is more straightforward in the homogeneous scenario and might include only edges representing transactions between users or other entities. However, the heterogeneous model offers a richer representation, providing more context for each vertex and edge within the graph. Depending on the platform, the vertices could represent users, cards, devices, or IP addresses, while the edges might denote interactions involving a card, device, IP address, or transaction. Additionally, each vertex and edge is associated with a set of features that typically relate to each entity's transactional activity or behavior, such as users within the digital platform.

Alternatively, the graph can be modeled using a static or dynamic approach. Suppose the data include timestamps that indicate when an edge connects two vertices and represent various events, such as card registration, payment attempts, device connections, and other specific occurrences on the online platform. In that case, the information can

be modeled using an Event-Based Temporal Graph (ETG). This type of representation offers the advantage of incorporating time as an additional layer of information, which is particularly useful for fraud prevention.

Using an Event-Based Temporal Graph (ETG), the MPTGNN algorithm can be employed for fraud prevention classification tasks. In this context, the ETG, which includes features, edges, vertices, and timestamps, serves as the input graph to be processed by the MPTGNN algorithm (as outlined in the previous section). The proposed algorithm extends the MPTGNN algorithm presented in [19] to classify fraudulent activities conducted by malicious users, essentially performing a binary classification (fraudulent vs. normal). The pseudocode in Algorithm 1 demonstrates the use of an MPTGNN and the use of the memory module discussed earlier. Similarly, the classification task is executed as described in the prior section. The diagram in Figure 1 helps to illustrate the process. The raw data need to be converted into an ETG, and this serves as input for the algorithm.

Algorithm 1 MPTGNN algorithm

Input: ETG $G_{\mathcal{T}}^{E}$, Train ratio r_{train}, Validation ratio r_{val}, Test ratio r_{test}, Total number of timestamped events of the graph n, Number of epochs *n_epochs*

Output: Predicted y class, F_1^{train}, AUC^{train}, F_1^{val}, AUC^{val}, F_1^{test}, AUC^{test}

$n_{train}, n_{val}, n_{test} \leftarrow (r_{train} \times n), (r_{val} \times n), (r_{test} \times n)$ ▷ *Get the size of each set*
ChronOrder$(G_{\mathcal{T}}^{E})$ ▷ *Data in chronological order according to timestamps*
$G_{\mathcal{T}}^{E_{train}} \leftarrow G_{\mathcal{T}}^{E}[1 : n_{train}]$ ▷ *Get the train, val and test sets in chronological order*
$G_{\mathcal{T}}^{E_{val}} \leftarrow G_{\mathcal{T}}^{E}[n_{train} + 1 : n_{train} + n_{val}]$
$G_{\mathcal{T}}^{E_{test}} \leftarrow G_{\mathcal{T}}^{E}[n_{train} + n_{val} + 1 : n]$
states $\leftarrow 0$ ▷ *Start the memory states to zero*
mem_raw $\leftarrow \{\}$ ▷ *Start the raw messages*
for *epochs* $:= 1 \rightarrow$ *n_epochs* **do**
    for each *batch* $(u, v, e, t) \in G_{\mathcal{T}}^{E_{train}}$ **do** ▷ *Start training step*
        mem $\leftarrow \text{msg}(\textbf{mem\_raw})$ ▷ *Message computation*
        $\bar{\textbf{mem}} \leftarrow \text{agg}(\textbf{mem})$ ▷ *Combine the messages*
        $\hat{\textbf{states}} \leftarrow \text{mem}(\bar{\textbf{mem}}, \textbf{states})$ ▷ *Update the memory*
        $\mathbf{z}_u, \mathbf{z}_v \leftarrow \text{emb}(\mathbf{u}, \mathbf{t}), \text{emb}(\mathbf{v}, \mathbf{t})$ ▷ *Compute the embeddings*
        $y \leftarrow \text{sigm}(\mathbf{z}_i)$ ▷ *Apply sigmoid funct. to get the class*
        Train Loss $= w_n \times \text{CE}(y, c_n)$ ▷ *Compute the weighted CE Loss*
        $F_1^{train}, \text{AUC}^{train} \leftarrow F_1(y, c_n), \text{AUC}(y, c_n)$ ▷ *Compute metrics on train data*
        $\textbf{mem\_raw}_u, \textbf{mem\_raw}_v \leftarrow (\hat{\textbf{states}}_u, \hat{\textbf{states}}_v, \mathbf{t}, \mathbf{e})$ ▷ *Compute mem. for next batch*
        $\textbf{mem\_raw}_u, \textbf{mem\_raw}_v \leftarrow (\hat{\textbf{states}}_v, \hat{\textbf{states}}_u, \mathbf{t}, \mathbf{e})$
        $\textbf{m\_raw} \leftarrow \text{store\_raw\_message}(\textbf{m\_raw}_u, \textbf{m\_raw}_v)$ ▷ *Raw mem. for next batch*
        $\mathbf{s}_u, \mathbf{s}_v \leftarrow \hat{\mathbf{s}}_u, \hat{\mathbf{s}}_v$ ▷ *Updated mem. states for next batch*
    end for
    Repeat steps 10 to 14 on $G_{\mathcal{T}}^{E_{val}}$ ▷ *Compute embeddings on Val data*
    Val Loss $= \text{CE}(y, c_n)$ ▷ *Compute the Val Loss*
    if Val Loss is not decreasing **then** ▷ *Early stopping*
        Stop algorithm
    end if
    $F_1^{val}, \text{AUC}^{val} \leftarrow F_1(y, c_n), \text{AUC}(y, c_n)$ ▷ *Compute metrics on Val data*
    Repeat steps 10 to 14 on $G_{\mathcal{T}}^{E_{test}}$ ▷ *Compute embeddings on Test data*
    Test Loss $= \text{CE}(y, c_n)$ ▷ *Compute the Test Loss*
    $F_1^{test}, \text{AUC}^{test} \leftarrow F_1(y, c_n), \text{AUC}(y, c_n)$ ▷ *Compute metrics on test data*
end for

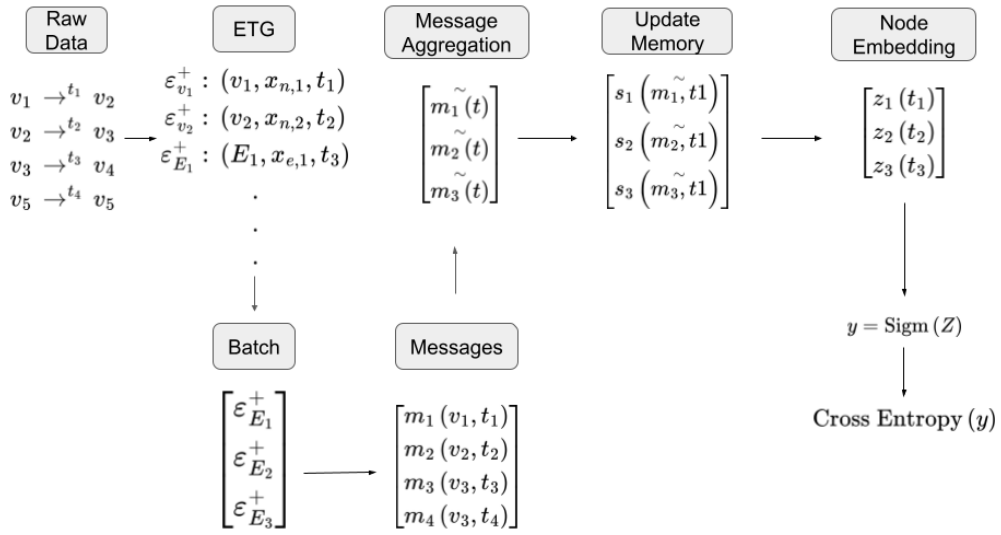


Figure 1. Diagram of the process followed by the MPTGNN. The process begins with the raw data provided by the payment platform. The data must be converted to an ETG to be processed by the algorithm. The steps described here summarize Algorithm 1.

Applying the MPTGNN algorithm with ETG data focused on fraud prevention involves directly classifying fraudulent interaction events initiated by fraudulent users. In this work, the algorithm processes all event streams, with timestamps of various events, including those previously mentioned, recorded on a real online digital payment platform.

5. Results

In this article, an algorithm designed to work with an Event-Based Temporal Graph (ETG) is developed to detect fraudulent activities carried out by fraudulent user accounts through a vertex classification process, specifically a binary classification (fraud vs. non-fraud). As previously mentioned, the data were provided by Moneypool, an online payment platform with global clients, though it is particularly prominent in Mexico. The edges include 43 features calculated at the time of each interaction (as outlined in Table 2). These features incorporate information from the connected vertices and they are related to their behavior on the platform, such as navigation patterns, interactions with various platform elements, and transaction behaviors.

Table 2. Example of some records of the ETG used in this work. The data did not contain vertex features, but an ETG could generally contain up to $d_{\mathcal{V}}$ vertex features.

Event	Vertex/Edge	Vertices involved	Features	Timestamp
$\mathcal{E}_V^+$	v_1	N/A	$[x_{1,1}, ..., x_{1,d_{\mathcal{V}}}]$	21 March 2023 00:00:00
$\mathcal{E}_V^+$	v_2	N/A	$[x_{2,1}, ..., x_{2,d_{\mathcal{V}}}]$	21 March 2023 00:01:00
$\mathcal{E}_E^+$	e_1	$v_1 \to v_2$	$[x_{1,1}, ..., x_{1,43}]$	21 March 2023 00:01:30
$\mathcal{E}_V^+$	v_3	N/A	$[x_{3,1}, ..., x_{3,d_{\mathcal{V}}}]$	21 March 2023 00:10:00
$\mathcal{E}_V^+$	v_4	N/A	$[x_{4,1}, ..., x_{4,d_{\mathcal{V}}}]$	21 March 2023 00:15:00
$\mathcal{E}_E^+$	e_2	$v_3 \to v_4$	$[x_{2,1}, ..., x_{2,43}]$	21 March 2023 00:25:00

The dataset spans two years, from March 2021 to July 2023, and includes the characteristics detailed in Tables 3 and 4. Each edge in the graph is timestamped, consistent with the ETG's properties.

Table 3. Characteristics of the graphs utilized in this study. The numerical values represent the counting of vertices or edges within each graph. Each graph is a bipartite graph between users and entities (cards, devices, bank accounts, etc.), i.e., entities are a subset of the vertices. This term is a standardization used in the online payment platform to refer to vertices that are not users. The edge-to-vertex ratio represents an intuitive density measure of edges over vertices.

Graph	Vertex	Edges	Users	Entities	Edge-to-Vertex Ratio
Cards	533,761	318,241	224,258	309,503	59.62
Devices	752,973	508,221	261,612	491,361	67.49
Bank Accounts	196,343	124,773	82,122	114,221	63.54
Cards–Devices	1,100,761	826,462	299,897	800,864	75.08
Cards–Bank Accounts	655,785	443,014	232,061	423,724	67.55
Devices–Bank Accounts	869,315	632,994	263,733	605,582	72.81
Cards-Devices–Bank Accounts	1,215,588	951,235	300,503	915,085	78.25

Table 4. Number of normal and fraud events present in each graph. The numerical values in the "Normal" and "Fraud" columns indicate the corresponding number of events. The "Fraud Percentage" indicates the percentage of fraudulent events in the graph.

Graph	Normal	Fraud	Fraud Percentage
Cards	314,894	3347	1.05
Devices	505,797	2424	0.48
Bank Accounts	123,574	1199	0.96
Cards–Devices	820,691	5771	0.7
Cards–Bank Accounts	438,468	4546	1.02
Devices–Bank Accounts	629,371	3623	0.57
Cards–Devices–Bank Accounts	944,265	6970	0.73

All features were sourced from the online payment platform's feature store, a database designed to compute and store machine learning features. Additionally, due to ethical considerations in fraud detection, the exact set of features is not disclosed; instead, they are grouped into general categories, such as navigation patterns, platform behavior, and transaction behavior, as previously mentioned. This approach is taken to prevent the misuse of fraud prevention research, ensuring it does not serve as a guide for fraudsters, as discussed in [28].

5.1. Characteristics of the Data and the Algorithm

The events considered in this work encompass various interactions, including card registration, device registration, IP registration, official identification registration, and bank account registration. After constructing the ETG, it was divided into training, validation, and test sets using a straightforward chronological split (70%, 15%, 15%). Specifically, the training set comprises the first 70% of the data in chronological order, followed by the validation set with the next 15%, and the test set with the remaining 15% of the data. The algorithm samples data in each iteration during the training process to enhance efficiency, as suggested in [12].

In this case, data sampling focuses on the n neighboring vertices with temporal interactions occurring before an event. The MPTGNN algorithm incorporates information from these neighboring vertices, and for this study, the 20 neighboring vertices with the most recent interactions were selected.

As outlined in the previous section and illustrated by the MPTGNN Algorithm 1, four key modules are central to the MPTGNN: the message function, message aggregator, memory updater, and embedding module. The message function gathers and processes information on the memory states of the vertices involved in an interaction event (which creates an edge between two vertices). This is accomplished using a Multi-Layer Perceptron (MLP—a fully connected neural network with at least three layers and a non-linear activation function). In this setup, the memory states serve as inputs to the MLP, and the output is utilized in the subsequent steps, as shown in lines 4 and 5 of Algorithm 1. Following this, the message aggregator combines the information from the message function associated with the same vertex. Typically, this combination can be achieved using a learnable function, but only the most recent message from a given vertex is used for simplicity.

When the information regarding the messages and aggregation of each vertex has been completed, another function is used during the memory updater step, which consists of a recurrent neural network, like GRU [29]. Through this module, the information on the states of each vertex is updated in the memory. Finally, the embedding module obtains temporal embeddings for each vertex using a generic time encoding function [30] and a neural network architecture based on multi-head attention mechanisms [31]; this attention function considers the temporal neighbors of a vertex. After obtaining an embedding, a temporal vertex classification process is performed according to the previous section. This step completes the fraud detection procedure, and its results rely on the available features and data.

As the implementation of [19], the TGN-type algorithm of the current work could be used with heterogeneous graphs; however, since no mechanism is used to distinguish between the different types of edges, this algorithm can work instead with bipartite graphs. The way to incorporate the information of each type of edge would be carried out directly on the features since they are interaction events (that is, different types of edges for different events). Each event has a different behavior than the others regarding their features. Considering this, the heterogeneous aspect of the graph was not taken explicitly in this work; such an approach to focus on an explicit heterogeneous aspect would involve additional considerations outside the current algorithmic approach (for example, [20] or [14]). However, as outlined in the following sections and subsections, the current proposal (algorithm and ETG construction) produces relevant results.

5.2. Experiments with Graph Combinations

It is of utmost importance to note that Algorithm 1 computes embedding from raw information, i.e., precomputed embeddings are not used during the training of the algorithm, in comparison to [19]. The entire process is evaluated using four types of vertices (users, cards, devices, and bank accounts), resulting in seven independent sets of bipartite graphs constructed from the data provided by Moneypool. The first three sets (as detailed in Table 3) consist solely of independent connections; one set links users through shared cards, another through shared devices, and the third through shared bank accounts. The remaining four sets (also listed in Table 3) combine these graphs, such as cards combined with devices, cards with devices and bank accounts, devices with bank accounts, and finally, cards with devices and bank accounts. No previous works use combinations of events in an ETG approach to serve as input for a TGN-type algorithm. Figure 2 illustrates how the temporal graph is configured, while Figure 3 presents a subgraph (sample) of the cards–devices–bank account graph for a schematic visualization. From Table 3, entities represent a subset of vertices. The term is a standardization used in the online payment platform to refer to vertices that are not users, and this helps clarify the difference between users, cards, devices, etc.

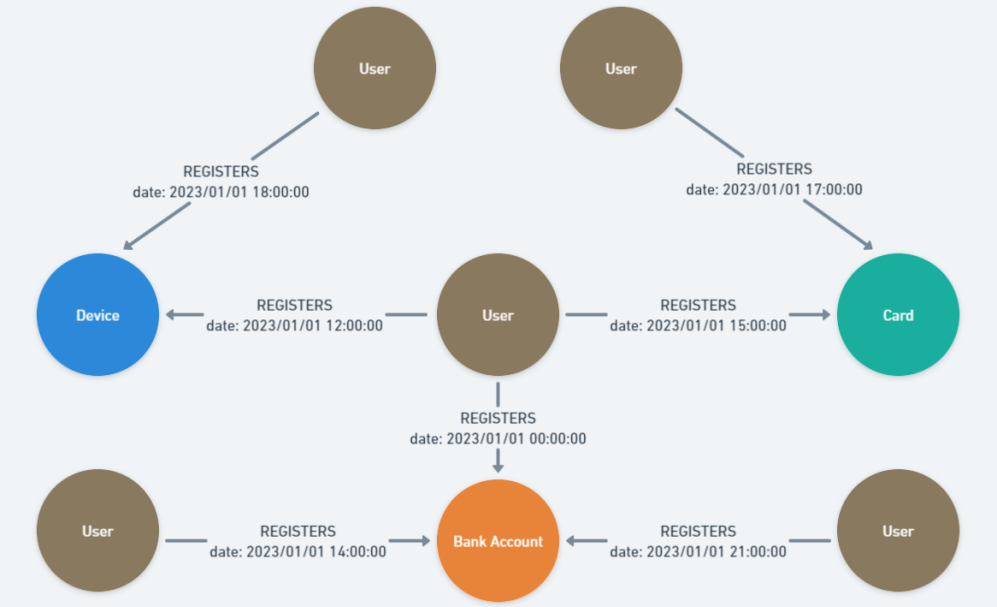


Figure 2. An example schema of the temporal graph with timestamps. The graph includes four distinct types of vertices: users (brown), devices (blue), cards (green), and bank accounts (orange). This particular graph represents the card–device–bank account registration, where each edge has an associated timestamp.

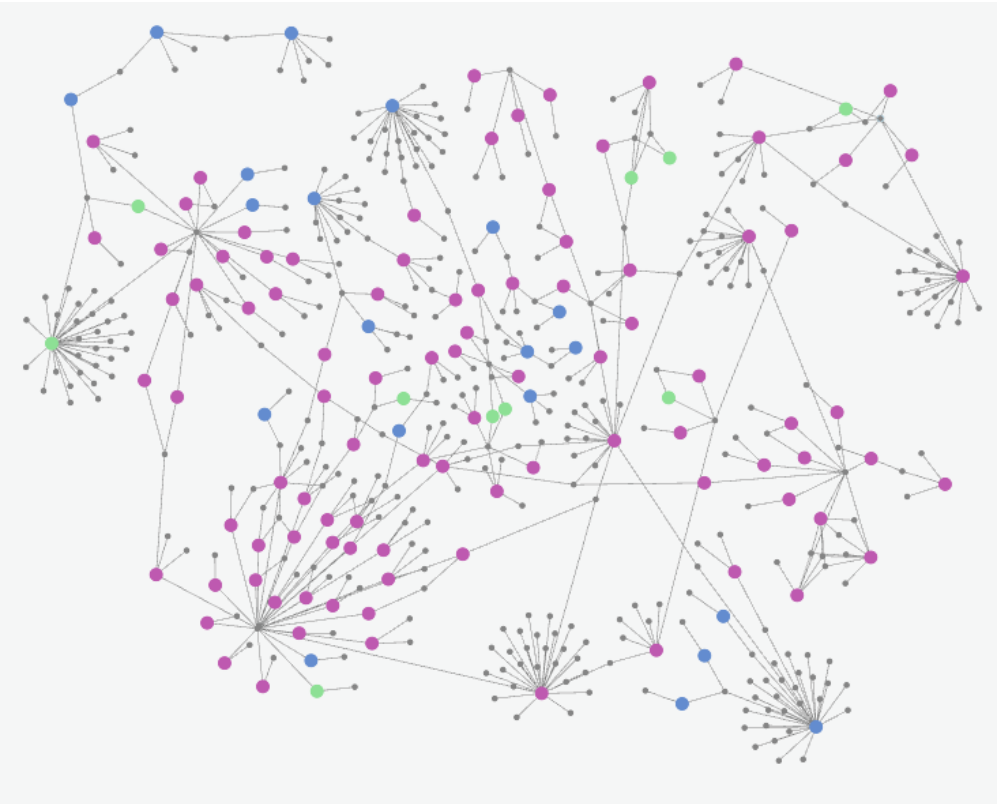


Figure 3. Sample subgraph of the cards–devices–bank accounts graph. Gray vertices represent users, blue vertices are devices, green vertices are cards, and purple vertices are bank accounts. In this sample, it can be observed that users share devices (blue vertices) and bank accounts (purple vertices) more prominently than they do cards (green vertices). In addition, some users (gray vertices) act as bridges (long edges) between some connected components of the graph. Each edge has a corresponding timestamp (not shown in this image). This is not a general behavior; the figure is only intended as an example.

The main objective of independently testing the MPTGNN algorithm on seven different graphs is to determine if the aggregation of structural information impacted the values for F_1 and AUC metrics. F_1 is a harmonic mean between precision and recall, two additional metrics related to true positives (observations that are predicted as fraud and indeed are fraud) and false positives (observations that are predicted as fraud but are not fraud). Because of these considerations, the F_1 metric is commonly used in imbalanced class problems.

On the other hand, AUC stands for the Area Under the Curve and is the area under the ROC curve; the ROC curve is a graphical representation between the true positives and false positives at each classification threshold. Tables 3 and 4 detail the characteristics of the seven graph sets, while Table 5 presents the results from the fraudulent user classification task. The values shown are the averages from 10 different runs. Given the class imbalance, class weighting was applied to the cross-entropy loss function.

Table 5. Results of the experiments performed on each graph. Each graph is a bipartite graph between users and cards, users and devices, and users and bank accounts.

Graph	F_1	AUC
Cards	0.3131 ± 0.2106	0.4578 ± 0.0678
Devices	0.1763 ± 0.0743	0.6755 ± 0.0525
Bank Accounts	0.1508 ± 0.0720	0.6285 ± 0.0142
Cards + Devices	0.4213 ± 0.1341	0.6727 ± 0.0552
Cards + Bank Accounts	0.3136 ± 0.2212	0.5330 ± 0.0737
Devices + Bank Accounts	0.4121 ± 0.1437	0.7014 ± 0.0249
Cards + Devices + Bank Accounts	0.4523 ± 0.0710	0.7255 ± 0.0196

An extended dataset (also provided by Moneypool) was used for completeness, consisting of the same events and additional IP address registration events. Also, the period is extended from March 2021 to February 2024, i.e., seven extra months of information. The extended data contain 118 features, and the characteristics can be observed in Tables 6 and 7.

Table 6. Characteristics of the extended graphs. The data span from March 2021 to February 2024.

Graph	Vertex	Edges	Users	Entities	Edge-to-Vertex Ratio
Cards	648,022	392,950	266,164	381,858	60.63
Devices	988,107	691,611	318,736	669,371	69.99
Bank Accounts	252,482	165,604	103,381	149,101	65.59
Cards–Devices	1,404,593	1,084,561	353,364	1,051,229	77.21
Cards–Bank Accounts	805,111	558,554	274,152	530,959	69.37
Devices–Bank Accounts	1,138,913	857,215	320,441	818,472	75.26
Cards–Devices–Bank Accounts	1,554,229	1,250,165	353,899	1,200,330	80.43
IPs–Devices–Bank Accounts	1,285,705	1,120,085	320,577	965,128	87.11
IPs–Cards–Devices	1,551,262	1,347,354	353,377	1,197,885	86.85

For these extended graphs, the percentage of fraud decreased (compared to Table 4) due to the expected behavior of the data provided by the digital platform. Despite that, the metric results improved significantly (see Tables 8 and 9). The reason for this is that there is more information related to the graph structure (more extended period), and also, there are more features that help to distinguish between fraud and expected behavior. At the same

time, the standard error also indicates that the results are more uniform at each trial, so the additional information is also related to a standardization of the algorithm's convergence.

Table 7. Number of normal and fraud events present in the extended graphs. Note that the percentage of fraud decreased because normal observations increased more than fraud observations.

Graph	Normal	Fraud	Fraud Percentage
Cards	389,231	3719	0.95
Devices	688,807	2804	0.41
Bank Accounts	164,280	1324	0.8
Cards–Devices	1,078,038	6523	0.6
Cards–Bank Accounts	553,511	5043	0.9
Devices–Bank Accounts	853,087	4128	0.48
Cards–Devices–Bank Accounts	1,242,318	7847	0.63
IPs–Devices–Bank Accounts	1,115,447	4561	0.41
IPs–Cards–Devices	1,340,398	6956	0.52

Table 8. Results of the experiments performed on the extended graphs.

Graph	F_1	AUC
Cards	0.5020 ± 0.0045	0.6192 ± 0.0184
Devices	0.3972 ± 0.0066	0.7166 ± 0.0081
Bank Accounts	0.6285 ± 0.0142	0.6819 ± 0.0158
Cards + Devices	0.5009 ± 0.0028	0.7291 ± 0.0104
Cards + Bank Accounts	0.4990 ± 0.0015	0.6699 ± 0.0052
Devices + Bank Accounts	0.4287 ± 0.0359	0.7341 ± 0.0034
Cards + Devices + Bank Accounts	0.4906 ± 0.0270	0.7376 ± 0.0081
IPs + Devices + Bank Accounts	0.4348 ± 0.0037	0.7516 ± 0.0112
IPs + Cards + Devices	0.4333 ± 0.0225	0.7554 ± 0.0070

Table 9. Additional metrics computed for some of the extended graphs. For these cases, it can be observed that graphs with more structural information improve the recall. On the other hand, the low values in precision indicate that for those graphs, the algorithm is catching moderate fraud observations at the expense of normal observations. This work does not focus on finding a balance between precision and recall. However, this last task can be considered for future research.

	Precision	Recall	Accuracy
Cards	0.4145 ± 0.0152	0.6433 ± 0.0226	0.8749 ± 0.0030
Dev	0.2738 ± 0.0058	0.7232 ± 0.0075	0.6313 ± 0.0070
BA	0.5692 ± 0.0197	0.7091 ± 0.0057	0.8695 ± 0.0058
IPs + Dev + BA	0.2978 ± 0.0017	0.8177 ± 0.0022	0.7428 ± 0.0105
IPs + Cards + Dev	0.2818 ± 0.0152	0.8272 ± 0.0396	0.8182 ± 0.1078

6. Discussion

From Table 5, the moderate values in the F_1 score indicate that class imbalance is a critical factor in this problem; the proportion of fraudulent observations is around 1% in almost all examined cases. Another factor contributing to the F_1 score results is the limited number of features distinguishing fraudulent users from legitimate ones. This study considered only basic features related to transactions and general behavior on the digital platform.

On the other hand, the AUC scores show more favorable results. A notable difference in the values for AUC can be observed when additional edges are included in the analysis, particularly when comparing graphs with single-edge types to those with two- or three-edge types.

In summary, it has been found that including more features and structural and temporal context enhances fraud detection, according to the values of the metrics: F_1 and AUC. Additionally, there appears to be no direct correlation between metric results and the percentage of fraudulent observations, suggesting that graph behavior information is more critical than the raw fraud data itself (as was shown in Tables 3 and 5).

When comparing the results with the additional event (IP address) to the same events of Table 5, it can be noticed that the aggregation of this event also helps to improve the metric results. This concurs with the previous outcomes, which show that adding more interaction events helps improve the metrics. For this work, there are only results of those new graph combinations because adding more events also has an impact on the execution time of the algorithm.

The improvement in the metrics due to the addition of more entities (for example, the cards–devices–bank accounts (Table 5) or IPs–cards–devices graphs (Table 8)) also means that more fraud events are captured in the detection. This can be appreciated from the results in Table 9, where the recall value (also known as true positive rate) tends to improve when adding more structural information. This makes sense from an empirical context because adding more data (IPs, devices, cards, bank accounts, etc.) has been noticed to help differentiate between fraud patterns and standard patterns.

As shown in Tables 5 and 8, adding structural information correlates with improving the metrics. This concurs with the empirical context of fraud detection, where fraudulent events are expected to exhibit different patterns from regular events by nonfraudulent users. Aggregating more data is beneficial from the point of view of having more context during the training process due to the use of deep learning with graphs [32,33]. On the other hand, the type of features considered span from behavior to transaction types. The improvement opens an opportunity for future works to explore if the combination of certain types of features in conjunction with new entities helps in the correct identification of fraud patterns.

Tables 10 and 11 show the percentage of improvement in the results when comparing the use of a single graph versus combinations of graphs (i.e., adding more events) and pairs of graphs versus the complete graph. In addition, Table 12 also shows the percentage of improvement when increasing the number of features. All these results highlight that adding more events (structural information) and features related to transactions/behavior helps to obtain better metric results. Something to note is that previous works related to fraud detection, such as [9] (the only one that uses a TGN approach for fraud detection), do not focus on metric improvements when more data are added (which is relevant in the task of fraud detection on online platforms). In contrast, they did not achieve any improvement for their proposal versus an algorithm on the TGN approach applied to fraud detection. Furthermore, other works cannot be compared because they use DTTG, i.e., they are the snapshot-based case.

Table 10. Percentage of improvement in the results using each single graph vs. the combined graphs. For each pair of values in the table, the first corresponds to the improvement over the F_1 score, and the second stands for the AUC. Generally, it can be observed that the aggregation of more events increases the percentage of improvement.

	Cards + Dev	**Cards + BA**	**Dev + BA**	**Card + Dev + BA**
Cards	34.55/46.94	0.1/16.42	N/A	44.45/58.47
Dev	38.96/0	N/A	33.74/3.83	56.55/7.40
BA	N/A	7.95/0	73.27/11.59	99.93/15.43

Table 11. Percentage of improvement in the results of pairs of combined graphs vs. the complete graph. The first number is the improvement over the F_1 score and the second stands for the AUC. Generally, it can be observed that the aggregation of more events increases the percentage of improvement.

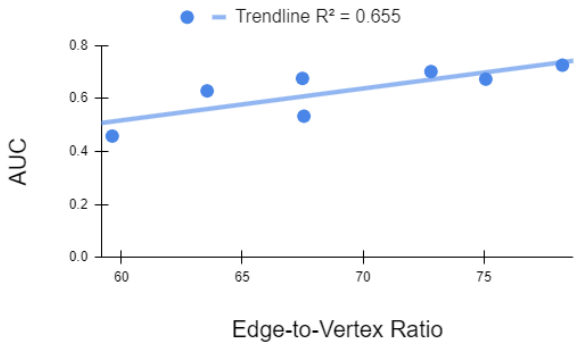
	Card + Dev + BA
Cards + Dev	7.35/7.84
Cards + BA	44.22/36.11
Dev + BA	9.75/3.43

Table 12. Percentage of improvement in the results when adding more features. This table compares the same graph with 43 edge features vs. 118 edge features. For each pair of values, the first number indicates the improvement over the F_1 score, and the second number represents the AUC improvement. It is noticeable that adding more features also helps to improve the metrics.

	Improvement(%)
Cards	60.33/35.25
Dev	125.29/6.08
BA	316.77/8.49
Cards + Dev	18.89/8.38
Cards + BA	59.11/25.68
Dev + BA	4.02/4.66
Cards + Dev + BA	8.46/1.66

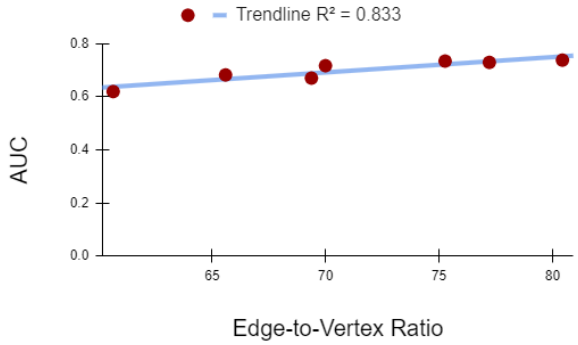
As a complement, plots in Figure 4a,b show the existing correlation between the AUC and the edge-to-vertex ratio, i.e., if the graph density increases, the AUC increases linearly. This is in accordance with results from Tables 5 and 8, where it can be observed that better metric results are obtained between the original graphs and the extended graphs. Therefore, for the fraud detection problem, it is helpful to increase the number of fraud-related features (such as transaction features or behavioral features) as well as the size and density of the graph. In addition, Table 9 confirms our result with another set of metrics that increasing the number of events (structural information) impacts the improvement of the metrics.

The results of this work are relevant since they indicate that fraudulent activity can be more effectively detected by incorporating structural information from different events (on online platforms) in a graph. In addition, the problem's temporality adds extra parameters that, together with the structural information, not only improve fraud detection but also help distinguish between fraudulent and legitimate behavior. In this work, several combinations of events were tested to determine if adding structural graph information contributed to improving detection metrics.



(**a**) Results for graphs with 43 features.

Figure 4. *Cont.*



(**b**) Results for graphs with 118 features.

Figure 4. AUC vs. edge-to-vertex ratio for the graphs with 43 edge features and 118 edge features. The considered features are related to user behaviors and transaction information in the online payment platform. It can be seen that if the graph density increases, the AUC increases linearly. The coefficient of determination R^2 is also reported. This also shows that aggregating more events (structural information) helps produce better metric results.

7. Conclusions

This study addressed the fraud detection problem by proposing an algorithm based on a Temporal Graph Neural Network (TGNN) to detect digital fraud using data from the Moneypool platform. The TGNN framework incorporates a series of interaction events representing users registering cards, devices, and bank accounts, resulting in three distinct types of edges. Using these data, seven different graph combinations were constructed; the first three represent individual interaction events (card registration, device registration, and bank account registration), as shown in Table 3. The remaining four graphs are formed from the initial graphs by the aggregations of card–device, card–bank account, device–bank account, and card–device–bank account registrations.

Class weighting was applied in the cross-entropy loss to address the class imbalance. The results were evaluated using the F_1 score and AUC metric. The values obtained for the F_1 score were moderate (Tables 5 and 8), suggesting that the class imbalance in the data might require additional considerations when using MPTGNN for this type of problem. This is likely because the F_1 score depends on the balance between true and false positives, and the data in question have a significantly high class imbalance. In addition, Table 9 shows another set of metrics that confirms the class imbalance's influence on the results, and at the same time, it can be noticed that graphs with more structural information improve relevant metrics as the recall.

Another factor that could influence the results for the F_1 score is the limited number of features used; this study only considered basic features related to transactions and user behavior on the platform. However, it was observed that incorporating structural and temporal context enhances fraud detection, as metrics improve with the addition of more interaction events. This is particularly evident for the AUC values, which improve when more types of interaction events are included, yielding better results than graphs with a single event type.

For completeness, another set of extended graphs was considered. These extended graphs contained more information about the number of features and events (Table 6). Experiments on these graphs also showed a better performance on metrics when adding more events, as well as when increasing the number of features.

For both graph configurations (original and extended), the edge-to-vertex ratio (density of the graph) was computed. There is a positive correlation between the AUC and the edge-to-vertex ratio, i.e., when the density of the graph increases, the AUC increases linearly. This suggests that aggregating more structural information is synonymous with achieving better metric results.

To the best of our knowledge, this is the first work to address the fraud detection problem using an ETG and a TGN-type algorithm that compares different sets of events in a graph constructed primarily with information from Latin America. In addition, since Latin America is the region with the highest fraud rate in the world [8], the results presented in this work can provide valuable insights into preventing this phenomenon in the area. Additionally, according to [10], most works (and research groups) on fraud prevention have used public and private data from entities outside Latin America.

It is important to note that while the algorithm considers three different types of edges (interaction events), it does not explicitly account for the graph's heterogeneity. As a result, although the method can handle various interaction events, it is restricted to processing bipartite graphs. Considering explicit heterogeneous aspects would involve additional considerations outside the current algorithmic approach. To address this limitation, explicit information about the edge type could be included as an additional feature. However, as outlined in this work, the current proposal (algorithm and ETG construction) produces relevant results.

Author Contributions: Formal analysis: D.S.-U. and G.D.I.L.; Research: D.S.-U. and G.D.I.L.; Software: D.S.-U.; Validation: G.D.I.L. and J.R.M.-R.; Writing review and editing: D.S.-U., G.D.I.L. and J.R.M.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: Data is unavailable because it is part of an ongoing study and due to ethical restrictions within fraud prevention research with real data.

Acknowledgments: We thank Moneypool for providing the data used in this work.

Conflicts of Interest: The authors declare no conflicts of interest. Although Diego Saldaña-Ulloa is affiliated with Moneypool, this affiliation did not influence the research's design, conduct, analysis, or reporting. Moneypool had no role in funding the research or shaping the study's outcomes. All the authors declare that the research was conducted without any commercial or financial relationship that could be constructed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GNN	Graph Neural Network
TGN	Temporal Graph Network
TGNN	Temporal Graph Neural Network
MPTGNN	Message-Passing Temporal Graph Neural Network
MLP	Multi-Layer Perceptron
GRU	Gated Recurrent Unit
AUC	Area Under Curve
ROC	Receiver Operating Characteristic

References

1. Caldeira, E.; Brandao, G.; Pereira, A.C.M. Fraud analysis and prevention in e-commerce transactions. In Proceedings of the 2014 9th Latin American Web Congress, Minas Gerais, Brazil, 22–24 October 2014.
2. Meng, C.; Zhou, L.; Liu, B. A Case Study in Credit Fraud Detection With SMOTE and XGBoost. *J. Phys. Conf. Ser.* **2020**, *1601*, 052016. [CrossRef]
3. Liu, Z.; Dou, Y.; Yu, P.S.; Deng, Y.; Peng, H. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, New York, NY, USA, 25–30 July 2020; pp. 1569–1572.
4. Nguyen, N.; Duong, T.; Chau, T.; Nguyen, V.; Trinh, T.; Tran, D.; Ho, T. A Proposed Model for Card Fraud Detection Based on CatBoost and Deep Neural Network. *IEEE Access* **2022**, *10*, 96852–96861. [CrossRef]
5. Cheng, D.; Wang, X.; Zhang, Y.; Zhang, L. Graph Neural Network for Fraud Detection via Spatial-Temporal Attention. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 3800–3813. [CrossRef]
6. Lucas, Y.; Jurgovsky, J. Credit card fraud detection using machine learning: A survey. *arXiv* **2020**, arXiv:2010.06479.

7. Rao, S.X.; Lanfranchi, C.; Zhang, S.; Han, Z.; Zhang, Z.; Min, W.; Cheng, M.; Shan, Y.; Zhao, Y.; Zhang, C. Modelling graph dynamics in fraud detection with "Attention". *arXiv* **2022**, arXiv:2204.10614.
8. Stripe. Lo Que Hay que Saber Sobre el Fraude en Línea. 2024. Available online: https://stripe.com/mx/guides/state-of-online-fraud#how-fraud-differs-by-region-country-and-company-size/ (accessed on 15 July 2024).
9. Wang, X.; Lyu, D.; Li, M.; Xia, Y.; Yang, Q.; Wang, X.; Wang, X.; Cui, P.; Yang, Y.; Sun, B.; et al. APAN: Asynchronous Propagation Attention Network for Real-Time Temporal Graph Embedding. In Proceedings of the 2021 International Conference on Management of Data, SIGMOD '21, New York, NY, USA, 15–18 December 2021; pp. 2628–2638.
10. Hove, D.; Olugbara, O.; Singh, A. Bibliometric Analysis of Recent Trends in Machine Learning for Online Credit Card Fraud Detection. *J. Scientometr. Res.* **2024**, *13*, 43–57. [CrossRef]
11. Derrow-Pinion, A.; She, J.; Wong, D.; Lange, O.; Hester, T.; Perez, L.; Nunkesser, M.; Lee, S.; Guo, X.; Wiltshire, B.; et al. ETA Prediction with Graph Neural Networks in Google Maps. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, New York, NY, USA, 1–5 November 2021; pp. 3767–3776.
12. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Red Hook, NY, USA, 4–9 December 2017; pp. 1025–1035.
13. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2018**, arXiv:1710.10903.
14. Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; Yu, P.S. Heterogeneous Graph Attention Network. In Proceedings of the The World Wide Web Conference, WWW '19, New York, NY, USA, 13–17 May 2019; pp. 2022–2032.
15. Manessi, F.; Rozza, A.; Manzo, M. Dynamic graph convolutional networks. *Pattern Recognit.* **2020**, *97*, 107000. [CrossRef]
16. Trivedi, R.; Dai, H.; Wang, Y.; Song, L. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In Proceedings of the 34th International Conference on Machine Learning, JMLR.org, ICML'17, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 3462–3471.
17. Trivedi, R.; Farajtabar, M.; Biswal, P.; Zha, H. DyRep: Learning Representations over Dynamic Graphs. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
18. Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; Yang, H. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In Proceedings of the 13th International Conference on Web Search and Data Mining, Online, 10–13 July 2020.
19. Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; Bronstein, M.M. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv* **2020**, arXiv:2006.10637.
20. Rao, S.X.; Zhang, S.; Han, Z.; Zhang, Z.; Min, W.; Cheng, M.; Shan, Y.; Zhao, Y.; Zhang, C. Suspicious Massive Registration Detection via Dynamic Heterogeneous Graph Neural Networks. *arXiv* **2020**, arXiv:2012.10831.
21. Lu, M.; Han, Z.; Rao, S.X.; Zhang, Z.; Zhao, Y.; Shan, Y.; Raghunathan, R.; Zhang, C.; Jiang, J. BRIGHT—Graph Neural Networks in Real-Time Fraud Detection. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM '22, New York, NY, USA, 17–22 October 2022; pp. 3342–3351.
22. Acevedo-Viloria, J.D.; Roa, L.; Adeshina, S.; Olazo, C.C.; Rodríguez-Rey, A.; Ramos, J.A.; Bahnsen, A.C. Relational Graph Neural Networks for Fraud Detection in a Super-App environment. *arXiv* **2021**, arXiv:2107.13673.
23. Ren, L.; Hu, R.; Li, D.; Liu, Y.; Wu, J.; Zang, Y.; Hu, W. Dynamic graph neural network-based fraud detectors against collaborative fraudsters. *Knowl.-Based Syst.* **2023**, *278*, 110888. [CrossRef]
24. Souza, A.H.; Mesquita, D.; Kaski, S.; Garg, V.K. Provably expressive temporal graph networks. *arXiv* **2022**, arXiv:2209.15059.
25. Longa, A.; Lachi, V.; Santin, G.; Bianchini, M.; Lepri, B.; Lio, P.; Scarselli, F.; Passerini, A. Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities. *arXiv* **2023**, arXiv:2302.01018.
26. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 1 July 2024).
27. Cui, Y.; Jia, M.; Lin, T.Y.; Song, Y.; Belongie, S. Class-Balanced Loss Based on Effective Number of Samples. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9260–9269.
28. Saporta, G.; Maraney, S. *Practical Fraud Prevention*, 1st ed.; OReilly: Sebastopol, CA, USA, 2022.
29. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
30. Xu, D.; Ruan, C.; Körpeoglu, E.; Kumar, S.; Achan, K. Inductive Representation Learning on Temporal Graphs. *arXiv* **2020**, arXiv:2002.07962.
31. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.U.; Polosukhin, I. Attention Is All You Need, *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2017; pp. 6000–6010.
32. Ravelin. Machine Learning for Fraud Detection. 2024. Available online: https://www.ravelin.com/insights/machine-learning-for-fraud-detection/ (accessed on 2 November 2024).
33. Seon. Fraud Detection with Machine Learning & AI. 2024. Available online: https://seon.io/resources/fraud-detection-with-machine-learning/ (accessed on 2 November 2024).