STAGS: A Graph-Sampling Approach for GNN-based Network Anomaly Detection

Saikat Dey
Dept. of Computer Science
Virginia Tech
dsaikat@vt.edu

Mark Gardner
Dept. of IT
Virginia Tech
mkg@vt.edu

Jeff Lang
Dept. of IT
Virginia Tech
jefflang@vt.edu

Wu-chun Feng
Dept. of Computer Science
Virginia Tech
feng@cs.vt.edu

***Abstract*—Graph neural networks (GNNs) are promising for network anomaly detection but face scalability challenges when applied to real-world network graphs. These graphs are typically large, dynamic, and attributed, leading to excessive memory consumption and training time when processed in full. While sampling can mitigate scalability issues, graph sampling is inherently difficult due to the need to capture the complex structural connectivity in large-scale dynamic attributed graphs. Moreover, ensuring scalability without sacrificing the accuracy of GNN-based network anomaly detection requires a sampling approach that identifies and preserves graph regions that are critical for anomaly detection. In large-scale dynamic attributed network graphs, this demands integrating not only structural connectivity but also temporal dynamics and attributes into the sampling process.**

To address these challenges, we propose STAGS (short for Structural, Temporal, and Attribute-aware Graph Sampling), a novel sampling method that improves both the scalability and the accuracy of GNN-based network anomaly detection. STAGS preserves network regions that show significant temporal spectral entropy changes (a strong indicator of anomalies) into the sampled subgraph. Extensive experiments on four state-of-the-art GNN-based anomaly detection models, using 24 synthetic and 2 real-world network graphs, show that STAGS improves scalability by accelerating training time by a factor of 2.44× (compared to training on the full graph) and improves accuracy (AUC-ROC) by up to 53%.

***Index Terms*—graph sampling, graph neural network, network graphs, anomaly detection, spectral entropy**

I. Introduction

With the rapid growth of big data, real-world network graphs are increasingly large, time-evolving, and rich in information, i.e., large-scale, dynamic, and attributed, respectively. Though these properties enhance their utility, particularly in deep learning applications like graph neural network (GNN)-based network anomaly detection, they also introduce scalability challenges [1]. Training GNNs on a full graph leads to high memory consumption and prolonged training times.

In machine learning, sampling methods mitigate these scalability issues by allowing models to train on a representative subset rather than the full dataset. However, in graph-based learning, selecting a truly representative subgraph from a graph is inherently challenging due to the complex structural connectivity imposed by vertex interdependencies. This challenge becomes even more pronounced in large-scale dynamic attributed graphs, where an effective graph sampling strategy must preserve not only structural patterns but also temporal dynamics and attribute-driven variations to ensure accurate representativeness in the sampled subgraph.

Existing graph-sampling methods, e.g., [2], [3], focus on structural pattern preservation but overlook temporal and attribute-driven patterns. This oversight not only limits representativeness but also results in the failure to identify and preserve the most "informative" regions of the graph in the sampled subgraph. — those that are critical for downstream tasks. Consequently, while these methods improve scalability, they do so at the cost of degrading the accuracy of downstream tasks, such as GNN-based network anomaly detection.

Real-world network graphs consist of vertices that represent diverse system entities (e.g., DNS, routers, and IoT devices) and edges that capture the interactions between these system entities. Anomalies in these real-world network graphs manifest not just via structural deviations but also via significant temporal shifts and attribute fluctuations. Prior studies [4], [5] show that considering only one aspect of the graph (i.e., structural, temporal, or attribute) leads to high false positives in anomaly detection. Thus, an effective sampling strategy must holistically integrate all three aspects — structural, temporal, and attribute awareness — to ensure that the sampled subgraph retains critical anomaly-relevant regions, and, in turn, improve both the scalability and accuracy of GNN-based network anomaly detection.

Our approach to integrate structural, temporal, and attribute awareness is called STAGS (Structural, Temporal, and Attribute-aware Graph Sampling), a

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract W911NF-18-C-0019 with the U.S. Army Contracting Command - Aberdeen Proving Ground (ACC-APG) under grant 2016431. Resources provided by Advanced Research Computing (ARC) at Virginia Tech and insights from Jack Davidson, Alastair Nottingham, and Jason Hiser (University of Virginia) are also gratefully acknowledged.

ISBN 978-3-903176-72-0 © 2025 IFIP

novel sampling method that improves both the scalability and accuracy of GNN-based network anomaly detection. STAGS strategically identifies and samples the most "informative" regions of a network graph by tracking significant *temporal* changes in spectral entropy — a strong indicator of anomalies. Our spectral entropy formulation itself incorporates both the *structural* and *attribute* information of the graph. Thus, STAGS offers two primary advantages over existing graph-sampling methods: (1) explicit incorporation of structural, temporal, and attribute-driven characteristics of the graph and (2) selective sampling of high-interest regions of the graph where anomalies are more likely to occur, improving both model scalability and accuracy.

We rigorously evaluate STAGS across four state-of-the-art GNN-based anomaly detection models using 24 synthetic and 2 real-world network graphs. Our results show that STAGS significantly improves GNN-based network anomaly detection with respect to scalability and accuracy. For scalability, STAGS accelerates training time by 2.44× when compared to training on the full graph. For accuracy, STAGS improves the AUC-ROC score by up to 53% and reduces the "prediction lag" (i.e., the time delay between an anomaly's occurrence and detection) by up to 42%.

We outline our contributions as follows:

- **STAGS** (Structural, Temporal, and Attribute-aware Graph Sampling), a novel sampling method that improves scalability and accuracy of GNN-based network anomaly detection.
- A rigorous evaluation of **STAGS** across four state-of-the-art GNN-based anomaly detection models using 24 synthetic network graphs (with up to 400M edges) and 2 real-world network graphs (with up to 12B edges).
- A new metric, *prediction lag*, that quantifies the time delay between an anomaly's occurrence and its detection. We show that STAGS significantly reduces prediction lag — a critical improvement for real-world network anomaly detection where delayed responses can have severe consequences.

II. Related Work

Anomaly detection in graphs has received considerable attention across domains such as cybersecurity [6], social media [7] and healthcare [8]. Graph neural networks (GNNs) [9]–[11] have become widely used for this task; however, applying GNNs to real-world graphs that are large-scale, dynamic, and attributed presents significant scalability challenges, such as high memory requirements and extended training times [1].

To mitigate these scalability challenges, existing graph sampling methods, such as ClusterGCN [12] and GraphSAINT [13], focus on generating multiple subgraphs (or mini-batches) that collectively cover the original graph, enabling memory-efficient GNN training. However, these approaches mainly serve as effective graph partitioning strategies *during* training rather than being tailored to identify a representative subgraph *before* GNN training. Selecting a representative subgraph before training is critical for anomaly detection, as it allows models to focus on the most "informative" regions of the graph, where anomalies are more likely to occur.

To find a representative subgraph, current state-of-the-art can be categorized into two types: (1) graph sampling and (2) graph sparsification/coarsening.

1) Graph Sampling: Graph-sampling methods aim to extract a representative subgraph while preserving essential structural properties. These methods can be broadly classified into vertex sampling [2], edge sampling [14], and traversal-based sampling [15]. However, these approaches are mainly designed for applications like community detection [3].

2) Graph Sparsification/Coarsening: Graph sparsification and coarsening methods aim to reduce the graph size while preserving its spectral or structural properties. Unlike sampling methods that select subsets of vertices or edges, these methods [16], [17] modify the graph's connectivity to enhance computational efficiency.

All of the aforementioned methods share a common limitation: they fail to prioritize high-interest regions in large-scale dynamic attributed graphs, where anomalous activity typically occurs.

III. Methodology

Figure 1 captures the workflow of STAGS (Structural, Temporal, and Attribute-aware Graph Sampling), our proposed sampling method for large-scale dynamic attributed network graphs. STAGS operates in three key phases: (1) spectral entropy (SE) calculation, (2) vertex selection based on the STAGS score (SS), and (3) neighborhood expansion of the selected vertices.

We first formalize the concept of dynamic attributed graphs before detailing each phase of STAGS.

A. Dynamic Attributed Graphs

A dynamic attributed graph is defined as:

$$G(t) = \{V(t), E(t), X_v(t), X_e(t)\}$$

where

- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices, with $V(t) \subseteq V$ representing the vertices at time t.
- $E(t) \subseteq V(t) \times V(t)$ represents the set of edges at time t.
- $X_v(t) \in \mathbb{R}^{|V(t)| \times d_v}$ is the matrix of vertex attributes at time t, where d_v is the dimension of the vertex attribute vector.

- $X_e(t) \in \mathbb{R}^{|E(t)| \times d_e}$ is the matrix of edge attributes at time t, where d_e is the dimension of the edge attribute vector.

We adopt a *graph snapshot* approach where all vertices and edges within a fixed time window are treated as part of the same snapshot. Formally, a dynamic attributed graph is represented as:

$$G = \{G(0), G(1), .., G(t), ...G(T)\}$$

where $G(t)$ is the graph snapshot at time t, and T is the total number of time windows.

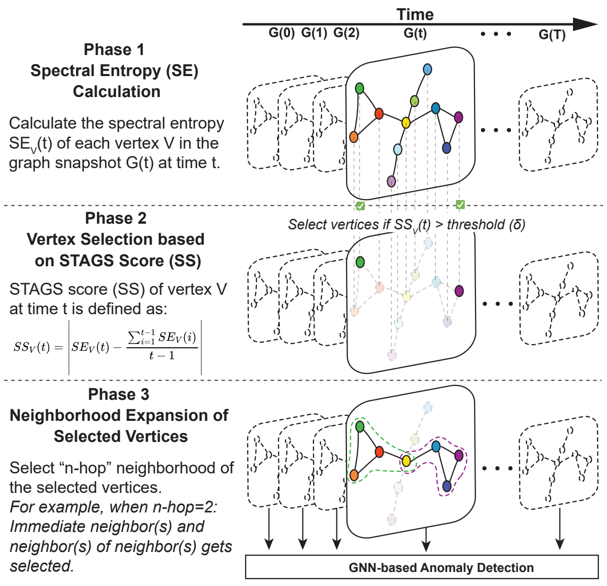

Time
G(0) G(1) G(2)
G(t)
G(T)
Phase 1
Spectral Entropy (SE) Calculation
Calculate the spectral entropy SE$_V$(t) of each vertex V in the graph snapshot G(t) at time t.
Phase 2
Vertex Selection based on STAGS Score (SS)
Select vertices if SS$_V$(t) > threshold (δ)
STAGS score (SS) of vertex V at time t is defined as:
$SS_V(t) = \left| SE_V(t) - \frac{\sum_{i=1}^{t-1} SE_V(i)}{t-1} \right|$
Phase 3
Neighborhood Expansion of Selected Vertices
Select "n-hop" neighborhood of the selected vertices.
For example, when n-hop=2: Immediate neighbor(s) and neighbor(s) of neighbor(s) gets selected.
GNN-based Anomaly Detection

Fig. 1: STAGS workflow

B. STAGS Workflow

Our STAGS method identifies and samples high-interest regions in large-scale dynamic attributed network graphs, capturing structural, temporal, and attribute-driven variations to improve both scalability and accuracy of GNN-based network anomaly detection. STAGS' approach consists of three main phases that are detailed below:

1) ***Phase 1: Spectral Entropy (SE) Calculation****:* In this phase, we compute the spectral entropy of each vertex within a given graph snapshot and track its historical mean spectral entropy over time. Our spectral entropy formulation incorporates both the vertex and edge attributes. Thus, we ultimately capture structural, temporal, and attribute information of the graph via spectral entropy. Deviations in spectral entropy signal anomalous behavior.

A vertex is considered *informative* (potentially anomalous) if its spectral entropy at time t deviates significantly from its historical mean across previous snapshots. This deviation is quantified in Phase 2 using the STAGS Score (SS), which guides the selection of these informative vertices for sampling. In Phase 3, the sampling region is further expanded by including neighboring vertices of the selected vertices from Phase 2, ultimately providing us informative (potentially anomalous) graph regions.

To compute the spectral entropy while considering both vertex and edge attributes, we enhance the adjacency matrix as follows:

$$\hat{A} = A + W_v S_v + W_e S_e,$$

where S_v is the vertex similarity matrix, capturing attribute similarity between vertices, and S_e is the edge influence matrix, incorporating edge attributes into the graph structure. The weighting factors W_v and W_e balance the contributions of vertex and edge attributes.

The similarity between two vertices i and j based on their attributes is computed using the Gaussian kernel:

$$S_v(i,j) = \exp\left(-\frac{||X_v^i - X_v^j||^2}{\sigma_v^2}\right)$$

Similarly, the influence of edge attributes is given by

$$S_e(i,j) = A_{ij} \exp\left(-\frac{||X_e^{ij}||^2}{\sigma_e^2}\right)$$

where σ_v^2 and σ_e^2 are scaling parameters that control the sensitivity of the Gaussian kernel for vertex and edge attributes, respectively.

Using this enhanced adjacency matrix, we compute the graph Laplacian:

$$L = D - \hat{A}$$

where D is the degree matrix. The normalized Laplacian is given by:

$$L = D^{-1/2} L D^{-1/2}$$

We compute the eigenvalues (λ) and eigenvectors ($\mathbf{v}$) of L, and derive the probability distribution for each vertex:

$$p_{ij} = \frac{(\mathbf{v}_{ij})^2}{\sum_k (\mathbf{v}_{ik})^2}$$

where $\mathbf{v}_{ij}$ is the component of the eigenvector j corresponding to vertex i, and k is the number of eigenvalues.

Finally, the spectral entropy of vertex i is computed as follows:

$$SE_{V_i} = -\sum_j p_{ij} \log(p_{ij})$$

and normalized as:

$$SE_{V_i}^{norm} = SE_i / S_{max}$$

where $S_{max} = log(k)$.

*2) **Phase 2: Vertex Selection based on the STAGS Score (SS):*** Here we identify vertices with significant deviations in spectral entropy compared to their historical mean (i.e., average of spectral entropy across all graph snapshots up to current time t). This deviation is quantified using the STAGS Score (SS):

$$SS_V(t) = \left| SE_V(t) - \frac{\sum_{i=1}^{t-1} SE_V(i)}{t-1} \right|$$

where $SS_V(t)$ represents the STAGS score of vertex V at time window t, and $SE_V(t)$ denotes the spectral entropy of vertex V at time t. Vertices with $SS_V(t)$ exceeding a predefined threshold (δ) are selected for the next phase.

*3) **Phase 3: Neighborhood Expansion of the Selected Vertices:*** To ensure that selected vertices retain sufficient structural context, we expand the sample to include their n-hop neighborhood. For example, when $n = 2$, we include both immediate neighbors and their direct neighbors. This step ensures that the sampled subgraph preserves local connectivity patterns, allowing the model to focus on high-interest regions indicative of anomalies rather than isolated vertices.

IV. Experimental Setup

To evaluate the effectiveness of STAGS in improving both the scalability and accuracy for GNN-based network anomaly detection, we conduct extensive experiments across four state-of-the-art GNN-based anomaly detection models. Our evaluation compares the models' performance with STAGS versus the full graph. Experiments use a diverse set of graphs comprising 24 synthetic network graphs with varying sizes (up to 400M edges) and anomaly percentages, as well as 2 large real-world network graphs (up to 12B edges). We assess scalability in terms of computation time that encompasses training time and inference time. We assess accuracy using the AUC-ROC score and a novel metric, prediction lag, that quantifies how quickly a model detects anomalous vertices after they become anomalous. Additionally, we benchmark STAGS against two widely used graph-sampling techniques: uniform sampling and snowball sampling.

The following subsections provide details on the GNN-based anomaly detection models, evaluation metrics, network graphs, and alternative graph-sampling methods. We then articulate the computational resources, graph construction, and GNN training setup used in our experiments.

A. GNN-based Anomaly Detection Models

To evaluate STAGS, we use four state-of-the-art GNN-based anomaly detection models. These models span diverse architectures and learning paradigms, including convolutional, recurrent, transformer-based, and embedding-driven methods. This diversity ensures a comprehensive evaluation of STAGS.

1) GCRN: GCRN [18] is a recurrent neural networks (RNN)-based framework that combines graph convolutional networks (GCNs) to capture structural information with RNNs to model temporal dependencies, enabling it to track changes in graph over time.

2) StrGNN: StrGNN [10] is a gated recurrent units (GRUs)-based framework that extracts h-hop enclosing subgraphs around target edges and employs a vertex labeling function to distinguish vertex roles.

3) TADDY: TADDY [11] is a transformer-based framework designed for anomaly detection in graphs.

4) NetWalk: NetWalk [9] is an embedding-based framework that learns adaptive representations to track graph evolution over time.

B. Graph-Sampling Methods

We compare our STAGS against two widely used graph-sampling methods. Both methods accept a sampling percentage as input, which specifies the proportion of the graph to be retained. For instance, a 40% sampling rate indicates that 40% of the vertices and their associated edges are preserved.

1) Uniform Sampling: Uniform sampling selects vertices randomly from the graph, ensuring that each vertex has an equal probability of being chosen.

2) Snowball Sampling: Snowball sampling is a variant of the breadth-first search (BFS) algorithm that incrementally expands a sampled graph by iteratively including a fixed number of neighboring vertices.

C. Graphs

To comprehensively evaluate STAGS, we test GNN-based anomaly detection models on a diverse set of network graphs, encompassing both synthetic and real-world graphs. This diversity ensures a thorough assessment across different graph sizes, varying anomaly percentages and attack properties.

1) Synthetic Graphs: To rigorously assess model performance under controlled conditions, we generate synthetic graphs by varying key parameters such as edge count and anomaly percentage. We utilize two types of synthetic graphs: (1) graphs generated using Graph Generator with Attributes and Anomalies (G2A2) model [19] and (2) graphs produced through network emulation (NE). A qualitative description of both the sets of graphs is provided below, with a quantitative summary in Table I.

- **G2A2:** Graphs generated via G2A2 emulate structures from multiple domains, including the Internet (IN) and social media networks (SN). The G2A2 framework allows users to tune parameters such as edge count, anomaly percentage, and skewness to create customized dynamic attributed graphs with

TABLE I: Summary of graphs

Type	Graph Name	Graph Size (Edges)	Duration (Mins)	Anomaly Percentage	Total Graphs
Synthetic	G2A2-Internet Network (IN)	50M, 100M, 200M	60, 120, 240	0.25%, 0.5%, 1%, 5%	12
Synthetic	Network Emulation (NE-5-10)	200M, 400M	87, 183	6.74%	2
Synthetic	Network Emulation (NE-10-10)	200M, 400M	82, 175	13.75%	2
Synthetic	Network Emulation (NE-20-10)	200M, 400M	72, 164	28.89%	2
Synthetic	Network Emulation (NE-5-60)	200M, 400M	99, 207	0.91%	2
Synthetic	Network Emulation (NE-10-60)	200M, 400M	95, 197	1.94%	2
Synthetic	Network Emulation (NE-20-60)	200M, 400M	93, 189	4.08%	2
Real	P-core (fast)	5B	2880	0.01%	1
Real	P-core (slow)	12B	7200	0.007%	1
				Total	26

anomalies. Each graph follows the naming convention **G2A2-X-N-Y**, where **X** represents the domain, **N** denotes the number of edges, and **Y** specifies the anomaly percentage. For this paper, we are only going to focus on the Internet networks (IN).

- **Network Emulation (NE):** The second set of synthetic graphs is generated using network emulation (NE) within a controlled environment on a locally hosted server with restricted public network access. The experiments are conducted in an isolated Docker container, where a predefined number of victim virtual machines (VMs) are deployed across multiple local subnets. Each subnet is equipped with a dedicated router that captures network traffic using `tcpdump`. The collected packet traces from all routers are aggregated and processed by Zeek to produce Zeek logs. Specifically, the `conn.log` file from the Zeek logs is utilized to construct a synthetic NE graph, in which virtual machines and public network IPs are represented as vertices.
  To emulate adversarial activity (network anomalies), a Mirai botnet attack—similar to P-core [20] is deployed within a designated subnet separate from the victim machines. The attack infrastructure comprises an attacker, a command-and-control (C&C) server, and a listener, all residing within the same subnet. The attacker can compromise victim systems, which, in turn, can propagate the infection by scanning and compromising additional machines. A schematic representation of the NE Docker container is provided in Figure 2.
  Within the Mirai botnet codebase, key parameters are adjusted to control attack dynamics. Specifically, we modify the number of unique IP addresses scanned per attack (attack rate), which determines the extent of propagation attempts, and the time interval between successive attacks (attack frequency). Each emulated graph is represented as **NE-N-X-Y**, where **N** is the number of edges, **X** is the attack rate, and **Y** indicates the attack frequency.

2) Real-world Graphs: In addition to synthetic graphs, we evaluate our models on two real-world graphs: P-core (fast) and P-core (slow), as introduced in Buchanan et al. [20]

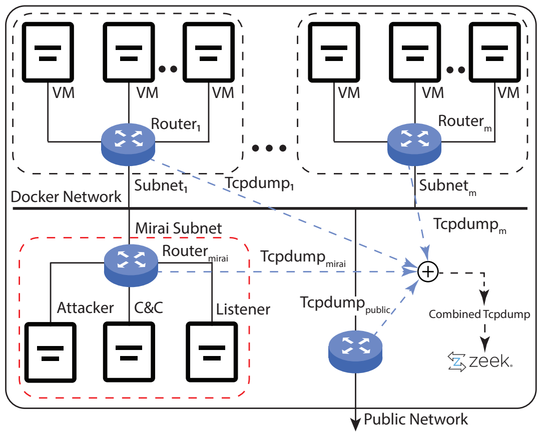


Fig. 2: Network emulation (NE) in the Docker container

The P-core dataset captures network traffic flow data transmitted through the edge server from the VT domain to the rest-of-the-world (RoW) domain and vice versa. The network graphs in P-core represent traffic traces, where vertices correspond to various systems, including servers, workstations, and IoT devices, while edges represent network connections. Each vertex is uniquely identified by its IP address and is enriched with attributes such as device type, geographic location, and historical connectivity data, including the number and types of past connections. Edges, which encode connection flows, contain multiple attributes extracted from Zeek's `conn.log`, including source and destination ports, protocol type, service, and connection duration. Additionally, edges incorporate historical information specific to the connection rather than the individual vertices.

Anomalous behavior in the P-core dataset is induced through a staged Mirai botnet attack, as described in Buchanan et al. [20]. The dataset is provided in two variants: P-core (fast) and P-core (slow). The P-core (fast) dataset represents a scenario where the Mirai botnet performs rapid scanning, targeting five unique IPs per attack (attack rate) and executing one scan or attack every 10 seconds (attack frequency) over a duration of two days. In contrast, the P-core (slow) dataset simulates a more stealthy attack pattern, targeting a single unique IP per attack (attack rate) with one scan occurring every five minutes (attack frequency) over a six-day period. Table I provides a quantitative summary of the graph characteristics.

D. Metrics

We evaluate STAGS in terms of scalability and accuracy improvement of GNN-based anomaly detection models. We compare results obtained with STAGS against without (i.e., with the full graph). Scalability is measured via tracking computation time that encompasses: 1) training time and 2) inference time, while accuracy is measured using two key evaluation metrics: 1) AUC-ROC score and 2) prediction lag. The evaluation metrics are detailed as follows:

1) Training and Inference Time: Training and inference time are computational efficiency metrics. Training time reflects the computational cost of model optimization, while inference time determines how quickly the model can process new data and detect anomalies.

2) AUC-ROC Score: The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) quantifies the model's ability to differentiate between normal and anomalous vertices in a dynamic attributed graph. A higher AUC-ROC score indicates superior classification accuracy for anomaly detection.

3) Prediction Lag: Prediction lag quantifies the temporal gap between the actual occurrence of an anomaly and the time at which the model identifies it. Given that t_{anomaly} represents the actual time of anomaly occurrence and t_{predict} denotes the time of detection, prediction lag is defined as:

$$\text{Prediction Lag (PL)} = t_{\text{predict}} - t_{\text{anomaly}}$$

A positive prediction lag indicates delayed detection, whereas a negative value corresponds to early detection. Since ground-truth labels for early anomaly occurrences are unavailable in our graph datasets, we focus here on delayed detection.

E. Compute Resources

All experiments were performed on the Falcon cluster from Advanced Research Computing (ARC) at Virginia Tech. Each node is equipped with an Intel® "Sapphire Rapids" Xeon® Platinum 8462Y+ processor running at 2.80 GHz with 64 cores, 512 GB of memory, and 40 NVIDIA L40s GPUs for model training and evaluation.

F. Graph Construction

When training GNNs on large-scale dynamic attributed network graphs, we encountered memory limitations when processing more than 200,000 vertices per snapshot. Additionally, it is crucial to maintain a small enough time window to capture changes in entropy. A larger time window would result in all anomalous vertices appearing in a single snapshot, which would hinder the effectiveness of STAGS. To address this, we adjust the time window for each graph to ensure that the snapshot fits within memory while still allowing for meaningful anomaly detection. For all graphs generated using G2A2 and for network emulation (NE), the time window is set to one minute. For real-world graphs, the time window is set to five minutes for the P-core (fast) and P-core (slow). The STAGS parameter weighting factors, W_v and W_e, are assigned in proportion to the ratio of vertex attributes to edge attributes. For all network graphs, we set $W_v = 0.2$ and $W_e = 0.8$, reflecting the common property of networks where edges generally contain more information and play a more significant role than vertices. The sigma values, σ_v^2 and σ_e^2, are set using the default settings of `scikit-learn`, which are inversely proportional to the number of features.

G. Training GNN

The four GNN models can be classified into two categories: (1) general-purpose GNNs and (2) GNNs designed for anomaly detection. We employed both oversampling and undersampling techniques to address class imbalances. For all models, we used the default parameters recommended by their respective papers and standardized the training procedure by setting the number of epochs to 200, ensuring a fair comparison across all models.

V. Results and Discussion

In this section, we compare and contrast the scalability and accuracy of GNN-based anomaly detection models with STAGS and without (i.e., with the full graph). We begin by addressing the scalability via computation time of models i.e., training time and inference time. Subsequently, we analyze the results on both synthetic and real-world graphs, examining the trends in accuracy metrics AUC-ROC score and prediction lag, and how these are further influenced by different graph properties such as edge count and anomaly percentage.

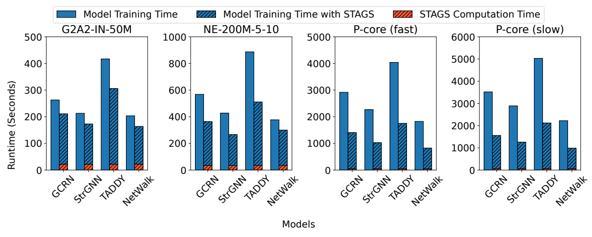
A. Computation Time (Training and Inference Time)

STAGS significantly reduces training time by decreasing the amount of data required for model training, as shown in Figure 3a. We observe that computationally intensive models like TADDY benefit more from this reduction compared to NetWalk. The greatest speedup occurs in larger graphs, such as P-core (slow), where we achieve up to a 2.44× improvement.

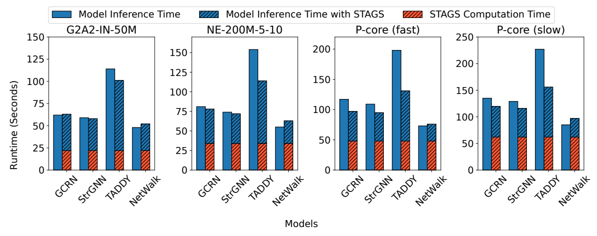
Additionally, the STAGS step introduces minimal computational overhead during training. However, its impact becomes more pronounced during inference, as shown in Figure 3b. While inference on 40 GPUs is typically fast and scalable, we still observe speedup in larger datasets, such as P-core (fast) and (slow), where data reduction exceeds 50%.

TABLE II: Accuracy (AUC-ROC) without vs. with STAGS and improvement (% improv.)

Graphs		GNN-based Anomaly Detection Models											
		GCRN			StrGNN			TADDY			NetWalk		
Name	**Anomaly%**	w/o STAGS	with STAGS	**% Improv.**	w/o STAGS	with STAGS	**% Improv.**	w/o STAGS	with STAGS	**% Improv.**	w/o STAGS	with STAGS	**% Improv.**
G2A2-IN-50M-0.25	0.25	0.54	0.67	**24.1**	0.61	0.7	**14.8**	0.65	0.76	**16.9**	0.4	0.5	**25.0**
G2A2-IN-50M-0.50	0.5	0.57	0.69	**21.1**	0.63	0.72	**14.3**	0.69	0.77	**11.6**	0.44	0.53	**20.5**
G2A2-IN-50M-1	1.0	0.64	0.75	**17.2**	0.67	0.76	**13.4**	0.75	0.82	**9.3**	0.47	0.55	**17.0**
G2A2-IN-50M-5	5.0	0.72	0.78	**8.3**	0.75	0.79	**5.3**	0.78	0.84	**7.7**	0.48	0.56	**16.7**
G2A2-IN-100M-0.25	0.25	0.53	0.65	**22.6**	0.61	0.71	**16.4**	0.66	0.75	**13.6**	0.41	0.52	**26.8**
G2A2-IN-100M-0.50	0.5	0.56	0.68	**21.4**	0.65	0.72	**10.8**	0.7	0.76	**8.6**	0.45	0.54	**20.0**
G2A2-IN-100M-1	1.0	0.65	0.75	**15.4**	0.67	0.76	**13.4**	0.74	0.81	**9.5**	0.48	0.56	**16.7**
G2A2-IN-100M-5	5.0	0.73	0.78	**6.8**	0.75	0.79	**5.3**	0.77	0.82	**6.5**	0.49	0.56	**14.3**
G2A2-IN-200M-0.25	0.25	0.5	0.63	**26.0**	0.63	0.72	**14.3**	0.67	0.76	**13.4**	0.42	0.53	**26.2**
G2A2-IN-200M-0.50	0.5	0.58	0.69	**19.0**	0.67	0.74	**10.4**	0.7	0.76	**8.6**	0.47	0.57	**21.3**
G2A2-IN-200M-1	1.0	0.64	0.74	**15.6**	0.71	0.77	**8.5**	0.73	0.78	**6.8**	0.51	0.59	**15.7**
G2A2-IN-200M-5	5.0	0.7	0.78	**11.4**	0.75	0.78	**4.0**	0.77	0.8	**3.9**	0.52	0.6	**15.4**
NE-200M-5-10	6.74	0.69	0.76	**10.1**	0.57	0.72	**26.3**	0.7	0.75	**7.1**	0.59	0.67	**13.6**
NE-400M-5-10	6.74	0.68	0.77	**13.2**	0.58	0.73	**25.9**	0.71	0.76	**7.0**	0.58	0.66	**13.8**
NE-200M-10-10	13.75	0.73	0.79	**8.2**	0.62	0.75	**21.0**	0.74	0.78	**5.4**	0.58	0.69	**19.0**
NE-400M-10-10	13.75	0.74	0.78	**5.4**	0.63	0.74	**17.5**	0.75	0.78	**4.0**	0.58	0.7	**20.7**
NE-200M-20-10	28.89	0.79	0.83	**5.1**	0.69	0.78	**13.0**	0.72	0.77	**6.9**	0.62	0.73	**17.7**
NE-400M-20-10	28.89	0.78	0.83	**6.4**	0.68	0.78	**14.7**	0.72	0.78	**8.3**	0.63	0.72	**14.3**
NE-200M-5-60	0.91	0.47	0.64	**36.2**	0.45	0.65	**44.4**	0.51	0.67	**31.4**	0.19	0.31	**63.2**
NE-400M-5-60	0.91	0.48	0.64	**33.3**	0.44	0.63	**43.2**	0.52	0.67	**28.8**	0.18	0.3	**66.7**
NE-200M-10-60	1.94	0.54	0.69	**27.8**	0.58	0.72	**24.1**	0.5	0.69	**38.0**	0.26	0.45	**73.1**
NE-400M-10-60	1.94	0.54	0.68	**25.9**	0.59	0.73	**23.7**	0.52	0.7	**34.6**	0.25	0.44	**76.0**
NE-200M-20-60	4.08	0.65	0.76	**16.9**	0.62	0.77	**24.2**	0.63	0.74	**17.5**	0.29	0.46	**58.6**
NE-400M-20-60	4.08	0.66	0.78	**18.2**	0.62	0.78	**25.8**	0.64	0.73	**14.1**	0.25	0.46	**84.0**
P-core (fast)	0.01	0.78	0.91	**16.7**	0.58	0.85	**46.6**	0.78	0.92	**17.9**	0.29	0.45	**55.2**
P-core (slow)	0.007	0.58	0.89	**53.4**	0.54	0.79	**46.3**	0.75	0.9	**20.0**	0.17	0.29	**70.6**



(a) Training time



(b) Inference time

Fig. 3: Computation time without vs. with STAGS

B. Accuracy (AUC-ROC)

The results from G2A2 graphs demonstrate that STAGS consistently improves accuracy for all of the models as shown in Table II. In particular, STAGS yields greater accuracy improvements in graphs with a lower percentage of anomalies than in those with higher percentages of anomalies. Even models designed for anomaly detection, such as StrGNN and TADDY, did not particularly perform well on their own (without sampling) with graphs that had the smallest anomaly percentages. Particularly, we see an accuracy improvement up to 24%, 21%, 16% and 25% for GCRN, StrGNN, TADDY and NetWalk when applied to G2A2's generated graphs.

For network emulation graphs, the most difficult anomalies to detect were those involving a low number of scanning IPs per attack and a large time gap between two attacks. For example, the NE-N-5-60 graph proved to be the most difficult for the models to identify anomalies, where $N = \{200M, 400M\}$. However, STAGS showed substantial improvements in accuracy, especially for these more challenging graphs. In particular, we see an improvement in accuracy up to 36%, 44%, 31% and 63% for GCRN, StrGNN, TADDY and NetWalk when applied to NE graphs.

A similar trend was observed in the real-world graphs, where the percentages of anomalies were particularly low ($< 0.15\%$). We see accuracy improvement upto 53%, 46%, 20% and 70% for GCRN, StrGNN, TADDY and NetWalk when applied on real-world graphs. In general, we observed that STAGS provides more improvement in accuracy when applied on low anomaly percentage graph as shown in Figure 4.

STAGS improves anomaly visibility post-sampling, creating a more balanced class distribution for training while filtering out noise. In the P-core dataset, DNS servers often resemble Mirai botnet attacks in structure. However, because DNS network patterns remain stable over time, their STAGS scores stay near zero. In contrast, victim and attacker systems exhibit significant structural and attribute shifts, leading to STAGS scores closer to one, making them more likely to be sampled in each graph snapshot.

Figure 5 illustrates the distribution of average vertex STAGS scores across snapshots on the network emulation graphs. We can see that the anomalous vertices tend to have higher STAGS scores. For the experiments in this paper, we fixed the threshold (δ) at 0.5, but this is an adjustable parameter. By setting δ to 0.5, we were able to select 91% of anomalous vertices of the graph while reducing the overall number of vertices by 51.9%. Altering this threshold results in a trade-off between data reduction and accuracy.

When compared to other graph sampling algorithms,

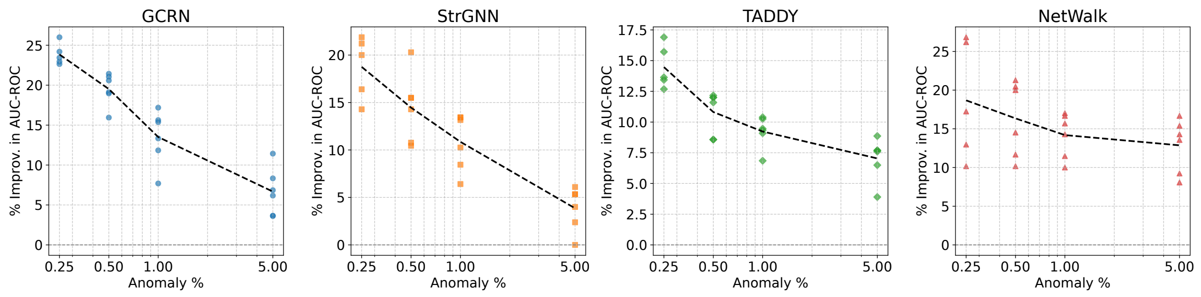


Fig. 4: Accuracy (AUC-ROC) improvement (% improv.) for GNN-based anomaly detection models with STAGS across different anomaly %. Results are for 24 G2A2 graphs (G2A2 enables simulation of varying anomaly %.)

as shown in Table III, STAGS outperforms random and snowball sampling, especially at low sampling percentages. Figure 6 illustrates the accuracy patterns across varying sampling percentages, where both random and snowball sampling show the best accuracy at 100%, indicating that these methods inevitably lose accuracy when sampling. In contrast, STAGS achieves its highest accuracy around 60%, demonstrating that it not only allows for data reduction but also improves model performance. Overall, we conclude that STAGS improves the accuracy of any GNN model, particularly when applied to graphs with sparse and complex anomalies.

C. Accuracy (Prediction Lag)

STAGS also reduces prediction lag across all models, as shown in Table IV. This improvement stems from the overall accuracy boost achieved with STAGS. However, we observe that certain models, like NetWalk, exhibit shorter prediction lag despite having a lower AUC-ROC than other models. This suggests that while some models are more accurate, their predictions may be delayed relative to others. Specifically, we see prediction lag improvement upto 22%, 42%, 28% and 26% for GCRN, StrGNN, TADDY, and NetWalk, respectively.

This behavior occurs because some models rely on the anomalous graph structure to fully develop before making confident predictions. STAGS accelerates this process by filtering out irrelevant vertices and edges, allowing anomalies to become apparent more quickly and reducing prediction lag.

In the P-core dataset, the fast and slow variants represent the same traffic, but the Mirai botnet attack progresses more slowly in the P-core (slow) graph, leading to longer prediction lag, as missed detections in the slower variant delay the occurrence of the next attack versus the faster P-core graph.

VI. Conclusion

We introduced STAGS (Structural, Temporal, and Attribute-aware Graph Sampling), a novel graph sampling method designed to enhance both the scalability and accuracy of GNN-based network anomaly detection. By incorporating structural, temporal, and

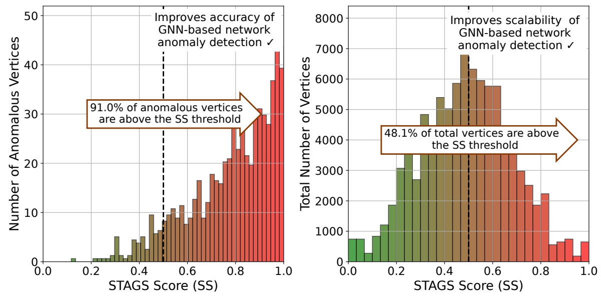


Fig. 5: STAGS score (SS) distribution of anomalous vertices and all the vertices. The SS of anomalous vertices are skewed towards one.

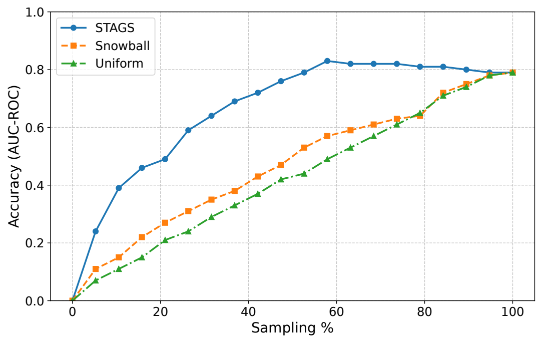


Fig. 6: STAGS vs. other sampling techniques

attribute-aware sampling, STAGS effectively identifies and retains the most informative regions of large-scale dynamic attributed graphs, ensuring that critical anomaly-relevant patterns are preserved while reducing computational overhead.

Our extensive evaluation on 24 synthetic and 2 real-world network graphs demonstrates the significant advantages of STAGS over training on the full graph. STAGS improves scalability (GNN training time by a factor of 2.44×) and accuracy (AUC-ROC by up to 53% and prediction lag by up to 42%). Moreover, STAGS outperforms widely used graph sampling techniques, such as random and snowball sampling, by ensuring that sampled subgraphs remain highly representative of anomaly-prone regions due to its structural, temporal and attribute awareness.

TABLE III: Accuracy (AUC-ROC) of STAGS vs. uniform and snowball sampling on real-world graphs.

Graphs	**Sampling %**	**GNN-based Anomaly Detection Models**											
		GCRN			**StrGNN**			**TADDY**			**NetWalk**		
		STAGS	Uniform	Snowball	**STAGS**	Uniform	Snowball	**STAGS**	Uniform	Snowball	**STAGS**	Uniform	Snowball
P-core (fast)	42	**0.91**	0.51	0.54	**0.85**	0.47	0.49	**0.92**	0.53	0.59	**0.45**	0.21	0.23
P-core (slow)	41	**0.89**	0.5	0.54	**0.79**	0.46	0.48	**0.9**	0.53	0.58	**0.29**	0.18	0.19

TABLE IV: Accuracy (prediction lag (mins)) without vs. with STAGS and improvement (% Improv.)

Graphs		**GNN-based Anomaly Detection Models**											
		GCRN			**StrGNN**			**TADDY**			**NetWalk**		
Name	**Anomaly%**	w/o STAGS	with STAGS	**% Improv.**	w/o STAGS	with STAGS	**% Improv.**	w/o STAGS	with STAGS	**% Improv.**	w/o STAGS	with STAGS	**Improv. %**
G2A2-IN-50M-0.25	0.25	10.9	9.4	**13.8**	13.8	11.2	**18.8**	9.7	8.9	**8.3**	7.9	6.1	**22.8**
G2A2-IN-50M-0.50	0.5	8.4	7.3	**13.1**	12.7	10.5	**17.3**	8.8	7.9	**10.2**	7.5	5.9	**21.3**
G2A2-IN-50M-1	1.0	7.8	6.6	**15.4**	11.1	9.5	**14.4**	7.9	7.1	**10.1**	6.9	5.4	**21.7**
G2A2-IN-50M-5	5.0	5.7	4.9	**14.0**	8.9	7.4	**16.9**	6.8	6.2	**8.8**	6.2	5.0	**19.4**
G2A2-IN-100M-0.25	0.25	10.7	8.9	**17.1**	13.2	10.9	**17.7**	9.5	8.7	**8.7**	7.8	6.3	**20.0**
G2A2-IN-100M-0.50	0.5	8.2	7.2	**12.5**	13.0	10.5	**19.0**	9.0	8.1	**10.2**	7.2	5.7	**20.6**
G2A2-IN-100M-1	1.0	8.3	6.8	**18.1**	11.3	9.4	**16.7**	7.4	6.6	**10.7**	7.1	5.3	**24.8**
G2A2-IN-100M-5	5.0	5.9	5.1	**12.5**	9.1	7.9	**13.2**	6.4	5.7	**10.6**	6.5	5.2	**20.5**
G2A2-IN-200M-0.25	0.25	11.2	9.4	**16.3**	13.5	11.0	**18.8**	9.8	8.7	**11.5**	7.7	5.9	**23.5**
G2A2-IN-200M-0.50	0.5	8.2	7.1	**13.4**	12.7	10.4	**17.8**	8.6	7.9	**8.1**	7.7	6.0	**22.0**
G2A2-IN-200M-1	1.0	8.2	6.4	**22.1**	11.3	9.2	**18.4**	8.0	7.1	**11.4**	7.1	5.4	**23.1**
G2A2-IN-200M-5	5.0	5.6	4.8	**13.9**	8.6	7.4	**13.4**	6.5	6.1	**6.0**	6.5	5.2	**20.5**
NE-200M-5-10	6.74	6.8	5.5	**19.1**	8.3	7.1	**14.5**	7.1	6.4	**9.9**	5.4	4.3	**20.4**
NE-400M-5-10	6.74	6.7	5.5	**17.9**	8.3	7.0	**15.7**	7.2	6.3	**12.5**	5.4	4.4	**18.5**
NE-200M-10-10	13.75	5.1	4.4	**13.7**	7.5	6.5	**13.3**	5.5	5.1	**7.3**	4.7	3.9	**17.0**
NE-400M-10-10	13.75	5.2	4.5	**13.5**	7.6	6.5	**14.5**	5.5	5.1	**7.3**	4.8	3.9	**18.8**
NE-200M-20-10	28.89	3.9	3.4	**12.8**	6.5	5.8	**10.8**	4.2	3.8	**9.5**	3.7	3.2	**13.5**
NE-400M-20-10	28.89	3.9	3.5	**10.3**	6.5	5.7	**12.3**	4.2	3.8	**9.5**	3.7	3.1	**16.2**
NE-200M-5-60	0.91	19.7	17.4	**11.7**	22.1	18.7	**15.4**	20.9	18.7	**10.5**	18.5	14.9	**19.5**
NE-400M-5-60	0.91	19.4	17.2	**11.3**	22.3	18.6	**16.6**	20.7	18.9	**8.7**	18.7	14.8	**20.9**
NE-200M-10-60	1.94	13.6	11.9	**12.5**	15.9	13.1	**17.6**	15.9	14.1	**11.3**	12.9	10.7	**17.1**
NE-400M-10-60	1.94	13.9	11.8	**15.1**	15.7	13.5	**14.0**	15.7	14.2	**9.6**	12.8	10.6	**17.2**
NE-200M-20-60	4.08	9.7	8.3	**14.4**	12.9	11.1	**14.0**	11.1	10.1	**9.0**	8.5	6.8	**20.0**
NE-400M-20-60	4.08	9.5	8.5	**10.5**	12.7	11.0	**13.4**	11.1	9.9	**10.8**	8.7	6.9	**20.7**
P-core (fast)	0.01	13.0	11.0	**15.4**	28.0	19.0	**32.1**	19.0	15.0	**21.1**	15.0	12.0	**20.0**
P-core (slow)	0.007	46.0	40.0	**13.0**	97.0	56.0	**42.3**	39.0	28.0	**28.2**	38.0	28.0	**26.3**

References

[1] H. Kim, B. S. Lee, W. Y. Shin, and S. Lim, "Graph Anomaly Detection With Graph Neural Networks: Current Status and Challenges," *IEEE Access*, vol. 10, 2022.

[2] J. Leskovec and C. Faloutsos, "Sampling from Large Graphs," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 2006, 2006.

[3] F. Wanye, V. Gleyzer, E. Kao, and W.-c. Feng, "SamBaS: Sampling-Based Stochastic Block Partitioning," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 3, pp. 3053–3065, 5 2024.

[4] Y. Wang, K. Hallgren, and J. Larson, "A graph-based framework for reducing false positives in authentication alerts in security systems," in *Companion Proceedings of the ACM Web Conference 2024*, ser. WWW '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 274–283. [Online]. Available: https://doi.org/10.1145/3589335.3648325

[5] E. Stalmans and B. Irwin, "A framework for dns based detection and mitigation of malware infections on a network," in *2011 Information Security for South Africa*, 2011, pp. 1–8.

[6] R. Paudel and H. H. Huang, "Pikachu: Temporal Walk Based Dynamic Graph Embedding for Network Anomaly Detection," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium 2022: Network and Service Management in the Era of Cloudification, Softwarization and Artificial Intelligence, NOMS 2022*, 2022.

[7] T. Bian, X. Xiao, T. Xu, P. Zhao, W. Huang, Y. Rong, and J. Huang, "Rumor Detection on Social Media with Bi-directional Graph Convolutional Networks," in *34th AAAI Conf. on Artificial Intelligence*, 2020.

[8] S. Jha and W. Feng, "GRAPPEL: A Graph-based Approach for Early Risk Assessment of Acute Hypertension in Critical Care," in *ACM-BCB 2023 - 14th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2023.

[9] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[10] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen, "Structural Temporal Graph Neural Networks for Anomaly Detection in Dynamic Graphs," in *Proc. International Conference on Information and Knowledge Management*, 2021.

[11] Y. Liu, S. Pan, Y. G. Wang, F. Xiong, L. Wang, Q. Chen, and V. C. Lee, "Anomaly Detection in Dynamic Graphs via Transformer," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, 2023.

[12] W. L. Chiang, Y. Li, X. Liu, S. Bengio, S. Si, and C. J. Hsieh, "Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[13] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph Sampling based Inductive Learning Method," in *Proc. 8th International Conference on Learning Representations*, 2020.

[14] R. Gao, H. Xu, P. Hu, and W. C. Lau, "Accelerating Graph Mining Algorithms via Uniform Random Edge Sampling," in *2016 IEEE International Conference on Communications, ICC 2016*, 2016.

[15] R. H. Li, J. X. Yu, L. Qin, R. Mao, and T. Jin, "On Random Walk based Graph Sampling," in *Proceedings - International Conference on Data Engineering*, vol. 2015-May, 2015.

[16] J. Batson, D. A. Spielman, N. Srivastava, and S. H. Teng, "Spectral Sparsification of Graphs: Theory and Algorithms," *Communications of the ACM*, vol. 56, no. 8, 2013.

[17] A. Loukas, "Graph Reduction with Spectral and Cut Guarantees," *Journal of Machine Learning Research*, vol. 20, 2019.

[18] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured Sequence Modeling with Graph Convolutional Recurrent Networks," in *Lecture Notes in Computer Science*, vol. 11301 LNCS, 2018.

[19] S. Dey, S. Jha, and W. Feng, "G2A2: Graph Generator with Attributes and Anomalies," in *Proc. 21st ACM Int'l Conf. on Computing Frontiers*, ser. CF '24, New York, NY, USA, 2024, pp. 3–11.

[20] M. Buchanan, J. W. Collyer, J. W. Davidson, S. Dey, M. Gardner, J. D. Hiser, J. Lang, A. Nottingham, and A. Oprea, "On Generating and Labeling Network Traffic with Realistic, Self-Propagating Malware," *CoRR*, vol. abs/2104.10034, 2021. [Online]. Available: https://arxiv.org/abs/2104.10034