STGraph-FS: Structural Temporal Graph with Feature Selection for Ethereum Fraud Detection

Duong Ngoc Vu, Tuan-Cuong Vuong, Kim-Ngan Thi Nguyen, Tien-Cuong Nguyen, Vu-Duc Ngo, Trong-Nghia Nguyen, Mai Xuan Trang, Huan Vu, and Thien Van Luong

***Abstract*—Fraud detection in blockchain systems plays a critical role in maintaining the security and integrity of decentralized networks. Existing approaches often pay limited attention to feature selection, despite its pivotal importance in addressing the power-law distribution of account transaction frequencies observed in prior studies. To address these limitations, we present STGraph-FS, an efficient framework that leverages the complementary strengths of feature selection and directed temporal graph modeling. Our approach decomposes the detection process into specialized components: Spearman-Guided Directed Temporal Aggregation to capture evolving money flows with discriminative features, and structural clustering to partition accounts with similar behaviors. Experiments on the MulDiGraph dataset demonstrate that our model outperforms state-of-the-art baselines by 3%, validating the effectiveness and robustness of our modular design, especially in highly imbalanced conditions.**

***Index Terms*—Blockchain, Ethereum, Feature Selection, Fraud Detection, Imbalanced Data, Temporal Graph Networks.**

I. Introduction

In recent years, blockchain technology has gained significant attention due to its decentralized architecture and inherent immutability [1]. Ethereum [2] has emerged as a leading platform for decentralized finance applications, supporting widespread adoption of smart contract-based decentralized applications and digital assets [3, 4]. This rapid growth has also introduced escalating security and socioeconomic challenges [5, 6]. Malicious actors can exploit the pseudonymous nature of blockchain transactions to carry out fraudulent activities, posing serious risks to the integrity and stability of blockchain ecosystems. These challenges highlight the urgent need for effective and scalable fraud detection mechanisms within Ethereum and other blockchain platforms.

Recent advances in graph-based modeling have opened new opportunities for blockchain fraud detection by enabling more effective extraction of structural patterns within transaction networks [7, 8]. However, fraudulent accounts often interact with numerous legitimate ones, introducing topological noise that remains a key challenge. In this context, Tian *et al.* [9] have demonstrated the potential of graph-based frameworks in capturing temporal behavioral signals, achieving promising detection performance. However, the shortcomings of existing methods highlight the necessity for specialized frameworks leveraging structural and statistical features to enhance embedding quality.

In this paper, we introduce the STGraph-FS framework, a supervised, graph-based architecture for blockchain fraud detection that utilizes selected features to enhance model performance in both balanced scenarios and real-world imbalanced settings.

The primary contributions of this work are as follows:

- We introduce STGraph-FS, an effective architecture designed to capture temporal transaction dynamics and discriminative information in blockchain networks. STGraph-FS comprises two main modules:
  1) Spearman-Guided Directed Temporal Aggregation (SG-DTA): This module captures directed temporal transactions and incorporates discriminative information to enhance embedding quality.
  2) Soft Clustering: This component integrates cluster-level information into embeddings and optimizes the representations of nodes within the same cluster.
- We validate the effectiveness of STGraph-FS through comprehensive evaluations on benchmark datasets, demonstrating outstanding performance under both balanced and imbalanced data settings, surpassing baselines by about 3% and 5% F1-score gains, respectively.

The remainder of this paper is organized as follows. Section 2 reviews the related work on blockchain fraud detection. Section 3 presents the proposed STGraph-FS framework. Section 4 describes the experimental setup, presents the results, and provides an analysis. Finally, Section 5 concludes the paper and discusses potential future research directions.

II. Related Work

A. Feature Selection-Based Methods

Feature selection-based methods excel in reducing computational complexity and handling high-dimensional data, enabling rapid inference on large-scale datasets [10]. In [11], a machine learning framework named Eth-PSD was introduced, utilizing feature engineering to reduce dimensionality and eliminate noise for enhanced Ethereum phishing detection. Addressing Bitcoin ransomware, Wang *et al.* [12] proposed XRAD, which overcomes the information sparsity of single

D. N. Vu, H. Vu, K.-N. T. Nguyen, T.-N. Nguyen and T. V. Luong are with Business AI Lab, College of Technology, National Economics University, Hanoi, Vietnam (e-mail: duongvn.bai@st.neu.edu.vn, {huanv, nganguyen, nghiant, thienlv}@neu.edu.vn).

T.-C. Nguyen is with VNPT AI, VNPT Group, Hanoi, Vietnam (e-mail: nguyentiencuongn@vnpt.vn).

V.-D. Ngo is with MobiFone HighTech Center, MobiFone Corp., Hanoi, Vietnam (e-mail: duc.ngo@mobifone.vn).

T.-C. Vuong and M. X. Trang are with A2I Lab, School of Computing, Phenikaa University, Hanoi, Vietnam (e-mail: {trang.maixuan, cuong.vuongtuan}@phenikaa-uni.edu.vn).

Corresponding author: Thien Van Luong (e-mail: thienlv@neu.edu.vn).

addresses by employing a cascade feature extraction mechanism to aggregate and prioritize features from related transaction networks. Similarly, the Evolve Path Tracer developed in [13] focuses on the early detection of malicious addresses, extracting dynamic features from topological structures while filtering out low-value static information. In the context of blockchain-based IIoT networks, Block Hunter [14] was developed by combining federated learning with automated feature selection to accommodate resource-constrained edge devices. However, these methods often rely on manual feature engineering or domain-specific heuristics, which limits their ability to capture complex, latent interaction patterns.

B. Graph-Based Methods

Static Graph Neural Networks (GNNs) have been widely applied in fraud detection tasks due to their ability to capture structural information. Early adaptations, such as the work by Alarab *et al.* [7], leveraged Graph Convolutional Networks (GCNs) [8] to distinguish illicit patterns in Bitcoin anti-money laundering scenarios. Subsequently, feature aggregation was further refined in [15] through the introduction of Graph Attention Networks (GATs), which emphasize critical transactional relationships via differential weighting. Nevertheless, to address the evolving nature of blockchain activities, research has shifted toward temporal GNNs that explicitly capture the dynamic progression of transaction networks. Specific examples include PDTGA in [16], a model based on Temporal Graph Attention Networks (TGAT) that captures intricate interactions between timestamps and structural attributes. Similarly, Li *et al.* [17] proposed TTAGN to incorporate temporal signals directly into the graph learning process for Ethereum phishing detection. Furthermore, the evolution of account behavior was addressed in DiT-SGCR [9], which captures complex structural patterns alongside temporal evolution in Ethereum networks. Despite explicitly representing time and structure, existing temporal GNNs [9, 16, 17] often produce embeddings that lack sufficient discriminative power.

To address these shortcomings, we propose STGraph-FS, the effective architecture that enhances the original DiT-SGCR framework by seamlessly integrating structural and statistical features, as will be presented in the next section.

III. Proposed Method

Figure 1 presents the overall framework of our proposed STGraph-FS. To effectively capture temporal transaction dynamics and discriminative patterns in blockchain networks, the architecture integrates two key components: SG-DTA, which models directed interactions using discriminative information, and Soft Clustering, which incorporates cluster-level semantics to enhance representation robustness.

A. Graph Initialization

We construct the Ethereum transaction network as a directed temporal graph $G = (V, E, T)$, where V represents the set of all accounts, $E \subseteq V \times V \times \mathbb{R}$ captures the directed transactions annotated with timestamps, and $T = \{t_i \mid i = 1, \ldots, K\}$ lists all recorded timestamps. Each node $u \in V$ records a chronological sequence of interactions, with $N_{in}(u, t_i)$ and $N_{out}(u, t_i)$ denoting the sets of accounts that sent to or received from u at time t_i. The objective is to learn node embeddings $H \in \mathbb{R}^{|V| \times d}$ that encode both structural connections and temporal dynamics with discriminative power.

Transactions are encoded as directed edges, each carrying information about the timestamp and the amount of Ethereum moved, inherently representing the temporal and directional patterns in the Ethereum network. We preprocess raw data from CSV files into a dictionary structure, where each node maintains a time-ordered list of $(t_i, N_{in}(u, t_i), N_{out}(u, t_i))$ tuples. This arrangement allows for efficient temporal aggregation and supports subsequent embedding computation that captures transaction patterns over time, as will be shown below.

B. Spearman-Guided Directed Temporal Aggregation

To further enhance the representational capacity of the directed temporal aggregation module in DiT-SGCR [9], we propose the SG-DTA mechanism. This approach is inspired by concepts demonstrated in the work of Temporal SIR-GN [18]. Unlike traditional temporal aggregation methods that rely solely on timestamp-based features, SG-DTA incorporates a feature selection technique based on Spearman's rank correlation coefficient [19]. This process operates through following five sequential stages:

1) **Embedding Initialization:** Each node embedding is initialized with a soft cluster assignment, where components denote the probability that node u belongs to C clusters:
$$\mathbf{e}_u \in \mathbb{R}^C.$$

2) **Spearman-Guided Feature Selection:** Let each node $u \in V$ be associated with a feature vector:
$$\mathbf{f}_u = [f_u^1, f_u^2, \ldots, f_u^M] \in \mathbb{R}^M,$$
where f_u^m denotes the m-th feature of node u. Each node also has a corresponding label $y_u \in \{0, 1\}$ indicating whether it belongs to a fraudulent or normal class.
The Spearman's rank correlation coefficient [19] between feature f^m and label y is computed as:
$$\rho_m = 1 - \frac{6 \sum_{u=1}^{|V|} (r(f_u^m) - r(y_u))^2}{|V|(|V|^2 - 1)}, \tag{1}$$
where $r(f_u^m)$ and $r(y_u)$ represent the ranks of the feature value and the label of node u, respectively.
A feature f^m is utilized if its absolute value of its Spearman correlation coefficient [19] with the label exceeds a threshold τ:
$$|\rho_m| \geq \tau. \tag{2}$$
The resulting selected features set is denoted as:
$$\mathbf{f}_u^* = [f_u^1, f_u^2, \ldots, f_u^P], \quad P < M, \tag{3}$$

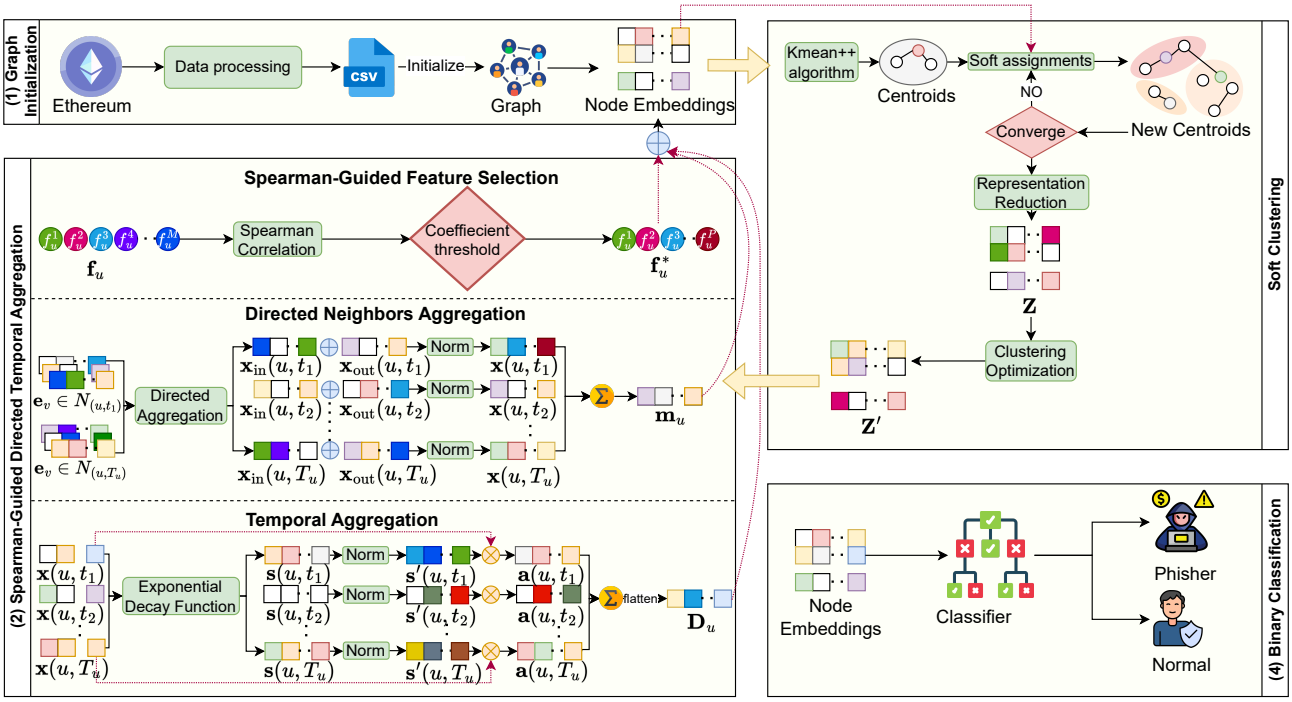

Fig. 1. The overall framework of the proposed STGraph-FS. The pipeline consists of three key phases: (1) **Graph Initialization**, where raw Ethereum data is processed to construct a directed transaction graph and initialize node embeddings; (2) **Spearman-Guided Directed Temporal Aggregation**, which filters discriminative features via Spearman correlation while simultaneously aggregating directed neighbor information and temporal dynamics; (3) **Soft Clustering**, which captures global structural patterns through representation reduction and clustering optimization to refine node representations; and (4) **Binary Classification**, which leverages the optimized embeddings to perform the final prediction task, distinguishing between phisher and normal accounts.

where P denotes the number of selected features that exhibit the strong monotonic relationship with the label, ensuring that the most discriminative features are retained for subsequent processing.

3) **Directed Neighbors Aggregation:** At each timestamp t_i, the directional neighbor representations are computed via a directed aggregation module, as illustrated in Fig. 1. These representations are formally derived as:

$$\mathbf{x}_{\text{in}}(u,t_i) = \sum_{v \in N_{\text{in}}(u,t_i)} \mathbf{e}_v, \quad \mathbf{x}_{\text{out}}(u,t_i) = \sum_{v \in N_{\text{out}}(u,t_i)} \mathbf{e}_v, \tag{4}$$

representing the aggregated transaction flows to and from node u, forming a temporally structured representation.

The timestamp-specific neighbor embedding is then obtained by normalizing the concatenated representations:

$$\mathbf{x}(u,t_i) = \frac{[\mathbf{x}_{\text{in}}(u,t_i),\, \mathbf{x}_{\text{out}}(u,t_i)]}{\left\|[\mathbf{x}_{\text{in}}(u,t_i),\, \mathbf{x}_{\text{out}}(u,t_i)]\right\|_2 + \epsilon}, \tag{5}$$

where ϵ is a small constant for numerical stability.

To capture the overall temporal behavior of node u, we aggregate the timestamp-specific embeddings across all time steps t_i:

$$\mathbf{m}_u = \sum_{i=1}^{T_u} \mathbf{x}(u,t_i), \tag{6}$$

where T_u denotes the total number of time steps associated with node u, and $\mathbf{m}_u \in \mathbb{R}^{2C}$ represents the temporal summary embedding of node u.

4) **Temporal Aggregation:** To track temporal evolution, we apply an exponential decay function that assigns higher weights to transactions occurring within shorter time intervals. We initialize the state vector as $\mathbf{s}(u,t_1) = \mathbf{0}$ and update subsequent states as follows:

$$\mathbf{s}(u,t_i) = \exp\left(-\frac{t_i - t_{i-1}}{\alpha}\right) \cdot \big(\mathbf{x}(u,t_{i-1}) + \mathbf{s}(u,t_{i-1})\big),$$
$$\mathbf{s}'(u,t_i) = \frac{\mathbf{s}(u,t_i)}{\|\mathbf{s}(u,t_i)\|_2 + \epsilon}, \tag{7}$$

where α is a decay factor controlling temporal sensitivity. Next, the temporal and structural embeddings are fused through an outer product operation:

$$\mathbf{a}(u,t_i) = \mathbf{x}(u,t_i) \otimes \mathbf{s}(u,t_i),$$
$$\mathbf{D}_u = \sum_{i=1}^{T_u} \mathbf{a}(u,t_i), \qquad \mathbf{D}_u \in \mathbb{R}^{C \times C}, \tag{8}$$

where we recall that C is the number of clusters.

5) **Final Embedding Generation:** As illustrated in Figure 1, the label-guided selected features $\mathbf{f}_u^*$ are concatenated with the temporal–structural representations derived in Eqs. (6) and (8) to form the final node embedding:

$$\mathbf{h}_u = [\mathrm{R}(\mathbf{D}_u),\, \mathbf{m}_u,\, \mathbf{f}_u^*], \quad \mathbf{h}_u \in \mathbb{R}^{4C^2+2C+P}, \tag{9}$$

where $\mathrm{R}(\cdot)$ denotes the flattening operation that reshapes the interaction matrix into a vector. Consequently, the comprehensive embedding matrix for all nodes is represented as $\mathbf{H} \in \mathbb{R}^{|V| \times (4C^2+2C+P)}$.

The detailed pseudocode for our proposed SG-DTA is presented in Algorithm 1.

Algorithm 1 SG-DTA: Spearman-Guided Directed Temporal Aggregation

Input: Graph $G = (V, E, T)$, node features $\mathbf{f}_u$, labels y_u, clusters C
Output: Node embeddings $\mathbf{H}$

Initialize $\mathbf{e}_u \in \mathbb{R}^C$ for all $u \in V$

for all $m \in \{1, \ldots, M\}$ **do** ▷ *Spearman-Guided Feature Selection*
  $\rho_m = \text{Spearman}(f^{(m)}, Y)$
  $\mathbf{f}_u^* = \{f_u^{(m)} \mid |\rho_m| \geq \tau\}$ for all $u \in V$
end for

for all $u \in V$ **do** ▷ *Directed Neighbors Aggregation*
  for $t_i \in T_u$ **do**
    $\mathbf{x}_{\text{in}}(u, t_i) = \sum_{v \in N_{\text{in}}(u,t_i)} \mathbf{e}_v$
    $\mathbf{x}_{\text{out}}(u, t_i) = \sum_{v \in N_{\text{out}}(u,t_i)} \mathbf{e}_v$
    $\mathbf{x}(u, t_i) = \frac{[\mathbf{x}_{\text{in}}, \mathbf{x}_{\text{out}}]}{\|[\mathbf{x}_{\text{in}}, \mathbf{x}_{\text{out}}]\|_2 + \epsilon}$
  end for
  $\mathbf{m}_u = \sum_{t_i \in T_u} \mathbf{x}(u, t_i)$
end for

for all $u \in V$ **do** ▷ *Temporal Aggregation*
  Initialize $\mathbf{s}(u, t_0) = \mathbf{0}, \mathbf{D}_u = \mathbf{0}$
  for $t_i \in T_u$ **do**
    $\mathbf{s}(u, t_i) = \text{ExpDecay}(\mathbf{x}(u, t_{i-1}), \mathbf{s}(u, t_{i-1}))$
    $\mathbf{D}_u \leftarrow \mathbf{D}_u + (\mathbf{x}(u, t_i) \otimes \mathbf{s}(u, t_i))$
  end for
end for

for all $u \in V$ **do** ▷ *Final Embedding Generation*
  $\mathbf{h}_u = [\mathrm{R}(\mathbf{D}_u), \mathbf{m}_u, \mathbf{f}_u^*]$
end for
$\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_{|V|}]$

C. Soft Clustering

Soft clustering integrates cluster-level information into node embeddings, enabling the representations to capture more comprehensive global structural patterns and thereby improving embedding quality. The final embeddings from the previous stage serve as the input for this process. The clustering procedure proceeds as follows:

1) **Centroid Initialization:** The K-means++ algorithm [20] is employed to initialize the cluster centroids $\boldsymbol{\mu}_c \in \mathbb{R}^{4C^2+2C+P}$, for $c = 1, \ldots, C$. To ensure the centroids are well-separated, K-means++ selects the first center at random and subsequent ones with a probability proportional to the squared distance to the nearest existing center. This probabilistic strategy effectively mitigates the risk of poor local optima and accelerates convergence during the training process.
2) **Soft Assignment:** The cosine similarity between node u and centroid $\boldsymbol{\mu}_c$ is computed as:

$$\text{sim}_{\cos}(u, c) = \frac{\mathbf{h}_u^\top \boldsymbol{\mu}_c}{\|\mathbf{h}_u\|_2 \, \|\boldsymbol{\mu}_c\|_2}, \tag{10}$$

The soft assignment coefficients are then obtained using a softmax function with an inverse temperature β:

$$w_{u,c} = \frac{\exp\big(\beta \cdot \text{sim}_{\cos}(u, c)\big)}{\sum_{c'=1}^{C} \exp\big(\beta \cdot \text{sim}_{\cos}(u, c')\big)}. \tag{11}$$

3) **Centroid Update:** The centroids are iteratively refined as the weighted mean of the node embeddings:

$$\boldsymbol{\mu}_c = \frac{\sum_{u \in V} w_{u,c} \mathbf{h}_u}{\sum_{u \in V} w_{u,c} + \epsilon}, \tag{12}$$

and subsequently normalized to have unit length.
4) **Representation Reduction:** A compact embedding matrix $\mathbf{Z} \in \mathbb{R}^{|V| \times C}$ is constructed based on the cosine distances to the cluster centroids:

$$\text{dist}_{u,c} = 1 - \text{sim}_{\cos}(u, c). \tag{13}$$

For each node u, the cosine distances to the cluster centroids are normalized to the $[0, 1]$ range using min-max scaling:

$$\bar{z}_{u,c} = \frac{\max_c(\text{dist}_{u,c}) - \text{dist}_{u,c}}{\max_c(\text{dist}_{u,c}) - \min_c(\text{dist}_{u,c}) + \epsilon}. \tag{14}$$

The normalized embeddings are further adjusted to form a probability-like distribution:

$$z_{u,c} = \frac{\bar{z}_{u,c}}{\sum_{c'=1}^{C} \bar{z}_{u,c'} + \epsilon}. \tag{15}$$

The resulting matrix $\mathbf{Z} \in \mathbb{R}^{|V| \times C}$, where each element $z_{u,c}$ denotes the normalized probability of node u belonging to cluster c, provides a low-dimensional structural representation that captures the global topology relative to the cluster centroids.
5) **Clustering Optimization:** To refine the low-dimensional structural embeddings, clustering optimization is employed to enhance both cluster coherence and global structural consistency. Given the initial embeddings $\mathbf{Z} \in \mathbb{R}^{|V| \times C}$, the optimization objective is defined as:

$$\min_{\mathbf{Z}'} \ \text{tr}(\mathbf{Z}'^\top \mathbf{L} \mathbf{Z}') + \gamma \sum_{c=1}^{C} \text{tr}(\mathbf{Z}'^\top \mathbf{L}_c \mathbf{Z}') + \phi \|\mathbf{Z}' - \mathbf{Q}\|_F^2, \tag{16}$$

where $\mathbf{Z}' \in \mathbb{R}^{|V| \times C}$ denotes the optimized embedding matrix, and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ represents the graph Laplacian, constructed from the degree matrix $\mathbf{D}$ and the adjacency matrix $\mathbf{A}$, $\mathbf{Q} = \phi \cdot \mathbf{Z}$ represents the initial embedding term, and γ, ϕ represent hyperparameters controlling the balance between cluster coherence and embedding fidelity. The optimization is solved via the conjugate gradient descent method:

$$(\mathbf{L} + \gamma \sum_{c=1}^{C} \mathbf{L}_c + \phi \mathbf{I}) \mathbf{Z}' = \phi \mathbf{Q}, \tag{17}$$

yielding optimized embeddings $\mathbf{Z}'$ that effectively capture both local clustering patterns and global topology. The node embeddings are iteratively updated through SG-DTA and clustering to continuously capture temporal dynamics and structural roles.

D. Binary Classification

Based on our preliminary experiments, XGBoost [21] demonstrates superior performance compared to other classifiers, as detailed in Table V in Subsection IV-C. Therefore, we employ the XGBoost [21] classifier in this study to ensure robust detection capabilities. The dataset is split into training (80%) and testing (20%) sets, where the classifier is trained to distinguish normal accounts from malicious ones. The primary evaluation focuses on Precision, Recall, and F1-Score of the phisher class to assess the effectiveness of the classification results.

IV. Experiments

In this section, we conduct a comprehensive experimental evaluation of the proposed STGraph-FS framework on the real-world MulDiGraph dataset to assess its effectiveness and robustness. STGraph-FS is designed as a supervised, graph-based framework that integrates Spearman-guided directed temporal aggregation and soft clustering. For comparison, we include several state-of-the-art (SOTA) baseline methods.

A. Experimental Setup

We describe the experimental setup used for evaluating STGraph-FS as follows:

- **Evaluation Dataset:** We utilize the MulDiGraph dataset to analyze Ethereum-based transaction behaviors and phishing activities. The dataset, publicly available on the XBlock platform [22], was released in December 2020 and represents a large-scale Ethereum transaction network constructed via a two-hop Breadth-First Search (BFS) from verified phishing nodes. A summary of the dataset statistics under two conditions is provided in Table I.

TABLE I
SUMMARY OF MULDIGRAPH DATASET UNDER DIFFERENT FRAUD RATIOS

Fraud Ratio	Attribute	MulDiGraph
5:5	Nodes	10,960
	Edges	2,933,773
	Phishing Accounts	5,480
1:9	Nodes	54,800
	Edges	3,654,308
	Phishing Accounts	5,480

- **Evaluation Metrics:** We use Precision, Recall, and F1-score to assess model performance. The best results are marked in bold, and the second-best are underlined.
- **Implementation Details:** All experiments were re-implemented on our dataset. To ensure the integrity of the evaluation and prevent data leakage, the Spearman-Guided Feature Selection module was fitted exclusively on the training set; the selected features were subsequently applied to the test sets. They were executed on an NVIDIA RTX 4090 GPU with 24 GB of memory.
- **Baselines:** We compare STGraph-FS against state-of-the-art models, including Dynamic Feature [23], TLMG4Eth [24], GCN [8], GraphSAGE [25], GAT [15], and DiT-SGCR [9]. The parameters for Dynamic Feature [23], TLMG4Eth [24], and DiT-SGCR [9] are set according to their open-source implementations. GCN [8] utilizes a 2-layer architecture (64 hidden units, learning rate 1×10^{-5}). In addition, GraphSAGE [25] is implemented with 3 layers using DGL, mean aggregation, ReLU, and dropout. Similarly, GAT [15] employs multi-head attention, concatenating heads in hidden layers and averaging them for the final output.

B. Comparison with Baseline Method

TABLE II
PERFORMANCE COMPARISON BETWEEN PROPOSED METHOD AND BASELINE METHODS UNDER DIFFERENT FRAUD RATIOS

Method	5:5			1:9		
	P	R	F1	P	R	F1
GCN [8]	50.08	57.06	53.34	12.01	74.69	20.69
GAT [15]	45.80	51.01	48.26	15.50	62.03	24.80
SAGE [25]	50.08	59.82	54.52	10.69	35.47	16.43
TLM4Eth [24]	91.26	91.93	91.59	55.02	83.96	66.48
Dynamic Feature [23]	88.46	97.06	92.56	58.06	**90.55**	70.75
DiT-SGCR [9]	89.15	97.85	93.73	84.25	76.44	80.16
Our STGraph-FS	**93.84**	**98.30**	**96.02**	**85.16**	86.14	**85.65**

Table II presents the performance of STGraph-FS, configured with the optimal feature set and the best-performing classifier as determined in Table III, Table IV and Table V respectively, compared to the baseline methods. STGraph-FS achieves an F1-score of 96.02% on balanced data (5:5) and 85.65% on highly imbalanced data (1:9). This is expected due to the fact that performance on balanced data is generally higher. Among the baselines, GCN, GAT, and SAGE perform poorly in both scenarios, indicating its limitations in capturing complex temporal and graph patterns. Dynamic Feature shows strong results on balanced data (92.56%) but drops significantly under imbalanced conditions (70.75%), revealing its sensitivity to label imbalance. TLM4Eth attains a high F1 score on balanced data (91.59%) but performs poorly on the 1:9 ratio (66.48%), suggesting difficulty in detecting rare fraud instances.

Two key insights emerge from these results. First, STGraph-FS consistently outperforms all existing models, especially on imbalanced datasets. This demonstrates its practical effectiveness. Second, integrating the proposed feature selection into temporal embeddings enhances the model's clustering ability. Better clustering produces more optimized embeddings, which improves the separation between fraudulent and normal accounts and ultimately boosts performance, particularly in highly imbalanced scenarios.

C. Ablation Study

Table III and Table IV present a detailed analysis of feature effectiveness, focusing on performance comparison under different fraud ratios and feature correlation ranking, respectively. As detailed in Table III, structural and temporal features such as Node In-degree (0.738) and Std. Hour Received (0.624) exhibit the highest correlation with fraud labels. Consistent with this ranking, the results in Table IV show that the model's

TABLE III
RANKING OF TOP 10 SELECTED FEATURES BASED ON CORRELATION ANALYSIS

Rank	Feature Name	Correlation Score
1	Node In-degree	0.738
2	Direction Ratio	0.679
3	Std. Hour Received	0.624
4	Mean Hour Received	0.578
5	Min Time Between Tx	0.532
6	Weekday Rx Ratio	0.457
7	Account Balance	0.359
8	Avg. Outgoing Amount	0.344
9	Max Outgoing Amount	0.344
10	Min Outgoing Amount	0.344

TABLE IV
PERFORMANCE COMPARISON UNDER DIFFERENT FRAUD RATIOS WITH DIFFERENT NUMBERS OF SELECTED FEATURES

Selected Features	5:5			1:9		
	P	R	F1	P	R	F1
6	93.48	97.27	95.34	84.10	84.33	84.21
8	**94.00**	97.36	95.65	84.81	85.98	85.39
10	93.84	**98.30**	**96.02**	85.16	**86.14**	**85.65**
Full	93.01	97.74	95.32	**86.12**	83.04	84.55

performance consistently improves as the number of selected features increases up to ten, demonstrating the effectiveness of the proposed feature selection strategy. In the balanced 5:5 ratio, the model achieves an F1-score of 96.02%, which is superior to using fewer features or all features. Similarly, under the imbalanced 1:9 ratio, the F1-score reaches 85.65%, representing the highest performance across cases. Notably, using the full feature set slightly reduces accuracy, suggesting that redundant or noisy features (those with lower correlation rankings) can hinder generalization.

These findings confirm that selecting an optimal subset of features enhances temporal and structural representation learning, enabling more robust and accurate fraud detection. The top ten performing features provide the best balance between Precision and Recall, highlighting the importance of feature selection in improving overall model effectiveness under both balanced (5:5) and imbalanced (1:9) settings. Therefore, we set $K = 10$ for all subsequent experiments.

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT CLASSIFIERS UNDER DIFFERENT FRAUD RATIOS

Classifier	5:5			1:9		
	P	R	F1	P	R	F1
Random Forest [26]	92.97	**98.55**	95.68	84.66	84.20	84.43
KNN [27]	81.90	93.64	87.37	65.94	51.82	58.03
MLP [28]	87.82	96.24	91.83	77.06	69.69	73.19
XGBoost [21]	**93.84**	98.30	**96.02**	**85.16**	**86.14**	**85.65**

Table V presents a detailed performance comparison under different fraud ratios with various classification algorithms. The results show that the XGBoost model's performance consistently outperforms other baselines, demonstrating the effectiveness of the gradient boosting strategy. In the balanced 5:5 ratio, the model achieves an F1-score of 96.02, better than Random Forest, MLP, or KNN. Similarly, under the imbalanced 1:9 ratio, the F1-score reaches 85.65, representing the highest performance across cases. Notably, distance-based methods like KNN show a significant reduction in accuracy under the imbalanced setting, suggesting that the sparsity of the minority class can hinder generalization in such models.

These findings confirm that employing robust ensemble learning techniques enhances the model's ability to handle class imbalance, enabling more reliable and accurate fraud detection. The XGBoost classifier provides the best balance between Precision and Recall, highlighting the importance of advanced boosting algorithms in improving overall model effectiveness under both balanced (5:5) and imbalanced (1:9) settings. Therefore, we select XGBoost as the primary classifier for all subsequent experiments.

TABLE VI
PERFORMANCE COMPARISON OF PROPOSED MODEL VARIANTS UNDER DIFFERENT FRAUD RATIOS

Method	5:5			1:9		
	P	R	F1	P	R	F1
w/o Feature Selection	93.38	96.88	95.10	84.00	81.19	82.57
w/o Neighbors	89.33	91.61	90.46	83.90	83.72	83.82
w/o Temporal	92.92	96.90	94.87	82.67	81.46	82.06
w/o Clustering Optimization	**93.95**	97.72	95.80	83.48	84.47	83.98
Our STGraph-FS	93.84	**98.30**	**96.02**	**85.16**	**86.14**	**85.65**

Table VI presents the ablation study results addressing RQ4, which investigates the individual contribution of each component within the proposed STGraph-FS framework. We systematically evaluate the model's performance by removing key modules: Feature Selection, Neighbors, Temporal, and Clustering Optimization, under both balanced (5:5) and imbalanced (1:9) fraud ratio settings.

Excluding Feature Selection decreases the F1-score to 95.10% (5:5) and 82.57% (1:9), highlighting its critical role in filtering redundant or noisy inputs. While the Neighbors module significantly impacts performance in the balanced setting-with the F1-score dropping to 90.46%-its removal results in a more moderate decline to 83.82% under the imbalanced setting. Conversely, eliminating the Temporal module lowers the F1-score to 94.87% (5:5) and notably to 82.06% (1:9), confirming the vital importance of modeling sequential transaction patterns, particularly when detecting fraud in imbalanced scenarios. Meanwhile, removing the Clustering Optimization module leads to reductions to 95.80% and 83.98%, reflecting its utility in stabilizing and smoothing embeddings. Overall, the Full Model achieves superior performance across all settings.

These results yield two key insights. First, the Full Model consistently attains the highest metrics, confirming that the integration of all components is essential for optimal performance. Second, both Feature Selection and Temporal modeling play central roles; Feature Selection enhances the quality of node embeddings by removing noise, while the Temporal module captures dynamic fraud patterns that are crucial for handling class imbalance. Therefore, the proposed approach effectively improves the model's overall discriminative power, demonstrating robustness under highly imbalanced conditions.

V. CONCLUSIONS

We present STGraph-FS, a supervised, graph-based framework for Ethereum fraud detection that leverages the complementary strengths of optimal feature selection and directed

temporal graph modeling. Specifically, the SG-DTA module employs a Spearman-guided mechanism to identify and prioritize discriminative features, mitigating noise in high-dimensional transaction data. This enables the clustering module to seamlessly incorporate structural information into the embeddings. Moreover, the clustering optimization promotes tighter intra-cluster compactness, thereby significantly boosting the model's overall performance. Finally, experimental validation on the real-world MulDiGraph dataset demonstrates that our framework outperforms state-of-the-art baselines, confirming the effectiveness of our architectural approach in robustly handling heavily imbalanced data distributions. Future research will extend this framework to real-time fraud detection systems and explore its applicability to cross-chain transaction networks, where complex inter-chain relationships present greater security challenges.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018.

[2] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[3] K. Wüst and A. Gervais, "Do you need a blockchain?" in *Proc. IEEE Crypto Valley Conf. Blockchain Technol. (CVCBT)*. IEEE, 2018, pp. 45–54.

[4] N. Deepa, Q.-V. Pham, D. C. Nguyen, S. Bhattacharya, B. Prabadevi *et al.*, "A survey on blockchain for big data: Approaches, opportunities, and future directions," *Future Gener. Comput. Syst.*, vol. 131, pp. 209–226, 2022.

[5] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on Ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–43, 2020.

[6] Z. Wang, H. Jin, W. Dai, K.-K. R. Choo, and D. Zou, "Ethereum smart contract security research: Survey and future research opportunities," *Frontiers Comput. Sci.*, vol. 15, no. 2, p. 152802, 2021.

[7] I. Alarab, S. Prakoonwit, and M. I. Nacer, "Competence of graph convolutional networks for anti-money laundering in Bitcoin blockchain," in *Proc. 5th Int. Conf. Mach. Learn. Technol. (ICMLT)*, 2020, pp. 23–27.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[9] Y. Tian, L. Song, P. Qian, Y. Wang, J. Sun, and Y. Jia, "DiT-SGCR: Directed temporal structural representation with global-cluster awareness for Ethereum malicious account detection," *arXiv preprint arXiv:2506.20123*, 2025.

[10] V.-D. Ngo, T.-C. Vuong, T. Van Luong, and H. Tran, "Machine learning-based intrusion detection: Feature selection versus feature extraction," *Cluster Comput.*, vol. 27, no. 3, pp. 2365–2379, 2024.

[11] A. H. H. Kabla, M. Anbar, S. Manickam, and S. Karupayah, "Eth-PSD: A machine learning-based phishing scam detection approach in Ethereum," *IEEE Access*, vol. 10, pp. 118 043–118 057, 2022.

[12] K. Wang, M. Tong, J. Pang, J. Wang, and W. Han, "XRAD: Ransomware address detection method based on Bitcoin transaction relationships," *ACM Trans. Web*, vol. 18, no. 4, pp. 1–33, 2024.

[13] L. Cheng, F. Zhu, Y. Wang, R. Liang, and H. Liu, "Evolve path tracer: Early detection of malicious addresses in cryptocurrency," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2023, pp. 3889–3900.

[14] A. Yazdinejad, A. Dehghantanha, R. M. Parizi, M. Hammoudeh, H. Karimipour, and G. Srivastava, "Block hunter: Federated learning for cyber threat hunting in blockchain-based IIoT networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8356–8366, 2022.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[16] L. Wang, M. Xu, and H. Cheng, "Phishing scams detection via temporal graph attention network in Ethereum," *Inf. Process. Manage.*, vol. 60, no. 4, p. 103412, 2023.

[17] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong, "TTAGN: Temporal transaction aggregation graph network for Ethereum phishing scams detection," in *Proc. ACM Web Conf. (WWW)*, 2022, pp. 661–669.

[18] J. Layne, J. Carpenter, E. Serra, and F. Gullo, "Temporal SIR-GN: Efficient and effective structural representation learning for temporal graphs," *Proc. VLDB Endow.*, vol. 16, no. 9, pp. 2075–2089, 2023.

[19] P. Sedgwick, "Spearman's rank correlation coefficient," *BMJ*, vol. 349, 2014.

[20] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. 18th ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2007, pp. 1027–1035.

[21] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[22] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "XBlock blockchain datasets: InPlusLab Ethereum phishing detection datasets," https://xblock.pro/, 2019.

[23] Z. Sheng, L. Song, and Y. Wang, "Dynamic feature fusion: Combining global graph structures and local semantics for blockchain phishing detection," *IEEE Trans. Netw. Service Manag.*, 2025.

[24] J. Sun, Y. Jia, Y. Wang, Y. Tian, and S. Zhang, "Ethereum fraud detection via joint transaction language model and graph representation learning," *Inf. Fusion*, vol. 120, p. 103074, 2025.

[25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017.

[26] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45,

no. 1, pp. 5–32, 2001.

[27] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for knn classification," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 3, pp. 1–19, 2017.

[28] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Trans. Circuits Syst.*, vol. 8, no. 7, pp. 579–588, 2009.