applied sciences

MDPI

Article

Adaptive DecayRank: Real-Time Anomaly Detection in Dynamic Graphs with Bayesian PageRank Updates

Ocheme Anthony Ekle [1,†], William Eberle [1,†] and Jared Christopher [2,*]

[1] Department of Computer Science, Tennessee Technological University, Cookeville, TN 38505, USA; oaekle42@tntech.edu (O.A.E.); weberle@tntech.edu (W.E.)
[2] Department of Computer Science, Southern Illinois University, Edwardsville, IL 62026, USA
* Correspondence: jarchri@siue.edu
† These authors contributed equally to this work.

Abstract: Real-time anomaly detection in large, dynamic graph networks is crucial for real-world applications such as network intrusion prevention, fraud transaction identification, fake news detection in social networks, and uncovering abnormal communication patterns. However, existing graph-based methods often focus on static graph structures, which struggle to adapt to the evolving nature of these graphs. In this paper, we propose ADAPTIVE-DECAYRANK, a real-time and adaptive anomaly detection model for dynamic graph streams. Our method extends the dynamic PageRank algorithm by incorporating an adaptive Bayesian updating mechanism, allowing nodes to dynamically adjust their decay factors based on observed graph changes. This enables real-time detection of sudden structural shifts, improving anomaly identification in streaming graphs. We evaluate ADAPTIVE-DECAYRANK on multiple real-world security datasets, including DARPA and CTU-13, as well as synthetic dense graphs generated using RTM. Our experiments demonstrate that ADAPTIVE-DECAYRANK outperforms state-of-the-art methods, such as ANOMRANK, SEDANSPOT, and DYNANOM, achieving up to 13.94% higher precision, 8.43% higher AUC, and more robust detection in highly dynamic environments.

Keywords: anomaly detection; real-time; dynamic graphs; node scoring; structural anomalies; Bayesian updating; dynamic PageRank

Academic Editor: Tao Zhou

Received: 6 February 2025
Revised: 12 March 2025
Accepted: 14 March 2025
Published: 19 March 2025

Citation: Ekle, O.A.; Eberle, W.; Christopher, J. Adaptive DecayRank: Real-Time Anomaly Detection in Dynamic Graphs with Bayesian PageRank Updates. *Appl. Sci.* **2025**, *15*, 3360. https://doi.org/10.3390/app15063360

1. Introduction

Research on anomaly detection in graph networks has increasingly focused on the development of adaptive and real-time solutions capable of handling complex streaming graph data [1,2]. Anomaly detection is an important task in both static and dynamic networks. Unlike static networks, where the topology remains constant, dynamic networks are continuously (or periodically) changing their nodes and edges [2]. Figure 1 illustrates two graph representations: (a) a static graph, G, and (b) presents a snapshot of the evolving dynamic graph ($\mathcal{G} = (V_t, E_t, \mathcal{T})$). The need for real-time graph-based detection algorithms tailored to dynamic graphs is critical for addressing diverse real-world challenges. These applications span network intrusion prevention [3], fraud detection in financial transactions [4], drug discovery and motif discovery in biological networks, power grid monitoring [5], and machine learning-based vulnerability categorization [6]. The wide-ranging applicability of these methods reinforces the importance of adaptive machine learning models in tackling real-world problems and highlights the growing demand for innovative approaches to anomaly detection in dynamic graph settings.

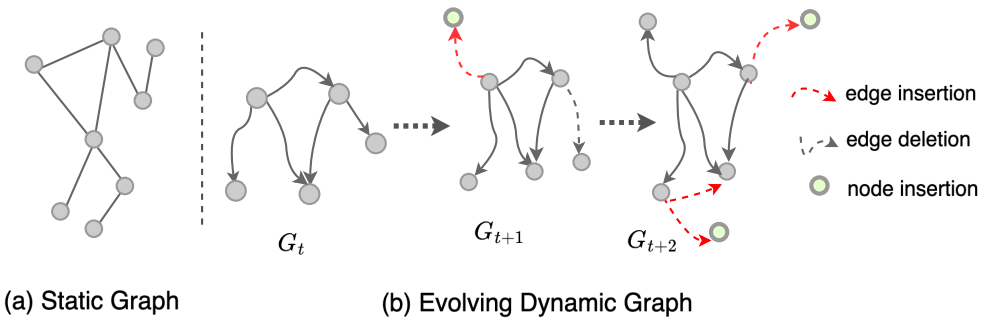

Figure 1. An illustration of graph representation: (**a**) Static graph, G, and (**b**) evolving dynamic graph, $\mathcal{G} = (V_t, E_t, \mathcal{T})$, showing a series of graph snapshots with edge insertions, deletions, and node insertions over time.

Existing graph-based anomaly detection methods in the literature focus more on static graphs [7,8]. However, graph-based approaches still face significant limitations when it comes to inductive learning and adapting to the evolving nature of real-world graphs [2], where edge connections and node importance can change rapidly.

Among the proposed methods for anomaly detection in dynamic graphs, these can be categorized into matrix factorization and tensor decomposition, probabilistic models, distance-based methods, and deep-learning graph embeddings [2]. Despite their respective successes, these techniques often face limitations in terms of speed, accuracy, adaptability to concept drift, and scalability. Detecting anomalous network behaviors in real-world critical infrastructures, such as power grids and nuclear plants, necessitates prompt responses with maximum accuracy.

1.1. Hypothesis and Research Questions

Given a large, time-evolving graph stream, $\mathcal{G} = \{G_t\}_{t=1}^{T}$, where structural changes occur dynamically across timestamps, t, can we develop an **adaptive, near-real-time, and scalable** anomaly detection framework that leverages an *adaptive decay factor with Bayesian updating* to effectively detect anomalous patterns with high precision and robustness.

We aim to address the following questions in our work:

- **Q1:** Can we detect sudden changes in dynamic graphs, including node insertions, edge formations, and structural deviations, with high precision while minimizing false positives?
- **Q2:** Can we ensure **scalability** in real-time anomaly detection for high-velocity graph streams while maintaining **low computational overhead**.

1.2. Contributions

In this paper, we propose ADAPTIVE-DECAYRANK, a novel real-time and adaptive anomaly detection model for dynamic graphs. This model utilizes a modified version of the dynamic PageRank algorithm [9] tailored to adapt and continuously update node scores based on recent structural changes in the graph. In ADAPTIVE-DECAYRANK, we introduced a novel **decay factor** formula with Bayesian updating; this was inspired by the *"Weight Decay Regularization in ADAM."* [10], where the authors propose decoupling gradient-based updates from weight decay in both the SGD [11] and Adam [12] optimizers. The *decay factor* in ADAPTIVE-DECAYRANK is strategically incorporated into the dynamic PageRank algorithm and is dynamically adjusted based on the observed changes of edge influx at each time step of the graph, utilizing a Bayesian updating mechanism.

The decay factor formula determines how quickly past information (nodes and edges) is discounted, allowing the algorithm to react swiftly to recent changes. By incorporating Bayesian updating inference, our model dynamically adjusts the decay factor for each node

based on observed changes in node scores. This ensures that nodes with rapid changes in connectivity receive higher scores, reflecting their increased importance and potential anomalous behavior.

The motivation behind ADAPTIVE-DECAYRANK lies in its ability to capture abrupt structural alterations in dynamic graph networks using the adaptive decay mechanism. We consider anomalous behavior as sudden changes in the structural composition of node scores, specifically the rapid addition of new edges between unrelated nodes as these networks evolve over time. For example, a sudden increase in node scores might signify a surge in activity, suggesting a noteworthy event or a potential security threat.

Our contributions are summarized as follows:

- We introduce ADAPTIVE-DECAYRANK, a modified dynamic PageRank with a Bayesian updating mechanism and an adaptive decay factor for fast anomaly detection and real-time adaptation to graph changes.
- We demonstrate that our ADAPTIVE-DECAYRANK prioritizes recent edge changes and assigns higher anomaly scores to structural deviation, thereby improving detection accuracy.
- We conducted extensive experiments on real-world dynamic graph datasets, demonstrating that ADAPTIVE-DECAYRANK outperforms state-of-the-art baselines in both precision and AUC while maintaining competitive runtime performance.

2. Related Work

In this section, we review existing methods for detecting anomalies in dynamic graphs. We categorize these methods into four groups based on their algorithmic structures. For a more comprehensive understanding of anomaly detection in dynamic graphs, we refer readers to the following reviews [1,2].

Distance and similarity-based methods measure similarities between nodes using time-evolving metrics to detect anomalies. Common metrics include PageRank [13] and Betweenness Centrality [14]. However, these similarity-based methods rely on static graphs, making them less effective in capturing real-time changes in dynamic graphs. The concept of ranking nodes by their importance or influence has been extensively studied in network science. Lü et al. [15] provide a comprehensive review of node centrality measures, including various personalized PageRank (PPR) [16] approaches, which have also been adapted for anomaly detection in dynamic graphs. Many methods in this category focus on identifying vital nodes to maximize influence propagation (for example, in social networks or epidemic spreading models [17]), but their applicability to streaming anomaly detection remains limited due to computational constraints. SedanSpot [18] is a randomized algorithm for detecting sparsely connected edges and identifying anomalies based on edge occurrence. However, SedanSpot processes input streams linearly, making it highly computationally expensive. In contrast, our method scales efficiently on large graphs in seconds. GBAD [19] leverages the Minimum Description Length (MDL) principle and a probabilistic approach to detect graph-based structural anomalies. AnomRank [20] applies personalized PageRank to capture both structural and edge-weight anomalies. However, in contrast to our method, AnomRank computes a global PageRank score, which does not scale efficiently for edge streaming processing. Holme and Saramäki [21] introduced the concept of "Temporal Networks", which highlights the challenges of detecting anomalies in streaming graphs with evolving nodes and edges. In contrast, our work focuses on anomaly detection in dynamic graphs; temporal networks provide a strong theoretical foundation for understanding evolving structures. SnapSketch [22] is a sketch-based approach; however, it struggles with detecting sudden structural changes in dynamic graphs, a limitation our method addresses by incorporating adaptive sensitivity to micro-changes in evolving graph

states. DYNWATCH [5] uses the Line Outage Distribution Factors (LODFs) sensitivity measure for real-time detection in power grid sensors. DYNWATCH constructs a graph from active grid devices such as nodes and performs temporal weighting based on graph distances for anomaly detection. However, DYNWATCH relies on predefined sensitivity measures like LODF, making it less adaptive to dynamic network topologies beyond power grids. DynAnom [23] is a method to track node-level and graph-level anomalies using personalized PageRank (PPR) in large graphs. It dynamically weights node representations to reflect evolving graph structures. However, it is limited by its reliance on PPR, which can introduce bias toward high-degree nodes and is less effective in capturing rapid structural changes in graphs.

Probabilistic methods use probabilistic models to represent neighborhood relationships in graphs. RHSS [24] employs Count-Min sketches to approximate graph properties and generate probabilistic error bounds on each edge outlier scoring function. However, RHSS has a limitation in its reliance on fixed probabilistic bounds, which may lead to overestimation or underestimation of anomaly scores due to hash collisions. In contrast, our method not only captures evolving patterns but also adjusts node importance scores in real time. PENminer [25] explores the persistence of activity snippets in edge streams, which is relevant for short sequences of recurring edge updates; however, it is not equipped to detect subgraph- or graph-level anomalies. MIDAS-R [26] detects microcluster anomalies in edge streams using Count-Min Sketches (CMSs) to track the frequency of edge occurrences at each timestamp and, subsequently, utilizes the Chi-squared test to assess the extent of deviation from typical edges in order to calculate anomaly scores. F-FADE [27] discovers anomalies by estimating edge patterns using the maximum likelihood rule of observed instances for each incoming interaction, while AnoEdge [28], an extension of MIDAS-R, focuses on higher-order sketches. However, these methods (F-FADE, MIDAS-R, and AnoEdge) require high computational time for large graph streams and primarily target the detection of edge-level or subgraph-level anomalies. In contrast, our method captures both node-level and structural anomalies in real time, offering a more comprehensive approach to anomaly detection in dynamic graph streams.

Matrix factorization methods decompose high-dimensional matrices into lower-dimensional forms, revealing evolving patterns in graphs. Recent methods include EdgeMonitor [29], an edge-detection approach that models dynamic graph evolution as a first-order Markov process. However, a major drawback is its reliance on consistent node ordering across all time steps, and it assumes a constant number of nodes per snapshot, which is often violated in large-scale graphs. DenseAlert [30] is an incremental and continuously updating method for detecting dense subtensors in tensor streams. LAD [31] applies the Laplacian spectrum for change detection by computing the singular value decomposition (SVD) of the graph Laplacian to obtain a low-dimensional graph representation. MultiLAD [32] generalizes LAD to multi-view graph detection. Despite the successes of matrix factorization-based methods like EdgeMonitor, DenseAlert, LAD, and MultiLAD, these approaches are computationally intensive, require manual extraction of graph properties, and are highly susceptible to noise. In contrast, ADAPTIVE-DECAYRANK dynamically updates node scores in real time without requiring global recomputation, ensuring scalability and efficiency for large-scale graph streams.

Deep graph learning methods leverage neural networks to extract and learn graph representations. H-VGRAE [33] employs graph autoencoders for embedding and node-level detection, constructing a hierarchical model by combining a variational graph autoencoder with a recurrent neural network (RNN). ROLAND [34] extends classical graph neural networks (GNNs) to capture dynamic graph structures by leveraging a hierarchical node state update mechanism. It allows static GNNs to be adapted for dynamic graphs using

recurrent updates and an incremental training approach. However, ROLAND and other deep learning approaches, such as H-VGRAE, assume that node embeddings can be incrementally updated without complete recomputation, leading to information loss over long sequences. In contrast, Adaptive-DecayRank is designed for real-time anomaly detection in dynamic graphs. Unlike deep learning-based methods, it does not require retraining from scratch or extensive hyperparameter tuning, making it computationally efficient and scalable for large graph streams. Additionally, Transformer models, such as Graphomer [35] and TADDY [36], have also been utilized for dynamic graph representation learning. However, these methods face limitations when dealing with highly irregular and evolving topologies, as they often struggle to capture local structural dependencies efficiently.

In summary, our method, ADAPTIVE-DECAYRANK, addresses a broader range of anomalies by detecting both node-level and structural anomalies in streaming graphs. ADAPTIVE-DECAYRANK aligns with distance-based and similarity-based approaches, leveraging dynamic PageRank with a *decay factor* and *Bayesian updating* as a similarity metric to assign node importance scores and conduct structural analysis. The dynamical Bayesian updating incorporated in our algorithm tracks micro-changes in edges at each timestamp, leading to faster and more accurate detection compared to existing methods in the literature, which are computationally intensive and slow to adapt to sudden deviations in dynamic graphs. We provide a summary of our method's properties compared to existing approaches in Table 1.

Table 1. Comparison of relevant anomaly detection methods in dynamic graphs.

Property	SedanSpot (2018)	MIDAS-R (2020)	F-FADE (2021)	DYNWATCH (2021)	DynAnom (2022)	MultiLAD (2023)	AnoEdge (2023)	DecayRank (May, 2024)	Our Method
Edge Anomaly	✓	✓	✓	✓			✓	-	-
Node Anomaly					✓	✓		✓	✓
Real-time Detection		✓	✓	✓	✓		✓	✓	✓
Structural Anomalies						✓		✓	✓
Sudden Edge Changes	✓	✓	✓	✓			✓	✓	✓
Adaptive Bayesian Updating									✓

Note: The checkmarks (✓) indicate that the corresponding method supports the listed property. A dash (-) means the feature is not applicable or not reported in the respective study.

3. Background

In this section, we present some definitions related to dynamic graphs, and review the personalized PageRank algorithm. Additionally, we outline the problem formulation of our anomaly detection method. The complete set of notations is provided in Table 2.

Definition 1. *A **graph**, $G = (V, E)$, is defined as a pair consisting of a set of nodes, $V = \{v_1, \ldots, v_n\}$, and a set of edges, E, which is a subset of the Cartesian product $E \subseteq V \times V$ and defines the connections between the nodes.*

Definition 2. *A **weighted graph,** $G = (V, E, W)$, is a graph where each edge $e \in E$ has an associated weight, $W(e) \in \mathbb{R}$.*

Definition 3. *A **graph snapshot** at a specific timestamp, t, is denoted as $G_t = (V_t, E_t, W_t)$, where $V_t = \{v \in V \mid i_v = t\}$ is the set of nodes present at time t, and $E_t = \{e \in E \mid i_e = t\}$ is the set of edges existing between the nodes in V_t at time t; W_t represents the weights assigned to the edges in E_t, which may consist of plain or labeled edges.*

Table 2. List of Notations.

Symbol	Definition
$G = (V, E)$	Weighted graph with nodes, V, and edges, E.
$\mathcal{G} = \{G_t\}_{t=1}^{T}$	Dynamic graph as a sequence of snapshots at each timestamp, t.
$G_t = (V_t, E_t, W_t)$	Snapshot of the graph at time t, with a set of vertices, V_t, a set of edges, E_t, and edge-weights, W_t.
T	Total number of time steps.
$W(e) \in \mathbb{R}$	Weight function, $W : E \to \mathbb{R}$, assigning a real-valued weight, $W(e)$, to each edge, $e \in E$.
V	Set of all vertices (nodes) in the graph.
v_u	Specific vertex, u, in the graph, with an associated PageRank score.
ΔE_t	Change in edges (insertions or deletions) between t and $t+1$.
ΔV_t	Change in nodes (additions or removals) between t and $t+1$.
P	Row-normalized adjacency matrix, $P \in \mathbb{R}^{n \times n}$, where $P = D^{-1}A$.
D^{-1}, A	Inverse degree matrix and adjacency matrix of the graph.
c	Damping factor of PageRank, commonly set to 0.85.
$h, h(t+1)$	Starting probability vector, $(n \times 1)$, with potential changes over time.
$f(v)$	Node-level anomaly scoring function.
$\hat{f}$	Summary statistic (e.g., mean or median) of the score function, $f(v), \forall v \in V$.
c_0	Threshold for detecting anomalies based on score deviations.
$\lvert f(v') - \hat{f} \rvert$	Absolute difference between the score, $f(v')$, and the summary statistic, $\hat{f}$.
N, n	Number of nodes and edges in $\mathcal{G}$.

Definition 4. *We define* ***dynamic graph,*** $\mathcal{G} = G_{t\,t=1}^{T}$, *as a sequence of ordered sets of graph snapshots at different timestamps, t, where T is the total time steps, and each* G_t *is a static graph representing the state of the graph at timestamp t.*

3.1. Personalized PageRank and RWR

The PageRank algorithm [13] assigns scores to web pages, treating the web as a directed graph with pages as nodes. Variants like Random Walk with Restarts (RWRs) [37] have been developed for specific tasks. The standard representation of the personalized PageRank vector, v_u [16], for the source node u is calculated as follows:

$$v_u = cP^T v_u + (1 - c)h, \tag{1}$$

where $P^T \in \mathbb{R}^{n \times n}$ is the column stochastic transition matrix, which can be further expanded as $P^T = D^{-1}A^T$. The parameter $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix, and $D \in \mathbb{R}^{n \times n}$ is the degree matrix of a given graph, $\mathcal{G}$, with n nodes. $\boldsymbol{h}$ is the indicator (restart) vector for node u (i.e., the teleport vector), and v_u represents the stationary probability distribution over all nodes, indicating the likelihood of being at any node in the graph after many random walk steps. The parameter $\boldsymbol{c}$ is the *damping factor*, representing the probability that the random walker follows an edge rather than teleporting, and it is commonly set to 0.85.

The personalized PageRank is commonly solved using power iteration and iterative refinement until the probability that a random walker navigates the graph converges to the personalized importance scores assigned to each node.

3.2. Dynamic PageRank Variant

Our method extends the Dynamic PageRank [9] variant to effectively handle real-time anomaly detection in graph streams by incorporating temporal adaptability for real-time updates. Traditional PageRank algorithms often fail to address challenges in dynamic

graphs, such as high-frequency updates, computational overhead, and the temporal decay of node relevance.

These properties make Dynamic PageRank particularly well-suited for dynamic graphs, where the current node importance score, v^{i+1}, can be updated incrementally from previous scores, v^i, based on recent structural changes in the graph. This adaptability is especially critical in applications like intrusion detection and critical cyber-infrastructure monitoring, where edge or node insertions and deletions often signify anomalies or potential cyber-threats [2], requiring immediate attention.

For example, consider a coordinated Distributed Denial-of-Service (DDoS) attack, where multiple computers collectively exhibit malicious behavior or a social network where a user's connections fluctuate daily. A standard PageRank algorithm treats the graph as static, missing short-term or evolving trends, such as a sudden influx of connections indicative of spam activity. Personalized Dynamic PageRank variants, like our proposed model, capture this behavior by incrementally updating scores with temporal decay (i.e., the gradual reduction in the influence of older node or edge interactions over time), thereby enabling timely anomaly detection. The formula to compute the dynamic PageRank score is shown below:

$$v_u(t+1) = cP^{(t)}v_u(t) + (1-c)h(t+1) \tag{2}$$

Equation (2) follows directly from Equation (1), where $P^{(t)}$ is the stochastic transition matrix, defined as $P^{(t)} = D^{-1}A^T$. This matrix depends on the graph's structure at each time step, t. The parameter c is the damping factor, and the indicator probability vector, $h(t+1)$, allows the teleport vector to vary over time.

This formulation enables incremental updates to the PageRank scores upon the insertion or deletion of edges, making it highly suitable for practical downstream tasks such as graph learning. Previous studies [9,23] have demonstrated that Dynamic PageRank performs effectively on graphs, establishing it as a robust scoring function for detecting node-level structural anomalies in dynamic graphs.

3.3. Problem Formulation

Before outlining our specific node-level problem, we first define a node-level structural anomaly in a dynamic graph.

Definition 5 (*Node-level Structural Anomaly*)**.** *Given a dynamic weighted graph, $\mathcal{G} = (V_t, E_t, W_t) = \{G_t\}_{t=0}^{T}$, consisting of the initial snapshot, G_0, where V_t is the set of nodes, E_t is the set of edges, and W_t is the set of edge weights at time t. The total node set is $V = \cup_{t=0}^{T} V_t$, and the total edge set is $E = \cup_{t=0}^{T} E_t$. Each snapshot can have changes in edge events, $|\Delta E_t| \geq 0$, and changes in node events, $|\Delta V_t| \geq 0$. Let $f : V \to \mathbb{R}$ be a specified scoring function. The set of anomalous nodes, $V' \subseteq V$, is defined such that $\forall v' \in V'$, $|f(v') - \hat{f}| > c_0$, where $\hat{f}$ is a summary statistic of the score $f(v), \forall v \in V$, and c_0 is a predefined threshold indicating a significant deviation.*

Problem 1. *Detect node-level structural anomalies in G_t. Specifically, a node, **u**, is considered anomalous at time **t** if it experiences a significant deviation characterized by a **sudden transition** in its connectivity, i.e., from its original set of outgoing edges to neighbors, $\langle v_1, \ldots, v_{\Delta n} \rangle$, to a new set, $\langle \hat{v}_1, \ldots, \hat{v}_{\Delta n} \rangle$.*

In Problem 1, Δ_n is defined as a substantial change in the set of a node's neighbors, $\langle v_1, \ldots, v_n \rangle$, as shown in Equation (3):

$$\Delta_n = N(u, t+1) \setminus N(u, t), \tag{3}$$

where $N(u,t)$ is the set of neighbors of node u at time t. A *sudden transition* occurs when $|\Delta_n|$ exceeds a predefined threshold, τ, indicating a notable structural shift.

Figure 2 illustrates node-structure anomalies across two graph snapshots. Sudden changes in node scores and the appearance of new edges could indicate potential cyber-attacks on computer networks or a series of fraudulent transactions. Intuitively, to detect such structural anomalies, our approach focuses on the existence of edges connecting nodes rather than the frequency of edge occurrences between them. Additionally, we consider the rapid occurrence of these edges (insertion or deletion) with fast adaptiveness and, ultimately, an optimized vector score for each node in the dynamic graph.

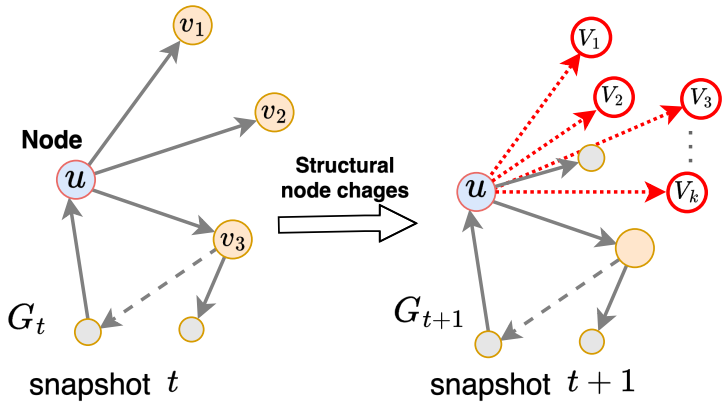

Figure 2. Node-level and structural anomalies across two snapshots (t and t + 1), with nodes and edges labeled to indicate changes and anomalies. These changes could represent cyber-attacks, such as DDoS, or bursts of fraudulent transactions in financial systems.

4. Proposed Method

In this section, we present our previous work, the DECAYRANK algorithm [38], along with our novel ADAPTIVE-DECAYRANK algorithm. These approaches incorporate a dynamic node scoring function, an adaptive decay factor enhanced with Bayesian inference, and an anomaly detection metric. The components are illustrated in Figure 3 and are discussed in detail in Section 4.2.

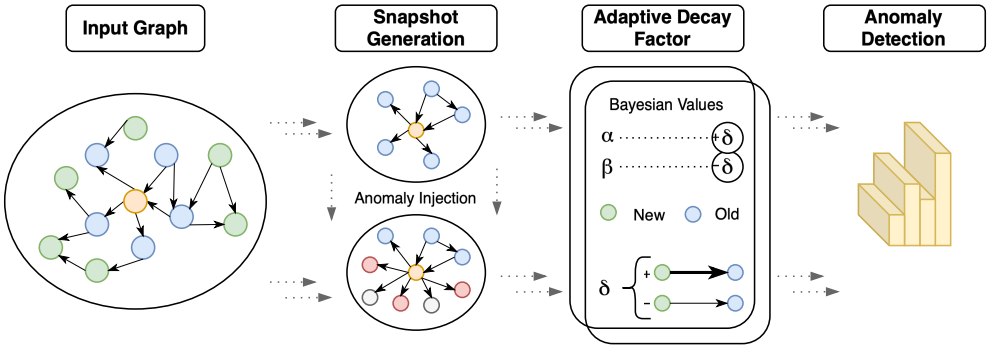

Figure 3. ADAPTIVE-DECAYRANK Framework: The framework consists of the *Graph Input and Snapshot Generation Phase*, where dynamic graphs are converted into snapshots; the *Dynamic Node Scoring Phase*, which computes node importance; the *Adaptive Bayesian Updating Phase*, where decay factors are adjusted based on graph changes; and the *Normalization and Anomaly Detection Phase*, which identifies nodes with significant deviations in real-time.

In Figure 3, the ADAPTIVE-DECAYRANK framework begins by transforming the dynamic graph input into snapshots based on timestamps, which serve as the foundation for node score computations. The algorithm then iteratively updates node scores while

dynamically adjusting the decay factor through Bayesian inference. This adaptive approach ensures sensitivity to observed changes in the graph structure, enabling the quick detection of evolving patterns. Finally, normalized anomaly detection metrics are applied to identify nodes exhibiting significant deviations over time. The framework emphasizes the integration of real-time updates, adaptive mechanisms, and robust anomaly detection tailored to dynamic graph environments.

4.1. Dynamic Node Scoring with Temporal Decay

Below, we provide a detailed explanation of the components of our proposed ADAPTIVE-DECAYRANK algorithm, including the node scoring function and the foundation for integrating an adaptive decay factor through Bayesian inference.

4.1.1. Node Scoring Function

Traditional offline scoring functions, such as HITS (Hyperlink-Induced Topic Search) [39] and centrality measures (e.g., Betweenness Centrality) [14], are effective at identifying degree centrality, influential nodes, or ego nodes in static graphs [2]. However, in an online setting, where graphs evolve over time, capturing dynamic fluctuations becomes essential.

In our previous work, the DECAYRANK algorithm [38] was introduced to assign importance scores to individual nodes by considering neighboring edges and evolving graph properties. Based on the iterative formula in Equation (2), DECAYRANK incorporates a **fixed decay factor** to dynamically adjust scores at every iteration. While effective, this fixed decay factor posed limitations in adapting to varying graph dynamics, an issue addressed by the proposed ADAPTIVE-DECAYRANK algorithm. The modified node score vector, v_u, for DECAYRANK is defined as

$$v_u = c\hat{P}v_u + (1-c)h_u e^{-\delta\cdot(t-A_u^{(T)})} \tag{4}$$

Here, $\hat{P}$ is the row-normalized adjacency matrix, and the other parameters follow the notation in Equation (2). The temporal decay function, $e^{-\delta\cdot(t-A_u^{(T)})}$, reduces the influence of older interactions, with the decay factor, δ, controlling the rate of decay. The term $\boldsymbol{A_u^{(T)}}$ represents the most recent timestamp when node u was last updated in the adjacency structure of the graph. Specifically, it is derived from the adjacency structure, $\boldsymbol{A_u}$, which maintains historical interaction records of node u. This ensures that more recent interactions contribute more significantly to the node score, while older interactions decay exponentially over time.

4.1.2. Temporal Decay Factor

The concept of our proposed novel *adaptive-decay factor* formula is inspired by the work on *"Weight Decay Regularization in the ADAM Optimizer"* [10], where the authors propose decoupling gradient-based weight updates in both the SGD [11] and Adam [12] optimizers.

In Equation (4), we introduced the temporal decay function $\boldsymbol{e^{-\delta\cdot(t-A_u^{(T)})}}$, which enhances the adaptability and responsiveness of our algorithm. The parameter $\boldsymbol{\delta}$, referred to as the ***decay factor***, controls the rate at which the influence of an edge or node decreases over time. A higher $\boldsymbol{\delta}$ value results in faster decay, gradually reducing the impact of older interactions or edges. This ensures that older data have progressively less influence on the computation of new graph snapshots and allows the algorithm to prioritize recent and relevant changes in the graph structure. The variable $\boldsymbol{t}$ represents the current timestamp, while $\boldsymbol{A_u^{(T)}}$ denotes the timestamp attribute of node $\boldsymbol{u}$. The term $\boldsymbol{(t - A_u^{(T)})}$ calculates the time difference between the current timestamp and the last update of node $\boldsymbol{u}$.

Algorithm 1 provides a summary of our previous work on the DECAYRANK algorithm, which employs a **fixed decay factor**. While effective, this fixed factor struggles to adapt to the varying dynamics of individual nodes, potentially resulting in suboptimal anomaly detection and reduced sensitivity to recent changes in the graph.

To overcome these limitations, we propose the ADAPTIVE-DECAYRANK algorithm, which utilizes a Bayesian updating mechanism to dynamically adjust the decay factor in response to observed changes in node rank scores. This adaptive approach not only enhances the algorithm's responsiveness to evolving graph structures but also significantly improves the accuracy and robustness of anomaly detection in dynamic and near real-time graph environments.

Algorithm 1 DECAYRANK: Streaming Anomaly Scoring

Input: A: Array of dynamic graph outEdges over time
Output: v: Updated PageRank scores
Initialization:
Initialize PageRank values $v[i] \leftarrow \frac{c}{n}$ for all nodes i
while each timestep *step* from 0 to $numSteps - 1$ **do**
▷ **Power Iteration:**
    Update scores based on graph structure.
▷ **Decay Factor Application:**
    Apply decay factor δ to adjust scores over time.
▷ **Update PageRank Scores:**
    Integrate new scores: $v[i] \mathrel{+}= nq[i]$
end while
▷ **Normalization Anomaly Score:**
return $v \leftarrow$ normalizeNodeScore(v, n)

4.2. ADAPTIVE-DECAYRANK *Algorithm with Bayesian Updating*

Next, we describe our ADAPTIVE-DECAYRANK algorithm, which incorporates a dynamic scoring function with *adaptive decay factors* for responsive updates based on graph changes, along with a Bayesian updating mechanism for further refinement. The ADAPTIVE-DECAYRANK algorithm extends our previous DECAYRANK approach. This innovation enables the algorithm to dynamically adapt to real-time changes in graph structures, ensuring robustness and accurate anomaly detection in dynamic graph settings.

4.2.1. Bayesian Inference and Updating

The ADAPTIVE-DECAYRANK algorithm utilizes Bayesian inference [40] to dynamically adjust decay factors, enabling responsive updates to evolving graph structures. Unlike frequency-based approaches, which rely on fixed parameters and repeated sampling, Bayesian inference treats probability as a measure of belief, updating prior distributions in light of new data and observed changes. This makes it well-suited for dynamic, evolving systems such as graphs.

Bayesian inference involves three key steps: **(1) The prior distribution,** $\pi(\theta)$, which represents initial beliefs about the parameter θ (in this case, the *decay factor* $\boldsymbol{\delta_u}$) before observing any data. **(2) The likelihood function,** $L(\theta)$, expressed as $p(X|\theta)$, captures the probability of observing data, X, given a specific value of θ. This function reflects how well θ explains the observed data. Finally, **(3) the posterior distribution,** $\pi(\theta|X)$, which combines the prior and likelihood to update our beliefs about θ after observing data X.

Using Bayes' theorem, the posterior is defined as

$$\pi(\theta|X) = \frac{L(\theta)\pi(\theta)}{\int L(\theta)\pi(\theta)d\theta}, \tag{5}$$

where $\pi(\theta|X)$ is the posterior distribution (updated belief about θ), $L(\theta)$ is the likelihood function defined as $L(\theta) = \prod_{i=1}^{n} p(X_i|\theta)$, and $\int L(\theta)\pi(\theta)d\theta$ is the normalizing constant ensuring the posterior is a valid probability distribution.

4.2.2. Bayesian Updating in ADAPTIVE-DECAYRANK

In the context of ADAPTIVE-DECAYRANK, Bayesian inference is employed to update the decay factor, $\boldsymbol{\delta_u}$, for each node, $\boldsymbol{u}$. The Gamma distribution is used as a conjugate prior, ensuring computational efficiency and tractability during updates on the observed changes in node scores. The probability density function of the **prior Gamma distribution** for δ_u is defined as

$$f(\delta_u \mid \alpha_u, \beta_u) = \frac{\beta_u^{\alpha_u}}{\Gamma(\alpha_u)} \delta_u^{\alpha_u - 1} e^{-\beta_u \delta_u}, \tag{6}$$

where $\boldsymbol{\delta_u}$ is a continuous variable taking non-negative values, $(\delta_u \geq 0)$, $\boldsymbol{\alpha_u}$ is the *shape parameter* that controls the shape of the distribution, $\boldsymbol{\beta_u}$ is the *rate parameter* that controls the scale of the distribution, and $\boldsymbol{\Gamma(\alpha_u)}$ is the Gamma function, which generalizes the factorial function.

4.2.3. Adaptive and Dynamic Scoring Function

The ADAPTIVE-DECAYRANK formula with Bayesian updating is defined in Equation (7). The Gamma distribution parameters $\boldsymbol{\alpha_u}$ and $\boldsymbol{\beta_u}$ in Equation (6) represent our prior beliefs about the decay factors. For each iteration, the algorithm observes and adjusts node scores based on the recent changes. The scoring function is given by

$$v_u^{(k+1)} = c\hat{P}v_u^{(k)} + (1-c)h_u e^{-\delta_u (k - A_u^{(T)})}, \tag{7}$$

where $\boldsymbol{v_u^{(k+1)}}$ is the updated score for node u at iteration $k+1$, $\boldsymbol{c}$ is the damping factor controlling the influence of neighboring nodes, $\hat{\boldsymbol{P}}$ is the transition probability matrix, $\boldsymbol{v_u^{(k)}}$ is the node score for $\boldsymbol{u}$ at iteration $\boldsymbol{k}$, $\boldsymbol{h_u}$ is the personalization vector capturing the inherent importance of node u, $\boldsymbol{\delta_u}$ is the decay factor for node u, dynamically updated using Bayesian inference, and $(k - A_u^{(T)})$ is the time elapsed since the last update of node u.

In Equation (7), the term $c\hat{P}v_u^{(k)}$ represents dynamic PageRank propagation, and the adaptive temporal decay is represented with the expression $(1-c)h_u e^{-\delta_u (k - A_u^{(T)})}$, emphasizing recent interactions and diminishing the influence of older data.

Posterior updates: Equation (6) shows the prior distribution for δ_u; that is, the conjugate prior, ensuring efficient computation. The *adaptive decay factor*, $\boldsymbol{\delta_u}$, is computed as the mean of the posterior Gamma distribution and is updated using α_u and β_u:

$$\delta_u = \frac{\alpha_u}{\beta_u}. \tag{8}$$

The *shape parameter* $(\boldsymbol{\alpha_u})$ is adjusted to reflect the magnitude of observed changes, with larger changes resulting in fast updates.

$$\alpha_u^{(t+1)} = \alpha_u^{(t)} + \left| v_u^{(k+1)} - v_u^{(k)} \right|. \tag{9}$$

The *rate parameter* $(\boldsymbol{\beta_u})$ is updated to maintain stability, preventing drastic shifts when changes are minimal:

$$\beta_u^{(t+1)} = \beta_u^{(t)} + \left(1 - \left| v_u^{(k+1)} - v_u^{(k)} \right|\right). \tag{10}$$

By leveraging Bayesian inference, the ADAPTIVE-DECAYRANK algorithm dynamically adjusts $\boldsymbol{\delta_u}$ in real time, ensuring sensitivity to significant changes in graph structure while mitigating noise. The conjugacy of the Gamma distribution guarantees computational

efficiency, enabling scalability for large-scale dynamic graphs. This approach addresses concerns about innovation, clarity, and robustness in anomaly detection for dynamic graphs.

4.3. Algorithm

Algorithm 2 implements the ADAPTIVE-DECAYRANK framework. The algorithm processes a dynamic graph represented by the **input,** $\boldsymbol{A}$, which is an array of outEdges serving as the transition matrix. Each node maintains a list of outgoing edges, denoted as $\boldsymbol{A[i].out} = \{j_1, j_2, \ldots, j_k\}$, with corresponding weights, $\boldsymbol{A[i].weight} = \{w_1, w_2, \ldots, w_k\}$, where w_k is the weight associated with the edge connecting to each neighboring node, j_k. Additionally, the last observed timestamp (denoted as $\boldsymbol{A[i].timestamp}$ in algorithm line 5) represents the last time node i was updated in the graph stream. In Equation (7), this timestamp corresponds to $A_u^{(T)}$, which represents the last time step at which node u was modified. For example, if node A has outgoing edges to nodes B and C with weights 0.5 and 0.2, respectively, and the timestamp is 10, this implies that node A's connections were last updated at time step 10. The edge weights are assigned dynamically based on edge updates in the evolving graph stream, while the adaptive decay factor, $\boldsymbol{\delta}$, reduces edge influence over time.

Algorithm 2 ADAPTIVE-DECAYRANK: Node Anomaly Scoring

Input: $\boldsymbol{A}$: Array of dynamic graph outEdges, $\boldsymbol{v}$: Array of PageRank scores, $\boldsymbol{n}$: Number of nodes, $\boldsymbol{numSteps}$: Streaming timesteps
Output: $\boldsymbol{v}$: Anomaly scores $\{v^{(0)}, \ldots, v^{(T)}\}$
▷ **Initialization of Parameters** (δ, α, β):
1: Initialize DecayRank values $v[i] \leftarrow \frac{c}{n}$ for all nodes i
2: **while** each timestep *step* from 0 to $numSteps - 1$ **do**
▷ **Power Iteration:**
3: Update scores based on graph structure.
▷ **Applying Adaptive Decay Factor:**
4: **for** $i = 0$ to $n - 1$ **do**
5: time_diff $\leftarrow$ step $- A[i]$.timestamp
6: observed_changes $\leftarrow$ (new_nq$[i]$ $-$ old_nq$[i]$)
7: $\alpha[i]$ += o
8: $\beta[i]$ += $(1 - o)$
9: $\delta[i] \leftarrow \frac{\Gamma(\alpha[i])}{\Gamma(\beta[i])}$
10: Adaptive_Decay$[i] \leftarrow \min(\delta[i], 1.0)$
11: new_nq$[i]$ *= $\exp(-\delta[i] \times$ time_diff$)$
12: **end for**
▷ **Dynamic Updating of DecayRank Scores:**
13: **for** $i = 0$ to $n - 1$ **do**
14: $v[i]$ += new_nq$[i]$
15: **end for**
▷ **Convergence Check:**
16: **if** $\sum_{i=0}^{n-1}$(new_nq$[i]$ $-$ old_nq$[i]$) $<$ EPSILON **then**
17: **break**
18: **end if**
19: **end while**
▷ **Normalization:**
20: **return** $v \leftarrow$ normalizeDecayRank(v, n)

The **output**, $\boldsymbol{v}$, is the sequence of stochastic vectors for node scores. The algorithm initializes node scores uniformly as $v[i] = c/n$, where c is the damping factor, and n is the total number of nodes (as shown in *line 1*). At each time step, the algorithm iteratively updates these scores based on the structure of $\boldsymbol{A}$ using power iteration (*lines 2–3*).

The **adaptive-decay factor** $\delta[i]$ (as shown in lines 7–9) is dynamically adjusted using Bayesian inference to reflect observed changes in the graph structure, enhancing sensitivity to recent interactions while mitigating noise. The decay factor is computed as $\delta[i]$ in *line 9*, where $\alpha[i]$ represents the *prior update*, and $\beta[i]$ represents the *posterior update*, both of which are updated based on observed changes (see details in Section 4.2.3). The scores, $\boldsymbol{v[i]}$, (updated in *line 14*) are adjusted with the computed decay values ($\boldsymbol{new_n q[i]}$), and the process continues until the **convergence criterion** (*line 16–19*) is satisfied. In *line 16* , the EPSILON is set to 0.0001 to balance accuracy and computational efficiency. After completing all iterations, the scores are **normalized** as $v[i]$ to maintain consistency across nodes and ensure comparability, preventing high-degree nodes from dominating anomaly detection results (see details in Section 4.4.1).

4.4. Anomaly Detection Metrics

After Algorithm 1 computes the score vectors for each node using an **adaptive decay factor**, we apply similarity metrics from the work of Yoon et al. **[20]** to detect anomalous behavior at each time step in the graph. The base parameter for the *Adaptive Decay Factor*, $\boldsymbol{\delta_u}$, has an empirical range of $[0.25, 0.55]$. For stability, the tuning occurs dynamically using *Bayesian updating*, where a higher decay factor reduces the influence of past edge interactions. The anomaly detection metric was first introduced and theoretically analyzed in **[20]** for tracking node score variations over time. Given a score vector, $\boldsymbol{v_u}$, the metric is defined as

$$v''_u = \frac{(v_u(t+\Delta t) - v_u(t)) - (v_u(t) - v_u(t-\Delta t))}{\Delta t^2} \tag{11}$$

In Equation (11), the metric $\boldsymbol{v''_u}$ represents the *discretization of the second-order derivative* of the score vector, $\boldsymbol{v_u}$. The second derivative captures *acceleration or deceleration* in score changes over time, making it effective for identifying sudden shifts in node behavior that may indicate anomalies. The discretization process approximates a **continuous function** in a discrete form, ensuring computational efficiency and robustness in time-evolving graphs.

The **theoretical justification** is provided in [20], where the authors established a basis for anomaly detection in dynamic graphs by modeling normal graph behavior using *Lipschitz continuity*, ensuring smoothness in time-series updates. The study demonstrates that normal graph streams exhibit bounded *first-* and *second*-order derivatives: $\|v'_u\|_1 \leq K_1, \quad \|v''_u\|_1 \leq K_2$, where K_1 and K_2 are positive real constants. Under this assumption, normal graphs exhibit gradual changes, leading to small values of $\|v'_u\|_1$ and $\|v''_u\|_1$. However, anomaly types, such as sudden structural changes, could cause these bounds to increase significantly, making them strong indicators of anomalous behavior (*see Section 4.3* in [20] for more details and proofs).

To maintain consistency across time and balance positive and negative fluctuations, we apply **centering and variance scaling**, ensuring that anomaly scores remain comparable across different time steps.

4.4.1. Online Normalization

To ensure that the derivative-based metrics are robust and consistent across time steps, we normalize the ADAPTIVE-DECAYRANK scores, $\boldsymbol{v_s}$, and their derivatives online by adjusting the mean, $\boldsymbol{\mu}$, and variance, $\boldsymbol{\sigma^2}$, of the scores, with the inclusion of the *adaptive decay factor* $\boldsymbol{\delta_u}$ to emphasize recent changes:

$$\mu_{new}[i] = \frac{t}{t+1} \times \mu[i] + \frac{1}{t+1} \times \boldsymbol{v}_s[i] \tag{12}$$

$$\sigma^2_{new}[i] = \frac{t}{t+1} \times \sigma^2[i] + \frac{1}{t+1} \times (\boldsymbol{v}_s[i])^2 \tag{13}$$

Following Equations (9) and (10), the Bayesian update prior values are set to $\alpha = 1.0$ and $\beta = 1.0$. These parameters control the rate of adaptation to changes in graph structures. We set the number of iterations for *Bayesian updating* to *numSteps = 10*, ensuring stable convergence while capturing micro-level changes in node importance. This normalization centers the derivatives around zero and scales them to unit variance, balancing positive and negative fluctuations as well as large and small variations, thereby making anomaly comparisons across different time steps more consistent.

4.4.2. Anomaly Score Calculation

Finally, the anomaly score is computed by combining the effects of both first- and second-order derivatives:

$$\boldsymbol{v_s} = \max_i \left(\sum_{j=0}^{n-1} |d[i][j]| \times \frac{\text{total\_max}}{\max[i]} \right) \tag{14}$$

where $d[i][j]$ is the derivatives for node i in step j, *total_max* is the product of the maximum derivative values across all nodes, and $max[i]$ is the maximum observed derivative for node i.

To further illustrate the computation of Equation (14), we consider a small dynamic graph with **three nodes**, where each **row** in the $d-$matrix represents a node, and each column represents the derivatives of a node across three timestamps, as shown below:

$$d = \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ 0.8 & 0.6 & 0.7 \\ 0.1 & 0.4 & 0.2 \end{bmatrix}$$

From the matrix, d, the maximum derivative per node (row_1, row_2, row_3) is $\max = \{0.5, 0.8, 0.4\}$, and the total maximum is **0.8**.

For **Node 1**, the anomaly score computed as

$$\begin{aligned} v_s(1) &= \sum_{j=0}^{2} |d[1][j]| \times \frac{\text{total\_max}}{\max[1]} \\ &= (0.2 + 0.5 + 0.3) \times \frac{0.8}{0.5} = \mathbf{1.6} \end{aligned}$$

Similarly, for **Node 2**,

$$v_s(2) = (0.8 + 0.6 + 0.7) \times \frac{0.8}{0.8} = \mathbf{2.1}$$

and for **Node 3**,

$$v_s(3) = (0.1 + 0.4 + 0.2) \times \frac{0.8}{0.4} = \mathbf{1.4}$$

Finally, the anomaly score is determined as follows:

$$\boldsymbol{v_s} = \max(1.6, 2.1, 1.4) = \mathbf{2.1}$$

This example illustrates how the anomaly score prioritizes nodes with significant derivative changes, emphasizing abnormal patterns in the dynamic graph. The final score, $\boldsymbol{v_s}$, reflects the anomalous behavior of the node relative to its past and to other nodes.

5. Experiments

In this section, we evaluate the performance of ADAPTIVE-DECAYRANK in comparison to three state-of-the-art baselines in terms of accuracy, scalability, and speed.

5.1. Datasets

We utilize three datasets—two real-world datasets and one synthetically generated dataset—to evaluate the robustness, adaptability, and scalability of ADAPTIVE-DECAYRANK under diverse graph structures and anomaly densities. These datasets were chosen to represent different types of dynamic graphs, including network intrusion detection, botnet traffic, and synthetic evolving networks with controlled anomaly injection. A summary of dataset attributes is provided in Table 3.

DARPA: DARPA [41] contains $4.5M$ *IP-IP* communications between $9.4K$ source IPs and $23.3K$ destination IPs over $87.7K$ timestamps, with each node representing an IP address, each edge representing network traffic, and each *IP-IP* communication represented as a directed edge, $(\text{timestamp, srcIP, destIP, attack})$. Anomalies in this dataset primarily consist of network attacks, including Denial-of-Service (DoS), remote-to-local (R2L), user-to-root (U2R), and probing (scanning) activities [41]. These anomalies were labeled using ground-truth attack logs provided in the dataset, where each attack instance is *timestamped*, allowing for an evaluation of how well the algorithms detect real-time intrusion attempts.

CTU-13: The CTU-13 dataset [42] consists of 13 scenarios capturing botnet traffic mixed with normal background traffic. It includes approximately **100** million *IP-IP* communications, where nodes represent IP addresses, and directed edges signify network flows. Anomalies in CTU-13 correspond to botnet activity, which involves infected machines communicating with command-and-control (C&C) servers, scanning targets, and launching attacks [43]. The dataset's PCAP (packet capture) files were processed to extract bidirectional NetFlows (*timestamp, srcIP, destIP, and label*). We classify botnet-related connections as anomalies, leveraging the dataset's ground-truth labels for evaluation.

RTM Synthetic Graph: The RTM dataset was synthetically generated using the Recursive Tensor Model (RTM) [44], where a Kronecker graph generation process is used to simulate realistic time-evolving graph structures. The generated dense graph contains 512 nodes and approximately 19.6 million edges across 1000 timestamps. To assess the robustness of our approach, we inject anomalies at two levels: a **10**% **injection** of structural node anomalies, disrupting the normal connectivity pattern, and a **30**% **injection** of a higher concentration of anomalies to test the model's robustness against varying anomaly densities and structural changes in dynamic graphs. The dataset statistics, including the number of nodes, edges, timespans, snapshots generated, and anomaly types, are summarized in Table 3.

Table 3. Dataset statistics and characteristics. Graphs are read and processed as directed, with snapshots $|\mathcal{G}_{0,\ldots,T}|$ generated over time. Anomalies are detected after the specified initial snapshot (*Init.* t).

Dataset	Nodes ($\lvert V\rvert$)	Edges ($\lvert E\rvert$)	Timespan	Snapshots ($\lvert\mathcal{G}_{0,\ldots,T}\rvert$)	Anomalies
DARPA	33,221	4,616,321	2 months	1463 (256 *init*)	Network traffic attacks
CTU-13	256	1,416,971	7 days	301 (50 *init*)	Botnet-related
RTM Synthetic	512	19,683,000	1000 timestamps	101 (10 *init*)	Injected (10% and 30%)

5.2. Experimental Setup

We implemented ADAPTIVE-DECAYRANK in C++ and ran experiments on a MacOS® with 3.2 GHz Intel 8-Core™ 64-bit processor running at 3.2 GHz, a 7-Core™ GPU, 16-Core™ Neural Engine, and 8 GB unified memory (RAM).

We compared ADAPTIVE-DECAYRANK to four state-of-the-art distance-based methods for anomaly detection: **SEDANSPOT** [18], **ANOMRANK** [20], **DYNANOM** [23], and **DE-**

CAYRANK [38]. Our approach extends DECAYRANK and is closely related to ANOMRANK; however, we introduce a more robust and dynamic scoring function incorporating an adaptive decay factor and Bayesian updating, enabling faster detection of micro-changes at each timestamp. The average precision and algorithm runtimes are presented in Table 4. The precision-recall curve for the top-k (50–800) anomalous snapshots is shown in Figure 4, the AUC-ROC for DARPA is depicted in Figure 5, and the scalability analysis is illustrated in Figure 6. However, due to constraints in algorithmic design and dataset compatibility, we were unable to evaluate SEDANSPOT and DYNANOM on the CTU-13 and RTM datasets.

Table 4. Average precision and runtime for different datasets. (**Note:** The reported average precision (Table 4) is **threshold-dependent**, computed as the mean of precision scores across multiple top-k thresholds **(Top-50, 100, ..., 800)**. Precision values for the **Top-50 to Top-500** thresholds are presented in Table 5. For a more robust, **threshold-free** evaluation, we also report the **precision-recall (PR) curve** (Figure 4) and the **Area Under the Curve (AUC)** (Figure 5), which provide a more comprehensive assessment of model performance).

Algorithms	DARPA	Runtime (s)	CTU-13	Runtime (s)	RTM-10	Runtime (s)	RTM-30	Runtime (s)
SEDANSPOT	0.56	480	-	–	-	–	-	–
DYNANOM	0.5840	379.334	-	–	-	–	-	–
ANOMRANK	0.5665	4.02551	0.4118	0.690561	0.3125	11.9526	0.3095	12.529
DECAYRANK ($\delta = 0.65$)	0.572	5.4555	0.6425	0.6265	0.5865	0.995	0.5699	1.09206
ADAPTIVE-DECAYRANK	**0.64554**	35.0083	**0.6856**	0.7040	**0.6746**	1.2231	**0.6976**	1.23785

Note: The average precision is computed as the mean of all precision scores across multiple thresholds (Top-k anomalies, where $k = 50, 100, 150, \ldots, 800$), as detailed in Table 5 and Figure 4. **DECAYRANK**: improves with a high decay factor, δ, of **0.65** to **0.95**.

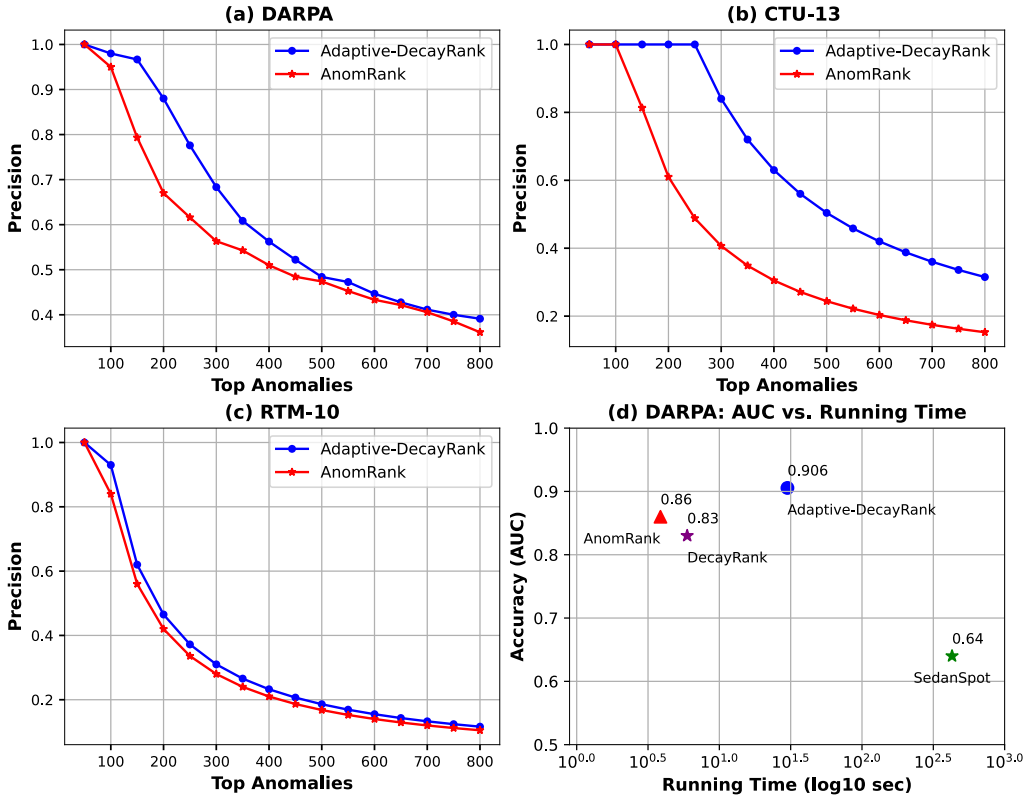

Figure 4. Precision-recall curves of ADAPTIVE-DECAYRANK vs. ANOMRANK across three datasets for Top 50–800 anomalies. (**a**) Precision-recall curve for the DARPA dataset, (**b**) Precision-recall curve for the CTU-13 dataset, (**c**) Precision-recall curve for the RTM dataset, and (**d**) AUC vs. speed (Running Time) on the DARPA dataset. The Area Under the Curve (AUC) quantifies the model's ability to distinguish between normal and anomalous patterns, where a higher AUC indicates better overall detection performance.

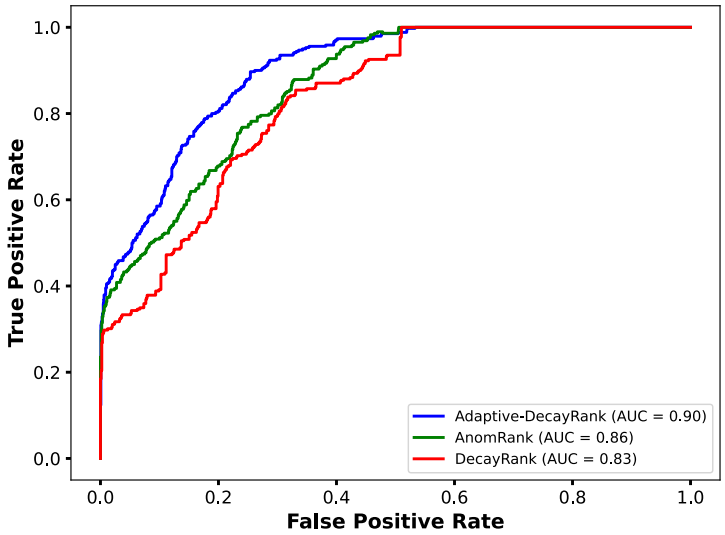

Figure 5. AUC-ROC for DARPA dataset: A comparison of ADAPTIVE-DECAYRANK (**0.90**), ANOMRANK (0.86), and DECAYRANK (0.83), showcasing the superior anomaly detection performance of ADAPTIVE-DECAYRANK.

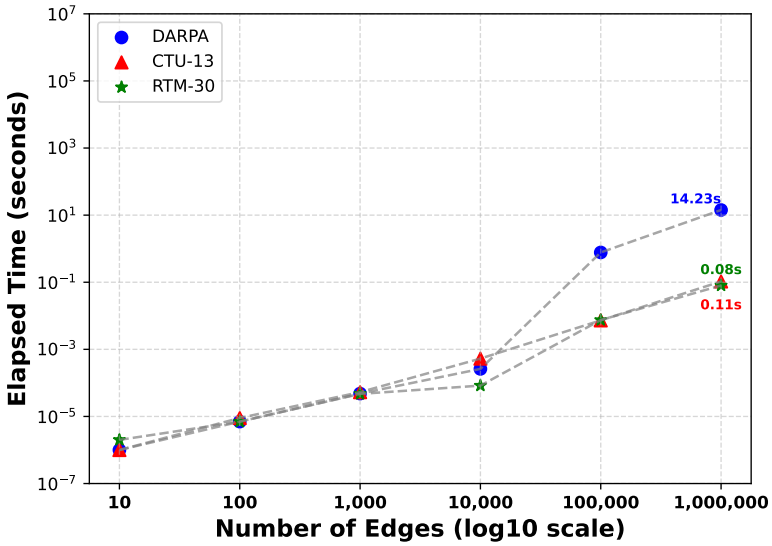

Figure 6. ADAPTIVE-DECAYRANK scalability analysis on DARPA, CTU-13, and RTM-30 datasets.

Table 5. Precision of Top 50–500 anomalies of **ADAPTIVE-DECAYRANK** vs. **ANOMRANK** across DARPA, CTU, and RTM datasets.

Anomalies	DARPA	CTU-13	RTM-30
	ADAPTIVE-DECAYRANK		
Top-50	1.000	1.000	1.000
Top-100	0.980	1.000	1.000
Top-10	0.980	1.000	1.000
Top-200	0.880	1.000	1.000
Top-300	0.683	0.907	0.880
Top-400	0.563	0.680	0.660
Top-500	0.520	0.544	0.528
	ANOMRANK		
Top-50	1.000	1.000	1.000
Top-100	0.950	0.980	0.840
Top-200	0.670	0.610	0.420
Top-300	0.563	0.407	0.280
Top-400	0.510	0.305	0.210
Top-500	0.474	0.244	0.168

5.3. Accuracy and Speed

5.3.1. Precision and Recall

Using each dataset, we evaluated the performance of ADAPTIVE-DECAYRANK using the precision-recall metric. First, we computed the anomaly score for all graph snapshots

in each dataset and calculated the top-k most anomalous snapshots ($k = 50, 100, \ldots, 800$). As shown in Table 3, DARPA has 1463 snapshots, CTU-13 has 301 snapshots, and RTM has 101 snapshots.

In Table 4, ADAPTIVE-DECAYRANK achieves a high average precision score of **0.6455** on **DARPA**, compared to the baselines: ANOMRANK with 0.5665, DECAYRANK with 0.572, DYNANOM with 0.58, and SEDANSPOT with 0.56. This represents a **13.94**% improvement over ANOMRANK, **12.86**% over DECAYRANK, **11.28**% over DYNANOM, and **15.27**% over SEDANSPOT. Additionally, our model's total runtime is **35** s faster than both SEDANSPOT and DYNANOM. However, ANOMRANK and DECAYRANK have runtimes of 4.03 and 5.46 s, respectively, which are faster than our method. Despite this, our model outperforms the baseline models in both precision and AUC, processing 1 million edges in 14.23 s (more details are provided in Section 5.4). This result is understandable given our iterative implementation of the decay factor with Bayesian updating at each time step.

Likewise, on **CTU-13**, ADAPTIVE-DECAYRANK achieves a high average precision of **0.6856** compared to ANOMRANK's 0.4118 and DECAYRANK's 0.6425. This represents a **66.49**% improvement over ANOMRANK and **6.71**% over DECAYRANK. On the **RTM-10** synthetic dataset (with 10% anomalies injected), ADAPTIVE-DECAYRANK achieves **0.6746** compared to DECAYRANK's 0.5865 and ANOMRANK's 0.31. This represents a **117.61**% improvement over ANOMRANK and **15.02**% over DECAYRANK. Similarly, on the **RTM-30** dataset (i.e., with 30% anomalies injected), ADAPTIVE-DECAYRANK achieves **0.6976** compared to ANOMRANK's 0.30 and DECAYRANK's 0.5699. This represents a **132.53**% improvement over ANOMRANK and **22.41**% over DECAYRANK. These results further demonstrate that our model is robust for detecting anomalies in large influx edge streams (due to algorithm and dataset compatibility constraints, we were unable to implement SEDANSPOT and DYNANOM on the CTU-13 and RTM datasets.)

5.3.2. Precision-Recall Curve

In Figure 4a–c, ADAPTIVE-DECAYRANK demonstrates higher precision and recall compared to ANOMRANK for all three datasets. It consistently achieves higher precision on various anomalous snapshots ($top50, 100, 250, \ldots, 800$). This indicates that ADAPTIVE-DECAYRANK effectively identifies anomalies while maintaining a low false positive rate. This is crucial in real-world scenarios, such as the power grid, fraud detection, and cyber-threats, considering that anomalous instances are associated with a significant influx of unusual patterns.

5.3.3. AUC-ROC

Figure 5 presents the ROC curves for ADAPTIVE-DECAYRANK, ANOMRANK, and DECAYRANK on the DARPA dataset. The ROC curve illustrates the trade-off between the *true positive rate (TPR)* and the *false positive rate (FPR)* at various thresholds. ADAPTIVE DECAYRANK achieves the highest AUC score of **0.90**, indicating its superior ability to distinguish between normal and anomalous patterns. This represents a significant improvement over ANOMRANK (AUC: 0.86) and DECAYRANK (AUC: 0.83), demonstrating the effectiveness of the adaptive decay factor and Bayesian updating mechanism employed by ADAPTIVE-DECAYRANK, particularly in capturing sudden changes in dynamic graph structures. Meanwhile, Figure 4d plots accuracy (AUC) vs. running time (log scale, in seconds, excluding I/O), further showcasing ADAPTIVE-DECAYRANK's efficiency and scalability compared to the baselines, including SEDANSPOT, which achieves an AUC of 0.64.

This represents a **4.65**% improvement in AUC over ANOMRANK and an **8.43**% improvement over DECAYRANK. Additionally, compared to SEDANSPOT (AUC: 0.64), ADAPTIVE-DECAYRANK shows a **40.63**% improvement, further highlighting its superior performance

in anomaly detection. In [23], the AUC score for DynAnom was not implemented; hence, it is missing from our Figure 5.

5.4. Scalability and Robustness

5.4.1. Robustness Across Different k

In Figure 4, we observe that at $k \geq 250$, Adaptive-DecayRank consistently outperforms baseline methods in terms of precision, demonstrating its robustness as the number of detected anomalies increases. Table 5 further highlights this, showing that Adaptive-DecayRank achieves *perfect precision* (1.000) for **Top-**50 anomalies across all datasets. Even at **Top-**100 and **Top-**200, its precision remains notably high, outperforming AnomRank, which drops significantly, especially on **RTM-30**. This suggests that Adaptive-DecayRank is more effective at identifying the most critical anomalies while maintaining a higher precision threshold.

As $\boldsymbol{k}$ increases, the performance gap between the two methods becomes more evident. At **Top-**300 anomalies, Adaptive-DecayRank maintains **0.683** on DARPA, **0.907** on CTU-13, and **0.880** on RTM-30, whereas AnomRank struggles at 0.563, 0.407, and 0.280, respectively. At **Top-**500, the difference becomes even more pronounced, particularly in CTU-13 and RTM-30, where Adaptive-DecayRank maintains a robust detection performance compared to the declining precision of AnomRank. These results demonstrate our model's superior ability to detect micro-changes and structural anomalies across diverse real-world dynamic graph datasets while preserving high detection quality, even as the number of anomalies increases.

5.4.2. Scalability

Figure 6 shows how well Adaptive-DecayRank scales with increasing edges across four datasets. We plot the wall-clock time required to process the first $2^1, 2^2, \ldots, 2^{22}$ edges of each dataset. On **DARPA**, which contains $4.6M$ edges and 1463 snapshots, the elapsed time for *small inputs* $(10, 100)$ remains in ***microseconds***, demonstrating Adaptive-DecayRank's efficiency for small graphs. At *medium sizes* (1000–10,000), the elapsed time remains relatively low (***milliseconds***), confirming its ability to handle moderate-sized graphs effectively. However, for *larger inputs* (100,000–1,000,000), the elapsed time increases slightly from **0.77** to **14.23** s, indicating higher computational requirements for large-scale graph updates. On **CTU-13**, which contains $1.4M$ edges and 301 snapshots, the algorithm processes small graphs efficiently (*microseconds* for $10, 100$), and even for the largest input size (1,000,000), it completes in **0.1057** s, reinforcing its scalability for real-time applications. On **RTM-30**, which includes $19M$ edges and 101 snapshots, processing times remain efficient, with **0.08** s required for 1,000,000 edges, demonstrating near real-time anomaly detection performance. Across datasets, Adaptive-DecayRank scales efficiently, maintaining *microsecond-level* times for small graphs, *millisecond-level* times for moderate-sized graphs, and handling *larger graphs in seconds*. Future work will explore how *graph structure* (e.g., density, connectivity) influences runtime performance.

6. Discussion

In this section, we analyze the key findings from our experiments, comparing Adaptive-DecayRank with state-of-the-art methods and explaining its superior performance in detecting anomalies in dynamic graph streams.

6.1. Comparison with State-of-the-Art Methods

6.1.1. Detection Accuracy

ADAPTIVE-DECAYRANK consistently achieves higher detection accuracy across all datasets. On the **DARPA** dataset, it attains an average precision of **0.6455**, outperforming ANOMRANK (**0.5665**), DECAYRANK (**0.572**), DYNANOM (**0.5840**), and SEDANSPOT (**0.56**). As shown in Table 4, this represents a notable **13.94%** improvement over ANOMRANK and a **12.86%** gain over DECAYRANK. Similarly, on the **CTU-13** dataset, ADAPTIVE-DECAYRANK achieves an even higher precision of **0.6856**, improving over ANOMRANK (**0.4118**) by **66.49%** and DECAYRANK (**0.6425**) by **6.71%**.

For the synthetic **RTM-10** dataset, which contains **10%** injected anomalies, our model achieves a precision of **0.6746**, significantly outperforming ANOMRANK (**0.31**) by **117.61%** and DECAYRANK (**0.5865**) by **15.02%**. The results are even more pronounced in the **RTM-30** dataset, where **30%** of nodes are anomalous. Here, our approach achieves **0.6976**, surpassing ANOMRANK (**0.30**) by **132.53%** and DECAYRANK (**0.5699**) by **22.41%**.

Beyond precision, the AUC-ROC curve in Figure 5 highlights the model's ability to distinguish between normal and anomalous patterns. ADAPTIVE-DECAYRANK attains the highest AUC score of **0.90** on DARPA, exceeding ANOMRANK (**0.86**) and DECAYRANK (**0.83**), reinforcing its capability to detect subtle structural deviations in dynamic graphs.

ADAPTIVE-DECAYRANK's superior performance stems from its ability to capture *short-term* deviations and *micro-pattern* changes in edge streams using an adaptive decay factor and Bayesian updates. Unlike static methods that treat all edges equally over time, our approach assigns higher weights to recent anomalies while gradually reducing the influence of older observations. This makes it particularly effective in dynamic environments where sudden structural changes or coordinated attacks occur.

6.1.2. Types of Anomalies Detected

ADAPTIVE-DECAYRANK effectively captures both **structural anomalies** and **micro changes** in node influence. Structural anomalies arise when nodes exhibit sudden, unexpected changes in connectivity patterns, as seen in **botnet behavior** in CTU-13 [42] or **network intrusion** patterns in DARPA [41]. These are particularly challenging for methods relying on static graph snapshots, as they fail to adapt to rapidly evolving structures.

Additionally, the model detects **micro-changes in node influence**, where nodes subtly shift their interaction patterns over time. Such anomalies occur in stealthy cyber-intrusions, where attackers gradually increase their influence within a network without abrupt changes [2]. The adaptive decay factor in our model ensures that these incremental deviations accumulate and get flagged as anomalies, making it particularly well-suited for cyber-threat detection.

6.1.3. Computational Efficiency

While ANOMRANK and DECAYRANK execute faster on small datasets (taking **4.03 s** and **5.46 s**, respectively, on DARPA), their lower precision limits their effectiveness. ADAPTIVE-DECAYRANK, in contrast, maintains efficiency, even at large scales. As seen in Table 4 and Figure 6, our model processes **1 million edges in just 14.23 s**, outperforming both SEDANSPOT and DYNANOM, which require substantially more computation.

The model's efficiency stems from its **incremental Bayesian updating and adaptive decay factor**, which avoids full-graph recomputation by updating rankings dynamically. Unlike traditional PageRank-based methods that recompute anomaly scores from scratch, ADAPTIVE-DECAYRANK maintains a dynamic node importance measure that adapts to graph changes in real time, significantly reducing redundant computations.

6.1.4. Robustness Against Noise

Figure 4 demonstrates that ADAPTIVE-DECAYRANK maintains **high precision across various anomaly thresholds**, effectively suppressing false positives, an essential feature for cyber-security and fraud detection applications. For **Top-500 anomalies** (as shown in Table 5), it retains a precision above **0.520** on DARPA, **0.544** on CTU-13, and **0.528** on RTM-30, whereas ANOMRANK drops to **0.30** or lower under similar conditions, highlighting its superior reliability in distinguishing true anomalies from random fluctuations.

Existing research [1,2] has shown that several anomaly detection models struggle with *false positive rate (FPR)*, meaning the proportion of normal instances that are incorrectly classified as anomalies, particularly in highly dynamic datasets, where normal variations may be misclassified as anomalies. ADAPTIVE-DECAYRANK mitigates this issue through its adaptive decay mechanism, which prioritizes recent, sustained anomalies while filtering out transient noise.

A practical example of this robustness is observed in the **CTU-13** dataset, where botnet interactions often mimic normal traffic patterns, making static ranking approaches like ANOMRANK highly prone to false positives. In contrast, ADAPTIVE-DECAYRANK effectively differentiates meaningful botnet behaviors from normal fluctuations, achieving a **6.71%** higher precision than DECAYRANK and a **66.49%** improvement over ANOMRANK (as shown in Tables 4 and 5).

6.1.5. Scalability

As demonstrated in Figure 6, ADAPTIVE-DECAYRANK scales linearly with increasing edge updates, making it suitable for real-time applications. On the DARPA dataset, it efficiently processes small updates in **microseconds**, moderate graphs in **milliseconds**, and $1M$ edges or large-scale datasets in **14.23 s**. Even on RTM-30, which contains **19M edges**, our model maintains a processing speed of **0.08 s per 1M edges**, demonstrating its capability for near real-time anomaly detection. This scalability is essential for applications in cyber-security, where rapid identification of network threats is critical. It is also beneficial in financial fraud detection, where monitoring evolving transaction patterns in real time can help detect coordinated fraudulent activities.

6.2. Addressing Research Questions

As outlined in Section 1.1, our study directly addresses the research questions posed in this paper. Specifically, ADAPTIVE-DECAYRANK successfully detects the sudden (dis)appearance of anomalous patterns in dynamic graphs, including node and structural anomalies **(Q1)**. The proposed method achieves high detection accuracy, precision, and AUC through its *adaptive decay factor* and *Bayesian updating*, effectively prioritizing recent anomalous behaviors. Additionally, our scalability analysis (Figure 6) demonstrates that ADAPTIVE-DECAYRANK efficiently processes large-scale dynamic graphs while maintaining sub-linear time complexity, confirming its suitability for real-time anomaly detection in streaming graph environments **(Q2)**.

6.3. Limitations and Future Work

While ADAPTIVE-DECAYRANK achieves strong performance, it requires more robust fine-tuning of the decay factor to optimize results across different datasets. Additionally, although it effectively detects anomalies in dynamic graphs, it does not leverage deep feature learning, as seen in graph neural networks (GNNs). Future work will explore integrating **self-supervised learning** to improve anomaly detection in complex graph structures. We also aim to optimize performance further using **GPU acceleration**, enabling even faster real-time analysis for large-scale networks. Lastly, we will extend our method

to more diverse domains, including Autonomous Vehicle (AV) security detection, medical data analysis, and industrial IoT security.

7. Conclusions and Future Work

In this work, we propose ADAPTIVE-DECAYRANK, an unsupervised, real-time anomaly detection approach for identifying node-level and structural anomalies in dynamic graph streams. ADAPTIVE-DECAYRANK is a modified dynamic PageRank algorithm with a *decay factor* and Bayesian updates, which enable rapid adaptation to sudden, short-term deviations and facilitate fast, accurate anomaly detection in graph streams. Our experiments demonstrate that ADAPTIVE-DECAYRANK outperforms baseline approaches by **11.28–15.27%** in terms of precision and **4.65–40.63%** in terms of AUC. While processing $4.6M$ edges across 1463 snapshots, our model achieves a processing rate of 1 million edges in just 14.23 s—making it **4.53** times faster than the baseline approaches (SEDANSPOT and DYNANOM). For future work, we will explore more real-world applications, such as financial transactions, medical data, and drug discovery. Additionally, we aim to investigate how GPUs could be leveraged to optimize linear scaling for larger streaming edge sets.

Author Contributions: Conceptualization, O.A.E. and W.E.; Methodology, O.A.E. and W.E.; Software, O.A.E. and J.C.; Validation, O.A.E., W.E. and J.C.; Formal analysis, O.A.E., W.E. and J.C.; Resources, W.E.; Data curation, J.C.; Writing—original draft, O.A.E.; Writing—review & editing, W.E.; Visualization, O.A.E.; Supervision, W.E.; Project administration, W.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially supported by the National Science Foundation (NSF-REU, #2349104).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions, including the code and datasets presented in this study, are openly available in the Adaptive-DecayRank GitHub repository at https://github.com/EkleTony/Adaptive-DecayRank. Further inquiries should be directed to the first author, O.A.E.

Acknowledgments: The authors wish to thank the College of Engineering, the Machine Intelligence and Data Science Center, and the Computer Science Department at Tennessee Tech University for providing the resources and funding for this work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ranshous, S.; Shen, S.; Koutra, D.; Harenberg, S.; Faloutsos, C.; Samatova, N.F. Anomaly detection in dynamic networks: A survey. *Wiley Interdiscip. Rev. Comput. Stat.* **2015**, *7*, 223–247. [CrossRef]
2. Ekle, O.A.; Eberle, W. Anomaly Detection in Dynamic Graphs: A Comprehensive Survey. In *ACM Transactions on Knowledge Discovery from Data*; Association for Computing Machinery: New York, NY, USA, 2024.
3. Bhatia, S.; Hooi, B.; Yoon, M.; Shin, K.; Faloutsos, C. Midas: Microcluster-based detector of anomalies in edge streams. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 3242–3249. [CrossRef]
4. Zhang, G.; Li, Z.; Huang, J.; Wu, J.; Zhou, C.; Yang, J.; Gao, J. efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Trans. Inf. Syst. (TOIS)* **2022**, *40*, 1–29. [CrossRef]
5. Li, S.; Pandey, A.; Hooi, B.; Faloutsos, C.; Pileggi, L. Dynamic graph-based anomaly detection in the electrical grid. *IEEE Trans. Power Syst.* **2021**, *37*, 3408–3422. [CrossRef]
6. Ekle, O.A.; Ulybyshev, D. Enhanced Categorization of Cybersecurity Vulnerabilities. In Proceedings of the 2024 IEEE 15th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Yorktown Heights, NY, USA, 17–19 October 2024; pp. 800–806.
7. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
8. Waikhom, L.; Patgiri, R. Graph neural networks: Methods, applications, and opportunities. *arXiv* **2021**, arXiv:2108.10733.

9. Yoon, M.; Jin, W.; Kang, U. Fast and accurate random walk with restart on dynamic graphs with guarantees. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 409–418.
10. Loshchilov, I.; Hutter, F. Fixing weight decay regularization in adam. *arXiv* **2017**, arXiv:1711.05101.
11. Bottou, L. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade: Second Edition*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 421–436.
12. Kingma, D.P. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
13. Page, L. *The Pagerank Citation Ranking: Bringing Order to the Web. Technical Report*; Stanford Digital Library Technologies Project 1998; Stanford Digital Library: Stanford, CA, USA, 1998.
14. Newman, M.E. A measure of betweenness centrality based on random walks. *Soc. Netw.* **2005**, *27*, 39–54. [CrossRef]
15. Lü, L.; Chen, D.; Ren, X.L.; Zhang, Q.M.; Zhang, Y.C.; Zhou, T. Vital nodes identification in complex networks. *Phys. Rep.* **2016**, *650*, 1–63. [CrossRef]
16. Leskovec, J.; Rajaraman, A.; Ullman, J.D. *Mining of Massive Data Sets*; Cambridge University Press: Cambridge, UK, 2020.
17. Yu, R.; Qiu, H.; Wen, Z.; Lin, C.; Liu, Y. A survey on social media anomaly detection. *ACM SIGKDD Explor. Newsl.* **2016**, *18*, 1–14. [CrossRef]
18. Eswaran, D.; Faloutsos, C. Sedanspot: Detecting anomalies in edge streams. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 953–958.
19. Eberle, W.; Holder, L. Discovering structural anomalies in graph-based data. In Proceedings of the Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), Omaha, NE, USA, 28–31 October 2007; pp. 393–398.
20. Yoon, M.; Hooi, B.; Shin, K.; Faloutsos, C. Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 647–657.
21. Holme, P.; Saramäki, J. Temporal networks. *Phys. Rep.* **2012**, *519*, 97–125. [CrossRef]
22. Paudel, R.; Eberle, W. Snapsketch: Graph representation approach for intrusion detection in a streaming graph. In Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG), San Diego, CA, USA, 23–27 August 2020.
23. Guo, X.; Zhou, B.; Skiena, S. Subset node anomaly tracking over large dynamic graphs. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 475–485.
24. Ranshous, S.; Harenberg, S.; Sharma, K.; Samatova, N.F. A scalable approach for outlier detection in edge streams using sketch-based approximations. In Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, FL, USA, 5–7 May 2016; pp. 189–197.
25. Belth, C.; Zheng, X.; Koutra, D. Mining persistent activity in continually evolving networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 934–944.
26. Bhatia, S.; Liu, R.; Hooi, B.; Yoon, M.; Shin, K.; Faloutsos, C. Real-time anomaly detection in edge streams. *ACM Trans. Knowl. Discov. Data (TKDD)* **2022**, *16*, 1–22. [CrossRef]
27. Chang, Y.Y.; Li, P.; Sosic, R.; Afifi, M.; Schweighauser, M.; Leskovec, J. F-fade: Frequency factorization for anomaly detection in edge streams. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Online, 8–12 March 2021; pp. 589–597.
28. Bhatia, S.; Wadhwa, M.; Kawaguchi, K.; Shah, N.; Yu, P.S.; Hooi, B. Sketch-Based Anomaly Detection in Streaming Graphs. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 93–104.
29. Wang, Y.; Chakrabarti, A.; Sivakoff, D.; Parthasarathy, S. Fast change point detection on dynamic social networks. *arXiv* **2017**, arXiv:1705.07325.
30. Shin, K.; Hooi, B.; Kim, J.; Faloutsos, C. Densealert: Incremental dense-subtensor detection in tensor streams. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 1057–1066.
31. Huang, S.; Hitti, Y.; Rabusseau, G.; Rabbany, R. Laplacian change point detection for dynamic graphs. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 349–358.
32. Xie, Y.; Wang, W.; Shao, M.; Li, T.; Yu, Y. Multi-view change point detection in dynamic networks. *Inf. Sci.* **2023**, *629*, 344–357. [CrossRef]
33. Yang, C.; Zhou, L.; Wen, H.; Zhou, Z.; Wu, Y. H-vgrae: A hierarchical stochastic spatial-temporal embedding method for robust anomaly detection in dynamic networks. *arXiv* **2020**, arXiv:2007.06903.
34. You, J.; Du, T.; Leskovec, J. ROLAND: Graph learning framework for dynamic graphs. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2358–2366.
35. Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; Liu, T.Y. Do transformers really perform badly for graph representation? *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 28877–28888.

36. Liu, Y.; Pan, S.; Wang, Y.G.; Xiong, F.; Wang, L.; Chen, Q.; Lee, V.C. Anomaly detection in dynamic graphs via transformer. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 12081–12094. [CrossRef]
37. Tong, H.; Faloutsos, C.; Pan, J.Y. Fast random walk with restart and its applications. In Proceedings of the Sixth International Conference on Data Mining (ICDM'06), Hong Kong, China, 18–22 December 2006; pp. 613–622.
38. Ekle, O.A.; Eberle, W. Dynamic PageRank with Decay: A Modified Approach for Node Anomaly Detection in Evolving Graph Streams. In Proceedings of the International FLAIRS Conference Proceedings, Sandestin Beach, FL, USA, 19–21 May 2024; Volume 37.
39. Kleinberg, J.M. Authoritative sources in a hyperlinked environment. *J. ACM (JACM)* **1999**, *46*, 604–632. [CrossRef]
40. Wasserman, L. *All of Statistics: A Concise Course in Statistical Inference*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
41. Lippmann, R.; Haines, J.W.; Fried, D.J.; Korba, J.; Das, K. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In Proceedings of the Recent Advances in Intrusion Detection: Third International Workshop, RAID 2000, Toulouse, France, 2–4 October 2000; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2000; pp. 162–182.
42. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur.* **2014**, *45*, 100–123. [CrossRef]
43. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. CTU-13 Dataset: A Labeled Dataset with Botnet, Normal, and Background Traffic. 2011. Available online: https://www.stratosphereips.org/datasets-ctu13 (accessed on 17 February 2025).
44. Akoglu, L.; McGlohon, M.; Faloutsos, C. RTM: Laws and a recursive generator for weighted time-evolving graphs. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 701–706.