Dynamic PageRank with Decay: A Modified Approach for Node Anomaly Detection in Evolving Graph Streams

Ocheme Anthony Ekle
Department of Computer Science
Tennessee Technological University
Cookeville, TN, USA
oaekle42@tntech.edu

William Eberle
Department of Computer Science
Tennessee Technological University
Cookeville, TN, USA
weberle@tntech.edu

Abstract

Given a large graph stream with dynamically changing structures over a given timestep, it is important to detect the sudden appearance of anomalous patterns, such as sudden spikes in IP-network attacks or unexpected surges in social media followers. In addition, it is important that we promptly identify these abrupt changes in the network by considering swift and short-term responses within the network structure. To design a model capable of adapting to dynamic changes, we introduce an approach that utilizes a modified dynamic "PageRank-with-Decay" as a node scoring function. This method enables the detection of sudden dynamic graph changes based on node importance scores, leveraging the temporal evolution of graph structures at each timestep. This approach provides a refined anomaly detection mechanism for tracking rapid structural changes in the network. Through experiments conducted on a real-world dataset, our model demonstrates faster and more accurate results (in terms of precision and recall) compared to state-of-the-art methods.

1 Introduction

The demand for more robust, scalable, and fast adaptive anomaly detection techniques has attracted wide interest in graph representation research. Dynamic network-based systems in the real world, like IP-IP traffic, social networks, financial transactions, and the spread of a disease, have been modeled as dynamic graph tasks because of the patterns and relationships between entities (like nodes, edges, and subgraphs). Some emerging challenges of these constantly evolving networks include: (1) how to effectively design techniques that could learn complex dynamic graph structures; and (2) models that can accurately detect rapid change in patterns and dependencies among connected entities.

The typical approach is to consider classical machine learning and deep learning algorithms. Unfortunately, these frameworks are limited to data sequences, images, and grid-like data, and are also susceptible to inductive errors when dealing with a large number of parameters (Leskovec, Rajaraman, and Ullman 2020). Several approaches in the literature have focused on detecting anomalies in static graphs where the network topology remains constant. However, real-world networks are represented as dynamic graphs with evolving changes.

In recent years, anomaly detection approaches have been proposed. These include matrix factorization and tensor decomposition (Huang et al. 2020), probabilistic models (Bhatia et al. 2022; Chang et al. 2021), distance-based methods (Eswaran and Faloutsos 2018; Yoon et al. 2019), and deep-learning graph embedding (You, Du, and Leskovec 2022) (detailed reviews are provided in Sec.2). Despite their respective successes, these techniques are limited in terms of speed, accuracy, adaptability (concept drift), and scalability. Detecting anomalous network behaviors in real-world critical infrastructures such as power grids and nuclear plants requires prompt responses and maximum accuracy.

In this paper, we propose DecayRank, a fast and accurate node-level anomaly detection algorithm for dynamic graphs. This is a modified version of the dynamic PageRank algorithm (Yoon, Jin, and Kang 2018) tailored to adapt and capture sudden spikes in network structures swiftly. DecayRank introduces a decay factor, strategically incorporated into the iterative dynamic PageRank formula, to provide a swift response to recent and short-term graph changes.

The motivation behind DecayRank lies in its ability to capture abrupt structural alterations in dynamic graph networks. 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. By assigning higher scores to nodes experiencing rapid changes, DecayRank serves as a refined mechanism for detecting anomalies in evolving graph streams. 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 DecayRank, a modified dynamic PageRank with a decay factor for swift responses to graph changes.
- We demonstrate that our approach assigns more weight to recent edges, thereby enhancing anomaly detection.
- We perform extensive experiments on real-world datasets to show the performance of our DecayRank algorithm.

2 Related Work

In this section, we review existing methods for detecting anomalies in dynamic graphs (Ranshous et al. 2015; Bhatia et al. 2022), categorizing them into four groups based on their algorithmic structures and limitations in spotting anomalous patterns.

Distance and Similarity-Based Methods: measure similarities between nodes using time-evolving metrics to detect anomalies. Commonly used similarity metrics include PageRank (Page 1998), HITS (Kleinberg 1999), and Betweenness Centrality (Newman 2005). NetWalk learns node representations and detects deviations based on a dynamic clustering algorithm. StreamSpot (Manzoor, Milajerdi, and Akoglu 2016) uses a similarity function for heterogeneous graphs; SpotLight (Eswaran et al. 2018) employs randomized sketching for densely directed subgraphs; and SedanSpot (Eswaran and Faloutsos 2018) detects sparsely-connected edges. GBAD (Velampalli and Eberle 2017) leverage background knowledge rules, with minimum descriptive length (MDL) and size metrics in detecting substructural anomalies. AnomRank (Yoon et al. 2019) applies Personalized PageRank to detect structure change and weighted edges in dynamic graphs. However, this does not work well for processing edges in a stream.

SnapSketch (Paudel and Eberle 2020) is a sketch-based approach with hashing of the discriminative shingles generated from a biased-random walk. However, this method cannot detect sudden changes or rapid bursts in networks. Recent methods include DYNWATCH (Li et al. 2021) for real-time detection of sensor data in a power grid. It calculates the graph distance using the Line Outage Distribution Factors (LODF) sensitivity measure. DynAnom (Guo, Zhou, and Skiena 2022) detects anomalies at both the node and graph levels in large weighted graphs.

Probabilistic Methods: uses probabilistic models to represent neighborhood relationships in graphs. RHSS (Ranshous et al. 2016) applies the Count-Min sketch to approximate graph properties and generate probabilistic error bounds on each edge. PENminer (Belth, Zheng, and Koutra 2020), explores the persistence of activity snippets in edge streams. However, PENminer is not equipped to detect subgraph and graph-level anomalies. MIDAS-R (Bhatia et al. 2022) detects microcluster anomalies in edge streams and uses Count-Min sketches (CMS) to count edge occurrences. However, MIDAS is limited in tracking node and community graph structures.

F-FADE (Chang et al. 2021) is a frequency-factorization method for detecting edge patterns by estimating the maximum likelihood rule for each incoming interaction. However, it requires a higher computational complexity.

MSTREAM (Bhatia et al. 2021) detects group anomalies in multi-aspect data using locality-sensitive hash functions (Charikar 2002). AnoEdge (Bhatia et al. 2023), an extension of MIDAS-R, uses a higher-order sketch instead of Count-Min sketches (CMS). However, these methods only focus on detecting edge- or subgraph-level tasks. In contrast, our proposed method (DecayRank) addresses node-level tasks by considering the node importance score and detecting rapid structural changes in the graph.

Matrix Factorization Methods: focus on decomposing high-dimensional matrices into lower-dimensional forms, revealing evolving patterns and relationships in dynamic graphs. A recent method is EdgeMonitor (Wang et al. 2017), an edge-detection approach that models dynamic graph evolution as a first-order Markov process. However, one major limitation is that it relies on consistent node ordering across all time steps. DenseAlert (Shin et al. 2017) is an incremental method for detecting dense subtensors in tensor streams. LAD (Huang et al. 2020) uses the Laplacian spectrum for change point detection, computing the singular value decomposition (SVD) of the graph Laplacian to obtain a low-dimensional graph representation. MultiLAD (Xie et al. 2023) generalizes the LAD algorithm to multi-view graph detection. Despite the recent successes with matrix factorization methods, they are computationally expensive and require manual extraction of dynamic graph properties.

Deep Graph Learning Methods: Recent methods leverage neural networks to extract and learn representations of complex graph structures. H-VGRAE (Yang et al. 2020) apply graph autoencoders for node embedding. ROLAND (You, Du, and Leskovec 2022) extends static graph neural networks (GNNs) to capture dynamic graphs. Transformer models such as Graphomer (Ying et al.) and TADDY (Liu et al. 2021) have also been used for dynamic graph learning. However, deep learning methods are limited in handling graph data due to the intricate topological structures of graphs.

Our method, DecayRank, falls within the category of *distance-based methods*. DecayRank leverages the dynamic PageRank-with-Decay algorithm as a similarity metric to assign node importance scores and rankings. The *Decay-factor* in DecayRank is instrumental in tracking and detecting micro changes in edges, such as insertions or deletions, at each graph timestamp. In contrast, several anomaly detection methods, like F-FADE, MIDAS-R, MSTREAM, AnoEdge, and others, use computationally intensive stochastic approximation methods that are mostly used for edge and subgraph detection tasks.

3 Preliminaries

In this section, we define dynamic graphs, review the original PageRank algorithm, and discuss its dynamic variants. Additionally, we outline the problem formulation of our node-level anomaly detection method. The complete set of notations is provided in Table 2.

Definition 1 *Given a graph $G = (V, E, W)$, where $V = \{v_1, \ldots, v_n\}$ is the node set, $E \subseteq V \times V$ is the edge multi-set, and W is the set of edge weights, a **dynamic graph with weights** $\mathcal{G} = \{G_t\}_{t=1}^{T}$ can be defined as a sequence of ordered sets of graph snapshots at different timestamps t, where T is the total number of time steps. Each snapshot is considered a static graph $G_t = (V_t, E_t \subseteq (V_t \times V_t), W_t)$ with vertex set $V_t = \{v \in V \| i_v = t\}$, edge set $E_t = \{e \in E \| i_e = t\}$, and weight set W_t representing the weights assigned to the edges in E_t. The edges may consist of plain or labeled edges.*

Table 1: Comparison of relevant anomaly detection methods in dynamic graph

Property	DenseAlert (2017)	SedanSpot (2018)	SpotLight (2018)	MIDAS-R (2020)	F-FADE (2021)	DYNWATCH (2021)	DynAnom (2022)	MultiLAD (2023)	AnoEdge (2023)	**Our Method**
Edge Anomaly	✓	✓		✓	✓	✓			✓	-
Node Anomaly							✓	✓		✓
Real-time Detection	✓			✓		✓	✓		✓	✓
Structural anomalies								✓		✓
Sudden Edges Changes	✓	✓	✓	✓	✓	✓			✓	✓

Table 2: List of Notations

Symbol	**Definition**
G	weighted graph with nodes V and edges E.
G_t	sequence of snapshots of a dynamic graph $\mathcal{G} = \{G_t\}_{t=1}^{T}$ at each timestamp t.
T	the total number of time steps.
V_t	the vertex set for the graph at time point t.
E_t	the edge set at time point t.
W_t	weights assigned to edges in E_t.
N, n	numbers of nodes and edges in $\mathcal{G}$
P	$(n \times n)$ row-normalized adjacency matrix, $\in \mathbb{R}^{n \times n}$ and $P = D^{-1}A$
D^{-1}, A	$(n \times n)$ adjacency matrix and degree matrix of $\mathcal{G}$
c	damping factor of PageRank
$h, h(t+1)$	$(n \times 1)$ starting probability vector with change over time.
$\lvert f(v') - \hat{f} \rvert$	the absolute difference between the score assigned to vertex v' by the scoring function $f(v')$
$f(v)$	node-level anomaly scoring function

3.1 Original PageRank

PageRank (Page 1998) is an algorithmic function that assigns vector scores to each page on the web, treating the web as a directed graph network with pages as nodes. The fundamental concept is that pages with higher scores signify greater importance. The basic recursive formula is:

$$PR(p_i) = \frac{1-c}{N} + c \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(pj)} \tag{1}$$

where $PR(p_i)$ is the PageRank score of page p_i, N denotes the total number of pages, and c is the *damping factor*. This factor represents the probability that a random surfer will teleport to another page or graph node and is commonly set to 0.85. **Note**: Without the restart probability, the random surfer would converge to a normalized variant of the eigenvector centrality. $M(p_i)$ is the set of pages that link to page p_i, and $L(p_j)$ is the number of edges from page p_j. Several variants of the PageRank algorithm have been developed to personalize scores according to specific tasks. One such variant is the Random Walks with Restarts (RWR) (Tong, Faloutsos, and Pan 2006). In the subsequent subsections, we will provide a brief overview of RWR and the dynamic PageRank variant.

3.2 Personalized PageRank and RWR

The standard representation of the personalized PageRank vector (Leskovec, Rajaraman, and Ullman 2020) is calculated as follows:

$$v_u = cPv_u + (1-c)h \tag{2}$$

where $P = D^{-1}A$ is the stochastic matrix, $P \in \mathbb{R}^{n \times n}$. $A \in \mathbb{R}^{n \times n}$ and $D \in \mathbb{R}^{n \times n}$ are the adjacency matrix and degree matrix of a given graph $\mathcal{G}$ with n nodes. $\boldsymbol{h}$ is the indicator vector for node u (i.e., the teleport vector), and v_u represents the stationary probability that a random walk starting at node $\boldsymbol{u_i}$ visits node $\boldsymbol{u_{i+1}}$. $\boldsymbol{c}$ is the *damping factor*, representing the probability that the random walker restarts at node u at each timestep.

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.

3.3 Dynamic PageRank Variant

Our method is based on the Dynamic PageRank (Yoon, Jin, and Kang 2018) variant for personalized PageRank that supports time-evolving components. This is suited for dynamic graph tasks where the current PageRank score v^{i+1} can be updated incrementally from previous scores v^i. The formula to compute the dynamic PageRank score is given below.

$$v_u(t+1) = cPv_u(t) + (1-c)h(t+1) \tag{3}$$

Equation 3 follows directly from Equation 2 where $P^{(t)}$ is the stochastic matrix (or transition matrix), $P = D^{-1}A$, and it is dependent on each timestep of the graph. c is the damping factor, and the indicator probability vector $h(t+1)$ allows for the teleport vector to change over time.

Equation 3 undergoes incremental updates upon the insertion or deletion of edges, making it well-suited for practical downstream tasks like graph learning. Several studies (Yoon, Jin, and Kang 2018; Guo, Zhou, and Skiena 2022) have shown that dynamic page rank works well on graph data. This is because it uses sparse matrix operations, provides for incremental updates, is memory efficient (by using data structures and algorithms that use as little memory as possible), and includes well-designed convergence criteria.

3.4 Problem Formulation

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

Definition 2 ***Node Anomaly in Dynamic Weighted Graph*** *Given a dynamic weighted-graph $\mathcal{G} = (V_t, E_t, W_t) = G_{t=0}^{T}$, consisting of the initial snapshot G_0, $E_t = \cup_{t=0}^{T} E_t$, and $V_t = \cup_{t=0}^{T} V_t$ is the total vertex set. Here, each snapshot has change in edge events $|\Delta E_t \geq 0|$ (i.e., edge deletion, insertion from t to t+1), and $f : V \rightarrow \mathbf{R}$ is a specified scoring function (e.g., PageRank, HITs). The set of anomalous vertices $V' \subseteq V$ is defined such that $\forall v' \in V'$, $|f(v') - \hat{f}| > c_0$, where $\hat{f}$ is a summary statistic of the score $f(v), \forall v \in V$.*

Problem 1 *Our major task is to detect anomalous structural changes in G_t. Specifically, when a node u experiences a sudden transition from its original set of out-going edges to neighbors $\langle v_1, \ldots, v_{\Delta n} \rangle$ to a new set $\langle \hat{v}_1, \ldots, \hat{v}_{\Delta n} \rangle$, it is labeled as a node-level structural anomaly.*

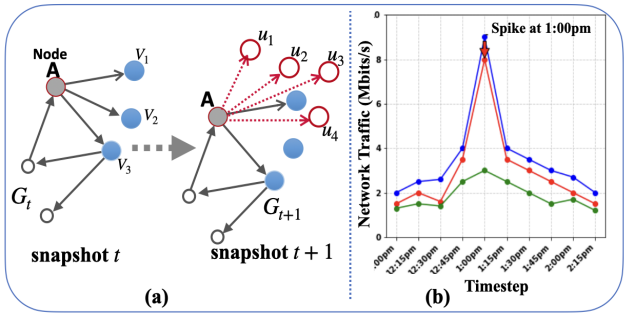

Figure 1: Node-level Structural Anomaly in Dynamic Graph

Figure 1 illustrates structure change in a network. For example, in Figure 1 **(a)**, the sudden change in node and influx of new set of edges could indicate potential cyber-attacks on computer networks or perhaps a series of fraudulent transactions. In Figure 1 **(b)**, there is a sudden spike at 1:00, suggesting a potential cyber attack. Intuitively, to detect this kind of structural anomaly, our approach should target the existence of edges connecting two nodes rather than the frequency of edge occurrences between them.

4 Proposed Method

Our objective is to assign a node importance score to nodes across the graph. To achieve this, our method DecayRank adapts the dynamic PageRank variant with the decay factor outlined in Equation 3 as our node scoring function. This modification involves the incorporation of a ***decay-factor***, taking into account several factors: **(a) Swift Response to Recent Changes:** The inclusion of the decay factor enables PageRank to respond rapidly to recent and short-term alterations in the graph. **(b) Enhanced Adaptability:** The decay factor enhances the algorithm's adaptability to dynamic changes, ensuring it can effectively capture the evolving nature of the graph. **(c) Prevention of Long-Term Impact Accumulation:** The decay factor plays a crucial role in preventing the accumulation of long-term edge change impacts and maintaining the algorithm's responsiveness and accuracy. This modification ensures that the algorithm assigns greater weight to recent edges, contributing to its effectiveness in capturing the dynamic nature of the evolving graph.

Algorithm 1 DecayRank: Node Anomaly Scoring

Input: $\boldsymbol{A}$: Array of dynamic graph outEdges, $\boldsymbol{v}$: Array of PageRank scores, $\boldsymbol{n}$: Number of nodes, $numSteps$: streaming timesteps
Output: v: Updated PageRank vectors $\{v^{(0)}, \ldots, v^{(T)}\}$

▷ **Initialization**:
for $i = 0$ to $n - 1$ **do**
  $v[i] \leftarrow \frac{c}{n}$ ▷ Initialization based on version
  $nq[i] \leftarrow 0$
  $nq_{\text{prev}}[i] \leftarrow v[i]$
end for
for $step = 0$ to $numSteps - 1$ **do**
▷ **Power Iteration:**
  for $i = 0$ to $n - 1$ **do**
    if $nq_{\text{prev}}[i] \neq 0$ **then**
      $\delta \leftarrow$ calculateDelta$(nq_{\text{prev}}, A, i, c, version)$
      for j in $A[i]$.out **do**
        $nq[j]$ += $\delta \times A[i]$.weight$[j]$
      end for
    end if
  end for
▷ **Decay Factor Application:**
  for $i = 0$ to $n - 1$ **do**
    $nq[i] *= e^{-\text{DECAY\_FACTOR} \times (step - A[i].\text{timestamp})}$
  end for
  for $i = 0$ to $n - 1$ **do**
    $v[i]$ += $nq[i]$
  end for
end for
▷ **Normalization:**
$v \leftarrow$ normalizePageRank(b, n)
return v

4.1 Dynamic PageRank with Decay: (DecayRank)

NodeScorer: This phases aims to assign PageRank score values to individual nodes in the graph considering the neighboring edges and evolving properties. Our modified node scoring function is based on the iterative formula from Equation 3. We introduce the **decay factor** into the iterative formula and define the modified version of the node score vector v_u below:

$$v_u = c\hat{P}v_u + (1 - c)h_u \mathbf{e}^{-\delta \cdot (\mathbf{t} - \mathbf{A_u}.\mathbf{timestamp})} \quad (4)$$

Here, $\hat{\boldsymbol{P}}$ is the row-normalized adjacency matrix, and all other parameters remain the same as in Equation 3.

Decay Factor: The use of the decay factor in our work is inspired by the research of (Loshchilov and Hutter 2018) on *"Weight Decay Regularization in ADAM."* The authors propose decoupling gradient-based updates from weight decay in both the SGD and Adam optimizers. Additionally, they introduce a variant of Adam with warm restarts, achieving state-of-the-art results.

Equation 4 introduces the exponential decay function $\boldsymbol{e^{-\delta \cdot (t - A_u.\text{timestamp})}}$, contributing to the adaptability and

quick responsiveness of the algorithm. $\boldsymbol{\delta}$ is the "*decay factor*", and it controls the rate at which edge weight decreases over time. A higher $\boldsymbol{\delta}$ results in a faster decay, signifying a reduced influence of older edges on PageRank scores. $\boldsymbol{t}$ is the current timestamp, and $\boldsymbol{A_u}$ denotes a specific attribute related to the timestamp of node $\boldsymbol{u}$. The expression $\boldsymbol{(t - A_u.\text{timestamp})}$ calculates the time difference between the current timestamp and that of node $\boldsymbol{u}$. In summary, the decay factor function utilizes this time difference and $\boldsymbol{\delta}$ to control the exponential decay of edge weights based on their age or time of occurrence.

Lemma 1 ***(Exponential Decay of Expected Node Score Change).*** *For any node i in an evolving (un)directed graph stream, let P_i^t be the node score at timestep t computed by the dynamic PageRank-with-decay. Define ΔP_i^t as the change in node score from timestep $t-1$ to t, and decay factor $e^{-\delta \cdot (t - A_u.\text{timestamp})}$. The expected change in node score, $\mathbb{E}[\Delta P_i^t]$, follows an exponential decay, emphasizing a recent change of edge interactions in the graph.*

Note: *Due to space constraints, the proof of the Lemma has been omitted. We will provide a proof in our future work.*

4.2 Algorithm

Algorithm 1, an implementation of PageRank-with-Decay, describes how the node scores are calculated. The algorithm takes inputs $\boldsymbol{A}$, an array of outEdges representing the transition matrix, and $\boldsymbol{v}$, a sequence of stochastic vectors. In the initialization phase, the algorithm uniformly initializes scores for all nodes. During power iteration over timesteps, PageRank updates its scores based on the graph structure and edge weights. Then, in Lines 15–21, the exponential decay factor is applied to compute updated scores based on evolving graph information. After each iteration, the new PageRank score is updated by assigning values or weights from the temporary buffer $\boldsymbol{nq}$ in line 16. Finally, the DecayRank algorithm normalizes each node's scores by subtracting their mean and dividing by their standard deviation to ensure they sum to 1.

4.3 Similarity Metrics for Node Anomaly Detection

After Algorithm 1 computes the score vectors for each node with an exponential decay factor, we apply similarity metrics from (Yoon et al. 2019) to detect anomalousness at each time step in the graph. Given scores vector $\boldsymbol{v_u}$, the metric formula is shown below.

$$v_u'' = \frac{(v_u(t + \Delta t) - v_u(t)) - (v_u(t)) - v_u(t - \Delta t)}{\Delta t^2} \quad (5)$$

Equation 5 is the discretization of the second-order derivative of the score vector $\boldsymbol{v_u}$. The derivative characterizes the rate of change of a function, and discretization, in this context, approximates a continuous function in discrete form for easy computation and analysis.

5 Experiments

In this section, we evaluate the performance of DecayRank for node-level anomaly tasks in comparison to three state-of-the-art baselines in terms of accuracy, scalability, and speed.

5.1 Datasets

DARPA: DARPA (Lippmann et al. 2000) 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 and each edge representing network traffic, and each *IP-IP* communication is represented as a directed edge (sourceIP, destIP, timestamp, anomalyAttack). We present a summary of the data in Table 3.

Table 3: Statistics of DARPA Data

Graphs	**Nodes $\lvert V\rvert$**	**Edges $\lvert E\rvert$**	**Time Span**
Darpa	$33,221$	$4.6M$	2 months

5.2 Experimental Setup:

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

We compare DECAYRANK to three state-of-the-art distance-based methods for anomaly detection. **SEDANSPOT**(Eswaran and Faloutsos 2018) detects edges that are sparsely connected, **ANOMRANK** (Yoon et al. 2019) for node-level detection with global Personalized PageRank, and **DYNANOM** (Guo, Zhou, and Skiena 2022) which uses a dynamic push algorithm for calculating Personalized PageRank. Our work is closely related to **ANOMRANK**, however, we introduce the decay factor for fast response to short-term changes in an evolving graph. The average precision and runtimes are shown Table 4.

Table 4: Average Precision and Run-time

Algorithms	Darpa	Run-time (sec)
SedanSpot	0.56	$83.66s$
AnomRank	0.62	$4.02551s$
DynAnom	0.5425	$379.334s$ *(Python)*
DECAYRANK	**0.64**	$\mathbf{5.33487}\boldsymbol{s}$

5.3 Accuracy and Speed

Precision and Recall: We evaluate the performance of DECAYRANK using the precision-recall metric. First, we compute the anomaly score for all 1139 graph snapshots and calculate the top-k most anomalous snapshots ($k = 50, 100, \ldots, 800$). In Table 4, DECAYRANK achieves a high average precision score of $\mathbf{0.64}$ compared to baseline ANOMRANK with 0.62 and DYNANOM with 0.54. This is a $\mathbf{3.23}\%$ and $\mathbf{18.5}\%$ accuracy improvement at $\mathbf{5.33487}\boldsymbol{s}$ faster speed compared to SEDANSPOT and DYNANOM. However, ANOMRANK has a run-time of $4.02551s$ which is slightly faster than our method, and understandable considering the iterative implementation of the decay factor at each timestep.

Precision-Recall curve: In Figure 2, DECAYRANK demonstrates higher precision and recall compared to ANOMRANK and SEDANSPOT. It consistently achieves higher precision on high ranks ($top{-}50, 100, 250, \ldots, 800$).

This indicates that DECAYRANK effectively identifies anomalies while maintaining a low false positive rate. This is crucial in real-world scenarios, especially considering that anomalous instances are associated with a significant influx of unusual patterns in the network.

Robustness across Different k: In Figure 2, we can see that at $k \geq 250$, DECAYRANK consistently outperforms baselines in terms of precision. This shows the robustness of DECAYRANK with an increase in anomalies. This is further illustrated in Table 5.

Table 5: The Precision of Top 250-600 Anomalies: **ANOMRANK** vs. **DECAYRANK** ($\delta = \mathbf{0.95}$)

Anomalies	AnomRank	DecayRank
Top-250	0.608	**0.632**
Top-300	0.56	**0.59**
Top-350	0.52	**0.59**
Top-400	0.495	**0.588**
Top-450	0.471	**0.5689**
Top-500	0.462	**0.564**
Top-550	0.445	**0.527**
Top-600	0.436	**0.493**

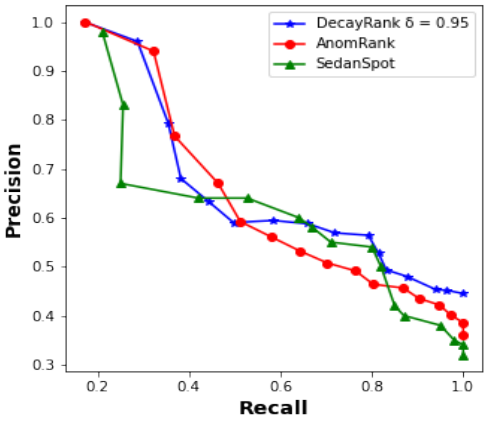

Figure 2: Precision-Recall Curve

Consistent Improvement with Decay Factor: In Table 5 we show that DECAYRANK consistently improves with a high decay factor of $\boldsymbol{\delta = 0.95}$ at the top-250 to 600 anomalies. We observe that DECAYRANK outperforms ANOMRANK significantly at each top-k anomaly. This further validates our *Lemma* 1 that a high $\boldsymbol{\delta}$ implies short-term adaptivity and high sensitivity to sudden edge changes in the graph.

5.4 Scalability

Figure 3 shows how well DECAYRANK scales linearly with an increasing number of edges. We plot the wall-clock time needed to run on the (chronologically) first $2^1, 2^2, \ldots, 2^{22}$ edges of the DARPA dataset. For *small input* $(10, 100)$ the elapsed time is in microseconds-indicating DECAYRANK processes small graphs quickly. At *medium size* $(1000 - 10000)$ the elapsed time is still relatively small (milliseconds), indicating that the algorithm scales to moderate-sized

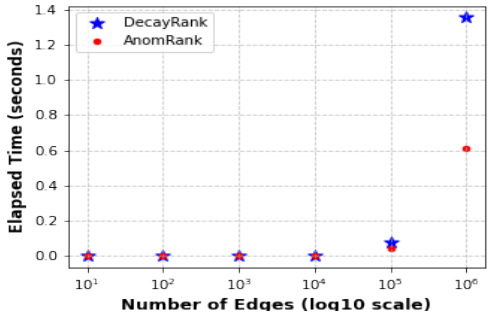

Figure 3: Scalability Analysis

graphs. However, at a *larger size* $(100,000 - 1,000,000)$, the elapsed time increases exponentially (seconds). This may be due to large computational resources in graph processing.

6 Summary and Future Work

In this work, we propose DECAYRANK, an unsupervised, online approach for detecting node-level anomalies in dynamic graph streams. DECAYRANK is a modified dynamic PageRank algorithm with a **decay factor** for fast and accurate detection of sudden and short-term changes in the graph. We present a theoretical claim that a decay factor enhances DECAYRANK to adapt to fast changes in evolving graphs. Our experiments show that DECAYRANK outperforms baseline approaches by $\mathbf{3.23\% - 18.5\%}$ in accuracy (in terms of precision and recall), and processes 1139 graph snapshots faster at 5.33 seconds when compared to baseline approaches. For future work, we will explore more real-world graph data, provide a detailed proof of the adaptiveness lemma 1, and explore how GPU computations could be used to optimize linear scaling for a larger number of edges. Furthermore, we aim to examine temporal-aware PageRank as a better node-scoring function for dynamic graphs.

7 Acknowledgments

The authors wish to thank the College of Engineering, the Machine Intelligence and Data Science (MInDS) Center, and the Computer Science Department at Tennessee Tech for providing resources and funding to work on this project.

References

Belth, C.; Zheng, X.; and Koutra, D. 2020. Mining persistent activity in continually evolving networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 934–944.

Bhatia, S.; Jain, A.; Li, P.; Kumar, R.; and Hooi, B. 2021. Mstream: Fast anomaly detection in multi-aspect streams. In *Proceedings of the Web Conference 2021*, 3371–3382.

Bhatia, S.; Liu, R.; Hooi, B.; Yoon, M.; Shin, K.; and Faloutsos, C. 2022. Real-time anomaly detection in edge streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16(4):1–22.

Bhatia, S.; Wadhwa, M.; Kawaguchi, K.; Shah, N.; Yu, P. S.; and Hooi, B. 2023. Sketch-based anomaly detection in streaming graphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 93–104.

Chang, Y.-Y.; Li, P.; Sosic, R.; Afifi, M.; Schweighauser, M.; and Leskovec, J. 2021. F-fade: Frequency factorization for anomaly detection in edge streams. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 589–597.

Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 380–388.

Eswaran, D., and Faloutsos, C. 2018. Sedanspot: Detecting anomalies in edge streams. In *2018 IEEE International conference on data mining (ICDM)*, 953–958. IEEE.

Eswaran, D.; Faloutsos, C.; Guha, S.; and Mishra, N. 2018. Spotlight: Detecting anomalies in streaming graphs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1378–1386.

Guo, X.; Zhou, B.; and Skiena, S. 2022. Subset node anomaly tracking over large dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 475–485.

Huang, S.; Hitti, Y.; Rabusseau, G.; and Rabbany, R. 2020. Laplacian change point detection for dynamic graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 349–358.

Kleinberg, J. M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46(5):604–632.

Leskovec, J.; Rajaraman, A.; and Ullman, J. D. 2020. *Mining of massive data sets*. Cambridge university press.

Li, S.; Pandey, A.; Hooi, B.; Faloutsos, C.; and Pileggi, L. 2021. Dynamic graph-based anomaly detection in the electrical grid. *IEEE Transactions on Power Systems* 37(5):3408–3422.

Lippmann, R.; Haines, J. W.; Fried, D. J.; Korba, J.; and Das, K. 2000. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *Recent Advances in Intrusion Detection: Third International Workshop, RAID 2000 Toulouse, France, October 2–4, 2000 Proceedings 3*, 162–182. Springer.

Liu, Y.; Pan, S.; Wang, Y. G.; Xiong, F.; Wang, L.; Chen, Q.; and Lee, V. C. 2021. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*.

Loshchilov, I., and Hutter, F. 2018. Fixing weight decay regularization in adam.

Manzoor, E.; Milajerdi, S. M.; and Akoglu, L. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1035–1044.

Newman, M. E. 2005. A measure of betweenness centrality based on random walks. *Social networks* 27(1):39–54.

Page, L. 1998. The pagerank citation ranking: Bringing order to the web. technical report. *Stanford Digital Library Technologies Project, 1998*.

Paudel, R., and Eberle, W. 2020. 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)*.

Ranshous, S.; Shen, S.; Koutra, D.; Harenberg, S.; Faloutsos, C.; and Samatova, N. F. 2015. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics* 7(3):223–247.

Ranshous, S.; Harenberg, S.; Sharma, K.; and Samatova, N. F. 2016. A scalable approach for outlier detection in edge streams using sketch-based approximations. In *Proceedings of the 2016 SIAM international conference on data mining*, 189–197. SIAM.

Shin, K.; Hooi, B.; Kim, J.; and Faloutsos, C. 2017. Densealert: Incremental dense-subtensor detection in tensor streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1057–1066.

Tong, H.; Faloutsos, C.; and Pan, J.-Y. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*, 613–622. IEEE.

Velampalli, S., and Eberle, W. 2017. Novel graph based anomaly detection using background knowledge. In *The thirtieth international flairs conference*.

Wang, Y.; Chakrabarti, A.; Sivakoff, D.; and Parthasarathy, S. 2017. Fast change point detection on dynamic social networks. *arXiv preprint arXiv:1705.07325*.

Xie, Y.; Wang, W.; Shao, M.; Li, T.; and Yu, Y. 2023. Multiview change point detection in dynamic networks. *Information Sciences* 629:344–357.

Yang, C.; Zhou, L.; Wen, H.; Zhou, Z.; and Wu, Y. 2020. H-vgrae: A hierarchical stochastic spatial-temporal embedding method for robust anomaly detection in dynamic networks. *arXiv preprint arXiv:2007.06903*.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T. Do transformers really perform bad for graph representation? arxiv 2021. *arXiv preprint arXiv:2106.05234*.

Yoon, M.; Hooi, B.; Shin, K.; and Faloutsos, C. 2019. 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*, 647–657.

Yoon, M.; Jin, W.; and Kang, U. 2018. Fast and accurate random walk with restart on dynamic graphs with guarantees. In *Proceedings of the 2018 World Wide Web Conference*, 409–418.

You, J.; Du, T.; and Leskovec, J. 2022. Roland: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2358–2366.