6-20-2024

AN EMPIRICAL STUDY ON DETECTING AND EXPLAINING GLOBAL STRUCTURAL CHANGE IN EVOLVING GRAPH USING MARTINGALE

Tarun Teja Kairamkonda
Rowan University

AN EMPIRICAL STUDY ON DETECTING AND EXPLAINING GLOBAL STRUCTURAL CHANGE IN EVOLVING GRAPH USING MARTINGALE

by

Tarun Teja Kairamkonda

A Thesis

Submitted to the
Department of Computer Science
College of Science and Mathematics
In partial fulfillment of the requirement
For the degree of
Master of Science in Computer Science
at
Rowan University
May 3, 2024

Thesis Chair: Shen-Shyang Ho, Ph.D., Associate Professor, Department of Computer Science

Committee Members:
Anthony Breitzman, Ph.D., Professor, Department of Computer Science
Guimu Guo, Ph.D., Assistant Professor, Department of Computer Science

Dedications

I dedicate this manuscript to my incredible family, especially to my mother Sujatha Kairamkonda, whose unwavering support, boundless encouragement, and selfless sacrifices have been my guiding light. Their strength, determination, and love have continually inspired me to strive for excellence. Throughout every challenge and triumph, they have been my constant source of motivation. I would not be where I am today without the profound impact they have had on my life.

Acknowledgements

I would like to express my sincere appreciation to Dr. Shen-Shyang Ho for his exceptional guidance and unwavering support throughout my thesis journey. I extend my thanks to the Association for Computing Machinery (ACM) SIGAPP Student Travel Award Program and Rowan University School of Graduate Studies (SGS) for the travel grants that supported my travel to present a conference paper related to this thesis at the 39th ACM/SIGAPP Symposium On Applied Computing in Avila, Spain, and to the Rowan University Computer Science Department for the teaching fellowship, which have been instrumental in supporting my academic pursuits. This thesis was also partially supported by the National Science Foundation (NSF) grant no. 2213658.

Abstract

Tarun Teja Kairamkonda
AN EMPIRICAL STUDY ON DETECTING AND EXPLAINING GLOBAL STRUCTURAL CHANGE IN EVOLVING GRAPH USING MARTINGALE
2023-2024
Shen-Shyang Ho, Ph.D.
Master of Science in Computer Science

There is a growing interest in practical applications involving networks of interacting entities such as sensor networks, social networks, urban traffic networks, and power grids, all of which can be represented using evolving graphs. Changes in these evolving graphs can signify shifts in the behavior of interacting entities or alterations in the patterns of their interactions. Identifying and detecting these changes is crucial for addressing potential challenges or opportunities in various domains. In this study, we propose an approach for detecting structure change in evolving graphs based on the martingale change detection framework on multiple graph features extracted over time. Our research investigates the impact of different graph properties encoded using different feature representations on the evolving graph change detection task. Moreover, performing change detection on multiple graph features allows us to pinpoint which aspects of the graph behavior have changed and provide explanations for identified changes. We demonstrated empirically the robustness of our proposed approach on synthetic evolving graphs generated using different graph-generating algorithms, including scale-free, phase transition, and random small-world evolving graphs. Furthermore, we demonstrate that the proposed martingale approach maintains a desirable false positive bound even when different feature representations are used. Finally, we demonstrate monitoring martingale values for different graph properties on a real-world evolving graph sequence to explain the detected change points.

Table of Contents

Table of Contents (Continued)

List of Figures

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

Chapter 1

Introduction

In the era of interconnected systems and ever-evolving networks, the analysis and understanding of dynamic graph structures have become paramount in various domains which involve interconnected entities such as social networks, cybersecurity, transportation, and biological systems, power systems. Dynamic graph networks, characterized by their changing edges (connections/ interactions) and nodes (objects/ entities) over time, present unique challenges and opportunities for researchers and practitioners alike. One critical aspect of managing dynamic graph networks is the identification and characterization of anomalous or changed patterns, as these deviations often signify potential threats, irregularities, or insights within the evolving system. So it is important to monitor the evolving process to detect unusual changes in patterns either locally or globally [1, 2, 3, 4]

Change detection in evolving graphs involves identifying points in time where the structure or properties of a graph undergo significant changes. For example, In a social media platform like Twitter, a change point might be detected when a viral event occurs. For instance, when a celebrity makes a major announcement, the interaction graph (representing retweets, replies, and mentions) may show a sudden increase in connectivity and activity around that time. Identifying these points can help in understanding the dynamics of information spread and influence within social networks. In a network representing financial transactions between entities, a change point might be detected during a financial crisis or significant market event. For instance, during the 2008 financial crisis, the network of interbank loans showed significant structural changes as banks altered their lending behaviors. Early detection of such changes can provide valuable insights for risk management and regulatory oversight. The idea is not limited to social network and financial networks but also include broad variety of domains including biological networks like in gene regu-

latory networks, detecting a change point might involve identifying when gene expression patterns shift significantly, such as when a disease starts to affect an organism or during a response to a treatment. It is crucial for understanding disease mechanisms and improving treatment strategies.

One particular task of interest related to change detection for evolving graphs is anomaly detection. The detection of anomalies in dynamic graph networks is a multi-faceted problem that requires an understanding of both graph theory and anomaly detection techniques. This thesis aims to study the problem of global structural change detection in dynamic graph networks using a martingale change-detection framework to discern structural pattern change within the dynamic structures of interconnected entities.

The challenges associated with change detection in dynamic graph networks are diverse and include issues such as identifying local changes, maintaining low false positive rates (FPR), and reduction of delay time for the detection of a true positive. In this thesis, we focus on addressing the issues of maintaining low false positive by utilizing a martingale-based framework to detect global structural changes using multiple graph features. Our research delves into dynamic graph networks, extracting diverse features such as node centrality, edge weight fluctuations, temporal patterns, and community structures, to be used for change detection. We utilize various graph embedding techniques tailored to capture different aspects of evolving graph structures.

Performing change detection on multiple graph features allows us to pinpoint which aspects of the graph behavior have changed and provide explanations for identified changes. We demonstrated empirically the robustness and performance of our proposed approach on synthetic evolving graphs generated using different graph-generating algorithms, including scale-free, phase transition, and random small-world evolving graphs. Furthermore, we demonstrate that the proposed martingale approach controls and maintains a desirable false positive rate even when different feature representations are used. In addition, we demonstrate monitoring martingale values for different graph properties on a real-world evolving

graph sequence to explain the detected change points.

By fusing theoretical rigor with practical applicability, our approach aims to advance the state-of-the-art in dynamic graph analysis, offering a valuable tool for researchers, practitioners, and decision-makers across diverse domains. In fact, this research contributes to the broader field of network science and security in a dynamic setting, offering insights that can be applied across diverse domains to enhance decision-making processes and fortify the resilience of interconnected systems.

The main results of this thesis was published in [5]:

- Shen-Shyang Ho and Tarun Teja Kairamkonda, Change Point Detection in Evolving Graph using Martingale, Proceeding of the 39th ACM/SIGAPP Symposium On Applied Computing, Avila, Spain, 2024.

The thesis is organized as follows. In chapter 2, we describe the graph notations and definitions, change detection problem setting, background on martingale method for change detection, modeling of feature distribution for an evolving graph. We provide a brief review of existing change detection methods for evolving graph and explanation techniques for graph-related tasks in chapter 3. In chapter 4, we describe the evolving graph change detection method using martingale and monitoring feature distribution change using martingale and providing feature explanations for detected change. In chapter 5, we present and discuss our results on synthetic data and a demonstration using a real-world dataset. We also compare the performance of our model with existing methodologies using few evaluation metrics . In chapter 6, we conclude by elucidating the applicability of our proposed research, discussing both its strengths and weaknesses within the scope of future studies aimed at enhancing its overall effectiveness and applicability.

Chapter 2

Background

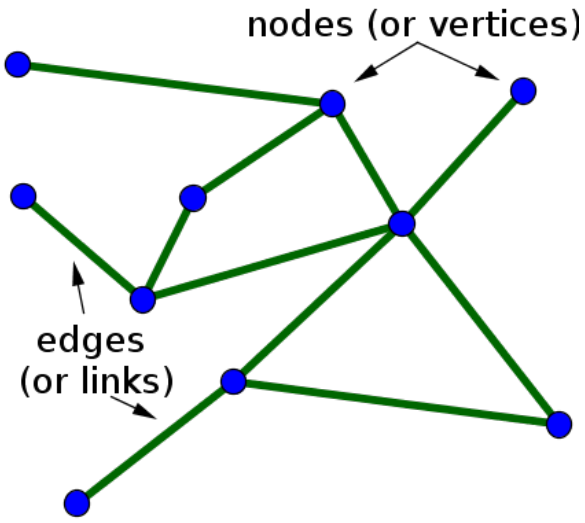
2.1 Graph: Notation and Definitions

Definition 1. *A* **graph** $\mathcal{G}$ *is a pair* (V,E)*, where* V *is a set of vertices (or nodes), and* E *is a set of edges between the vertices* $E \subseteq \{(u,v)|u,v \in V\}$.

A **graph** $\mathcal{G}$ is called a **static** graph if its structure and (node or/and edge) feature values does not change over time. Figure Figure 1 shows an example of a static graph.

Figure 1

A Static Graph Structure Showing Nodes and Edges



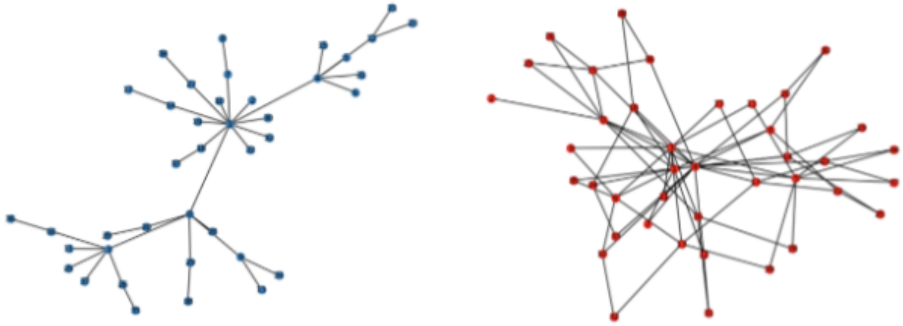
Definition 2. *An* **evolving graph** *is a graph G that changes over time and represented by a sequence of static graphs (or graph snapshots)* $(\mathcal{G}_k)_{1\leq k\leq n}$ *such that the observed time interval length is n.*

A **graph** $\mathcal{G}$ is called a **dynamic** graph if its structure and/or (node or/and edge) feature values change over time. In some situations, the changes are so significant that the

graph structure can be viewed as coming from two different distributions. For example, one can imagine that a graph evolves from a graph structure similar to the left one in Figure 2 to the right one. One can assume that the two graphs are from two different data sources.

Figure 2

Example of Two Graphs Drawn from Two Distributions of Graph of Different Structures.



2.2 Change Detection Problem for Evolving Graph

In this thesis, each graph snapshot is observed one after another, and only once. As a snapshot is observed, graph properties and features are extracted from it.

We define the conditional density of the graph feature of interest by

$$p_\theta(f(\mathcal{G}_k)|f(\mathcal{G}_{k-1}),\ldots,f(\mathcal{G}_1))$$

such that $f : G \to \mathcal{F}$ is a transformation for an input graph G at time instance k and returning an output vector $\mathcal{F}$ (or a matrix, in general). Feature matrix $f(\mathcal{G}_k) \in R^{|V|\times d}$ are extracted from a graph at time instance k such that V is the set of nodes for an evolving graph and d is the feature dimension for each node. In this thesis, we assume that $|V|$ is fixed over time.

The change detection problem involves the identification of a time instance when a deviation from the current data generation model starts within a given data sequence. We formally state our problem statement for evolving graph similar to problem statement in

[6]:

Given a sequence of graph snapshots $(\mathscr{G}_k)_{1\leq k\leq n}$ *with conditional density for a particular graph feature*

$$p_\theta(f(\mathscr{G}_k)|f(\mathscr{G}_{k-1}),\ldots,f(\mathscr{G}_1)).$$

Before change occurs, the conditional density parameter $\theta=\theta_0$. *After the change,* $\theta=\theta_1$. *The online problem is to "detect the occurrence of the change as soon as possible, with a fixed rate of false alarms" [6].*

2.3 Martingale Method for Change Detection

The martingale change detection approach was first proposed in [7] for labeled data stream. It was further extended to handle the change detection problem for regression model and unlabeled data stream [8].

Definition 3. *Given a sequence of random variables* $\{M_i : 0\leq i<\infty\}$. *It is a* **martingale** *M with respect to the sequence* $\{Z_1 : 0\leq i<\infty\}$ *(in particular,* M_0 *is a constant value), if, for all* $i\geq 0$ *the following conditions hold:*

- M_i *is a measurable function of* $Z_0,Z_1,\cdots,Z_i$,
- $E(|M_i|)<\infty$,
- $E(M_{n+1}|Z_0,\cdots,Z_i)=M_n$.

To compute the martingale value at current time instance $t=n$, one used p-values p_t computed from $t=1$ to n. A function $h:X^n\rightarrow[0,1]$ is a *p-value function* with respect to any probability distribution P over X if for all $n\in N$ and $r\in[0,1]$,

$$P_n\{x\in X^n : h(x)\leq r\}\leq r \tag{1}$$

In statistical significance testing, the p-value provides a measure on how well the data discredit the statistical null hypothesis.

A family of martingales, called the *power martingale* [9], indexed by $\varepsilon \in [0,1]$, is defined as

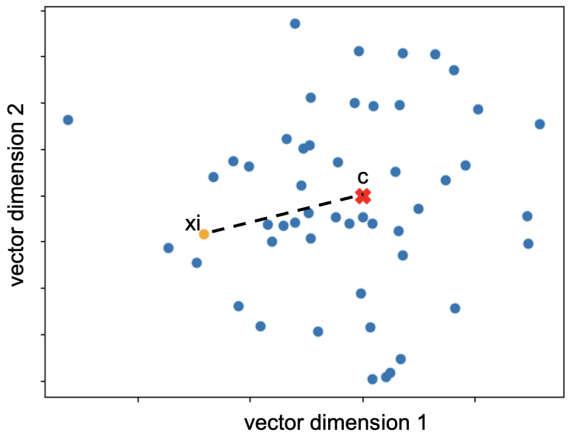
$$M_n^{(\varepsilon)} = \prod_{i=1}^{n} \left(\varepsilon p_i^{\varepsilon-1}\right) \tag{2}$$

where the p_is are the output p-values from the p-value function h, and the initial martingale $M_0^{(\varepsilon)} = 1$.

Vovk et al. [9] proposed the idea of testing exchangeability of streaming data using Equation 2. Ho [7] proposed the use of testing data exchangeability in the streaming data for detecting changes in the data distribution.

Figure 3

An Illustration of Non-Conformity Score for a Graph Snapshot x_i in a 2D Feature Space.



The fundamental building block of the martingale is called the *non-conformality score* which quantify how much a data instance (an observation or a prediction) is different from other data instances [8]. For our change detection problem, a snapshot of the evolving graph is represented by a feature matrix or vector. To compute the non-conformality scores for the snapshots of the graph observed so far represented by s a sequence of features $(f(\mathscr{G}_k))_{1\leq k\leq n}$, we utilize $\mathscr{K}-$mean clustering such that $\mathscr{K} = 1$. Let $C(f(\mathscr{G}_k))_{1\leq k\leq n}$ be the

cluster center. The non-conformality score for graph $\mathscr{G}_k$

$$S(\mathscr{G}_k) = ||f(\mathscr{G}_k) - C(f(\mathscr{G}_k))_{1\leq k\leq n}|| \tag{3}$$

such that $||\cdot||$ is some suitable distance measure (for vector or matrix).

Figure 3 shows an example of a 2D feature space for an evolving graph with each blue point representing a snapshot of the evolving graph. C is the cluster center based on all snapshots and x_i is a new point representing the most recent snapshot. The non-conformity score of x_i is the Euclidean distance between C and x_i.

A p-value p_n at time instance n in Equation 2 is computed using the p-value function,

$$\begin{aligned} & p_n(\{(\mathscr{G}_k)_{1\leq k\leq n}\}, \theta_i) \\ &= \frac{\#\{j : cs_j > cs_n\} + \theta_i \#\{j : cs_j = cs_n\}}{n} \end{aligned} \tag{4}$$

where cs_j is the *non-conformality score* for $\mathscr{G}_j, j = 1,2,\ldots,n$ and θ_n is randomly chosen from $[0,1]$ at time instance n [9]. The computed p-values are independent and uniformly distributed on $[0,1]$ [10]. Using this property, one can show that Equation 2 satisfies the martingale conditions in Definition 3.

Theorem 1. *(***Doob's Maximal Inequality***) Suppose that* $\{M_k : 0 \leq k < \infty\}$ *is a non-negative martingale. Then for any* $\lambda > 0$ *and* $n \in \mathscr{N}$*,*

$$\lambda P\left(\max_{0\leq k\leq n} M_k \geq \lambda\right) \leq E(M_n). \tag{5}$$

If $E(M_n) = E(M_0) = 1$, then from the Doob's Maximal Inequality (Equation 5) one has

$$P\left(\max_{k\leq n} M_k \geq \lambda\right) \leq \frac{1}{\lambda} \tag{6}$$

This inequality means that it is unlikely for any M_k to have a value higher than λ if λ is a big value. However, there is a risk to signal a change detection when there is none. Equation 6 is an upper-bound for the false positive rate for a change detection task given a detection threshold value λ when there is none. The amount of risk one is willing to take for a detection to be a false alarm will determine the λ value to use.

2.4 Graph Representations and Embeddings

As describe earlier in section 2.2, an evolving graph is represented by a sequence of static graph snapshots $(\mathscr{G}_k)_{1\leq k\leq n}$. Each graph snapshot is represented by a feature vector (or matrix).

We define the conditional density of the graph feature of interest by

$$p_\theta(f(\mathscr{G}_k)|f(\mathscr{G}_{k-1}),\ldots,f(\mathscr{G}_1))$$

such that $f : G \to \mathscr{F}$ is a transformation for an input graph G at time instance k and returning an output vector $\mathscr{F}$ (or a matrix, in general). Feature matrix $f(\mathscr{G}_k) \in R^{|V|\times d}$ are extracted from a graph at time instance k such that V is the set of nodes for an evolving graph and d is the feature dimension for each node. For this paper, we assume that $|V|$ is fixed over time.

For simplicity, we assume

$$p_\theta(f(\mathscr{G}_k)|f(\mathscr{G}_{k-1}),\ldots,f(\mathscr{G}_1)) = p_\theta(f(\mathscr{G}))$$

is the density function for some feature vector random variables.

The transformation f represents a great variety of feature extraction or embedding methods. It could be as simple as degree centrality calculation, SVD (Singular value decomposition), and spectral embedding. It can also be more computationally expensive static graph embeddings such as Node2Vec [11] and GraphSAGE [12] or dynamic graph

embeddings [13].

Note that we do not explicitly model the graph feature distribution function for the martingale approach, but we have utilized the well established graph embedding techniques which are available at Networkx[1] and Scikit Network[2] libraries respectively

We briefly describe the seven graph features/embeddings and their graph properties used in our empirical evaluation:

1. **Degree Centrality**:

   Degree centrality is a fundamental measure of node importance in a network, based on the number of connections (or edges) a node has. It quantifies the extent to which a node interacts with other nodes in the network, making it a key indicator of a node's prominence or influence.

   Mathematically, degree centrality $C_d(v)$ of a node v in a network is calculated as the ratio of the number of edges incident to node v to the total number of nodes in the network:

   $$C_d(v) = \frac{\text{degree of node } v}{n}$$

   where:

   - degree of node v represents the number of edges incident to node v.
   - n is the total number of nodes in the network.

   Nodes with higher degree centrality are typically considered more central or influential, as they have a greater number of connections to other nodes. The number of elements in this feature vector equals to the number of nodes in the graph.

[1] networkx, https://networkx.org/documentation/stable/reference/algorithms/index.html
[2] sknetwork, https://scikit-network.readthedocs.io/en/latest/reference/embedding.html

2. **Eigenvector Centrality**:

   Eigenvector centrality is a measure of node importance in a network that takes into account both the number of connections a node has and the importance of those connections. Nodes with high eigenvector centrality are not only connected to many other nodes but are also connected to nodes that themselves have high centrality.

   Mathematically, the eigenvector centrality $C_e(v)$ of a node v in a network is calculated as the eigenvector corresponding to the largest eigenvalue of the adjacency matrix of the network:

   $$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

   where:

   - $\mathbf{A}$ is the adjacency matrix of the network.
   - $\mathbf{x}$ is the eigenvector corresponding to the largest eigenvalue λ.

   The elements of the eigenvector $\mathbf{x}$ represent the eigenvector centrality values of the nodes in the network. Nodes with higher eigenvector centrality values are considered more central or influential in the network.

3. **Betweenness Centrality**:

   Betweenness centrality is a measure of a node's importance in a graph based on the number of shortest paths that pass through it. Nodes with high betweenness centrality act as bridges or connectors between different parts of the graph, facilitating the flow of information or resources. It quantifies the influence of a node in controlling the flow of information or resources between other nodes in the graph. It is particularly useful for identifying key nodes that play crucial roles in maintaining connectivity or facilitating communication within the network.

The betweenness centrality $C_B(v)$ of a node v is calculated as the fraction of shortest paths between all pairs of nodes that pass through v. Mathematically, it is defined as:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where:

- σ_{st} is the total number of shortest paths from node s to node t.
- $\sigma_{st}(v)$ is the number of those shortest paths that pass through node v.

To calculate betweenness centrality for each node in the graph:

(a) Compute all shortest paths between every pair of nodes in the graph using Dijkstra's algorithm [14];

(b) For each node v, count the number of shortest paths that pass through it while iterating over all pairs of nodes.

(c) Calculate the betweenness centrality of node v using the above formula.

Betweenness centrality provides valuable insights into the structural importance of nodes in a graph. Nodes with high betweenness centrality serve as crucial intermediaries or bottlenecks in the network, influencing the flow of information or resources.

4. **Singular value decomposition (SVD) of the adjacency matrix**:

   Singular Value Decomposition (SVD) is a matrix factorization technique that decomposes a matrix into three constituent matrices: $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}^T$. This technique is widely used for dimensionality reduction and feature extraction tasks.

   In the context of graph embedding, SVD is used to project the nodes of a graph onto a lower-dimensional space while preserving the structural information of the graph. This allows for the extraction of meaningful representations of nodes in a lower-

dimensional space, facilitating downstream machine learning tasks such as clustering or classification.

Given an adjacency matrix $\mathbf{A}$ representing the connectivity structure of a graph, SVD decomposes it as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

where:

- $\mathbf{U}$ is an orthogonal matrix containing the left singular vectors.
- $\mathbf{D}$ is a diagonal matrix containing the singular values.
- $\mathbf{V}^T$ is the transpose of an orthogonal matrix containing the right singular vectors.

The SVD embedding process involves the following steps:

(a) **Factorization**: Compute the SVD of the adjacency matrix $\mathbf{A}$ to obtain the matrices $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}^T$.

(b) **Dimensionality Reduction**: Retain only the top k singular values and their corresponding singular vectors to reduce the dimensionality of the embedding space.

(c) **Embedding**: Project the nodes of the graph onto the lower-dimensional space spanned by the selected singular vectors.

The resulting embedding matrix contains the low-dimensional representations of the nodes in the graph, with each row representing the embedding of a node.

5. **Spectral Embedding:**

Spectral embedding provides a powerful framework for capturing the structural properties of a graph in a low-dimensional space. By exploiting the spectral properties of the Laplacian matrix.

In spectral embedding, the Laplacian matrix of a graph is utilized to extract informative features that capture the graph's topology. The Laplacian matrix encapsulates the pairwise relationships between nodes in the graph, making it a suitable basis for spectral embedding. By computing the eigenvectors corresponding to the smallest eigenvalues of the Laplacian matrix, spectral embedding captures the intrinsic structure of the graph in a low-dimensional space.

Given an adjacency matrix $\mathbf{A}$, the Laplacian matrix $\mathbf{L}$ is defined as:

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

where:

- $\mathbf{D}$ is the degree matrix.

The Laplacian matrix $\mathbf{L}$ is then decomposed into its eigenvectors $\mathbf{U}$ and eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, where n is the number of nodes in the graph.

Spectral embedding is performed by selecting the k eigenvectors corresponding to the smallest eigenvalues to form the embedding matrix $\mathbf{X}$. Each row of $\mathbf{X}$ represents the low-dimensional embedding of a node in the graph.

The spectral embedding process involves the following steps:

(a) Compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ of the graph.

(b) Decompose the Laplacian matrix $\mathbf{L}$ to obtain its eigenvectors $\mathbf{U}$ and eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$.

(c) Select the k eigenvectors corresponding to the smallest eigenvalues to form the embedding matrix $\mathbf{X}$.

(d) Each row of $\mathbf{X}$ represents the low-dimensional embedding of a node in the graph.

6. **Singular Values of Laplacian Matrix (LSVD) :**

   Singular Value Decomposition on Laplacian Matrix (LSVD) embedding is a graph factorization technique specifically tailored for dimensionality reduction. It leverages the spectral properties of the Laplacian matrix of a graph to embed nodes into a lower-dimensional space. In LSVD, the singular values of the Laplacian matrix are utilized as the embedding vectors as proposed in Laplacian Anomaly Detection (LAD) [15].

   The LSVD embedding process involves the following steps:

   (a) **Laplacian** : Compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ of the graph as given in item 5

   (b) **Singular value decomposition** :Perform SVD on the Laplacian matrix $\mathbf{L}$ to decompose it into three matrices:$\mathbf{U}$, $\mathbf{D}$ and $\mathbf{V}$ using $\mathbf{L} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ as given in item 4

   (c) **Embedding** : The diagonal elements of the diagonal matrix $\mathbf{D}$ are extracted to form the resulting embedding vector

7. **Node2Vec [11]**:

   Node2Vec is a technique for generating embeddings for nodes in a graph by leveraging concepts from word embedding models like word2vec and adapting them for graph-based data. It learns continuous feature representations (embeddings) for nodes in a graph by optimizing a neighborhood-preserving objective.

   The node2vec embedding process involves the following steps:

   (a) **Random Walks** : Given a source node u, Node2Vec simulates a random walk of fixed length l, where each step in the walk is determined by sampling the next node based on transition probabilities. Formally, let c_i denote the ith node in the walk, starting with $c_0 = u$. The transition probabilities $P(c_i = x | c_{i-1} = v)$ are defined as follows:

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

where π_{vx} represents the unnormalized transition probability between nodes v and x, and Z is the normalizing constant.

(b) **Search Bias** α :To bias the random walks and guide the exploration process, Node2Vec introduces two parameters p and q. These parameters control how fast the walk explores and leaves the neighborhood of the starting node u.

- **Return Parameter** p: Controls the likelihood of immediately revisiting a node in the walk. Setting p to a high value encourages moderate exploration and avoids 2-hop redundancy in sampling. Conversely, a low value of p leads the walk to backtrack a step, keeping it "local" close to the starting node u.
- **In-Out Parameter** q: Allows the search to differentiate between "inward" and "outward" nodes. A value of $q > 1$ biases the walk towards nodes close to the previous node, resembling breadth-first search characteristics. In contrast, $q < 1$ encourages outward exploration, akin to depth-first-search characteristics.

The unnormalized transition probability π_{vx} is further modified by the search bias $\alpha_{pq}(t,x)$, defined as:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

where d_{tx} denotes the shortest path distance between nodes t and x. This search bias guides the random walk based on the distance from the previous node.

(c) **Embedding** : After performing biased random walks and generating node sequences, Node2Vec employs techniques such as Word2Vec to learn embeddings for each node in the graph. These embeddings capture the structural similarities between nodes based on their neighborhood relationships.

The resulting graph embedding matrix or vector provides a compact representation of the entire graph, where each node is represented as a low-dimensional feature vector. These embeddings preserve important graph properties and can be used for various downstream tasks such as node classification, link prediction, and graph visualization.

Chapter 3

Literature Review

3.1 Change Detection in Univariate Time Series

Sequential Probability Ratio Test (SPRT) [16] was proposed to effectively monitor and make decisions about aircraft performance and reliability during flight tests at the time of World War II. The SPRT provided a systematic way to balance the trade-offs between the risks of making incorrect decisions (e.g., false positives or false negatives) and the costs associated with collecting additional data.

Cumulative Sum (CUSUM) control chart [17] aimed to improve the detection of shifts or changes in a process mean. Instead of focusing on individual data points, the CUSUM chart accumulates the deviations of each data point from a target value over time. This cumulative approach allows for the detection of small, incremental shifts in the process mean that may not be easily detected using traditional control charts. CUSUM charts are widely used in quality control and process monitoring to quickly detect changes in a process and take corrective action if necessary.

3.2 Change Detection in Dynamic Network

Previous proposed approaches on network change detection problem focused on converting the sequence of network instances into univariate time series and employing conventional statistical control techniques for analysis [18, 19].

One of the first online probabilistic learning approach for network change detection was proposed by Peel et al. [20]. By combining a generalized hierarchical random graph (GHRG) model with Bayesian hypothesis testing, their approach involves inferring a structural "norm" for interactions across a sequence of graphs and accurately detecting shifts in

this norm over time. The method involves selecting a probability distribution family and a window size, inferring two models representing change and no-change scenarios within the window, and conducting a statistical test to determine the better fit model. Later, De Ridder et al. [21] incorporated implementation of Stochastic Block Model (SBMs) and Bhamidi et al. [22] implemented preferential attachment model alongside GHRG to improve the detection performance.

Huang et al. [15] propose Laplacian Anomaly Detection (LAD), and later, Multi-LAD [23] utilizing Laplacian matrix spectrum for low-dimensional embeddings. LAD explicitly models short and long-term dependencies using two sliding windows. Their method of low dimensional representation of the graph snapshot is implemented in one of our graph features/ embeddings. MultiLAD is introduced as a simple extension of Laplacian Anomaly Detection (LAD) to multi-view graphs.

Aggarwal et al. [24] addresses the problem of differential classification in graph streams, focusing on predicting significant classification events, such as changes in node labels, which may indicate important events or patterns of activity. Unlike static collective classification, this approach emphasizes dynamic and real-time detection of changes in node classification rather than the actual classification of nodes. Constant-curvature Riemannian manifolds (CCMs) [25] provide a non-Euclidean embedding space for graphs. The method uses neural networks and adversarial learning to compute graph embeddings on CCMs and introduces two novel change detection tests operating within this framework, results show this method surpassing approaches based on traditional Eucledian embeddings. DeltaCon [26] handled the graph change detection problem as a lack of similarity based on node/edge characteristics between two graphs. DeltaCon derives features from each snapshot of a dynamic network based on sociological theories and calculates feature similarity between consecutive snapshot pairs. Wang et al. [27] proposed a method using a dissimilarity scoring function, and a threshold to detect events or changes in the evolving graph when the dissimilarity of two consecutive snapshots is above the threshold in the

latent space. Similar to [28] but considered temporal dependencies of snapshots.

3.3 Other Relevant Recent Work

One recent work on detecting anomalous crowd behaviors in videos involved using graph networks [29]. By representing individuals as nodes and their movements as edges in an evolving graph, and utilized max-flow/min-cut optimization which made suitable for real-time video surveillance applications. Utilization of Siamese graph neural network for change-point detection by similarity scores of current graph snapshot and comparing with the recent history was proposed by Sulem et al. [30]. Multi-View Feature Interpretable Change Point Detection (MICPD) [2] employed a vector auto regressive (VAR) model to encode high-dimensional network data into a low-dimensional representation and identify multiple change points by tracking the evolution of multiple targets and their interactions over time.

In recent years, there is a growing interest in the development of explanability and interpretability capabilities for concept drift and change detection in a dynamic environment which the data is streaming [31, 32]. This is an important task for some real-world applications such as manufacturing process [33]. Existing work related to explanability issues for graph-related tasks include the ability to explained learned time-evolving graph embeddings [34], reasoning link prediction on temporal knowledge graph [35] and (mostly) predictions using graph neural network [36, 37, 38]. To the best of our knowledge, there is no existing work on explainable *global* change detection approach to monitor evolving graphs.

Chapter 4

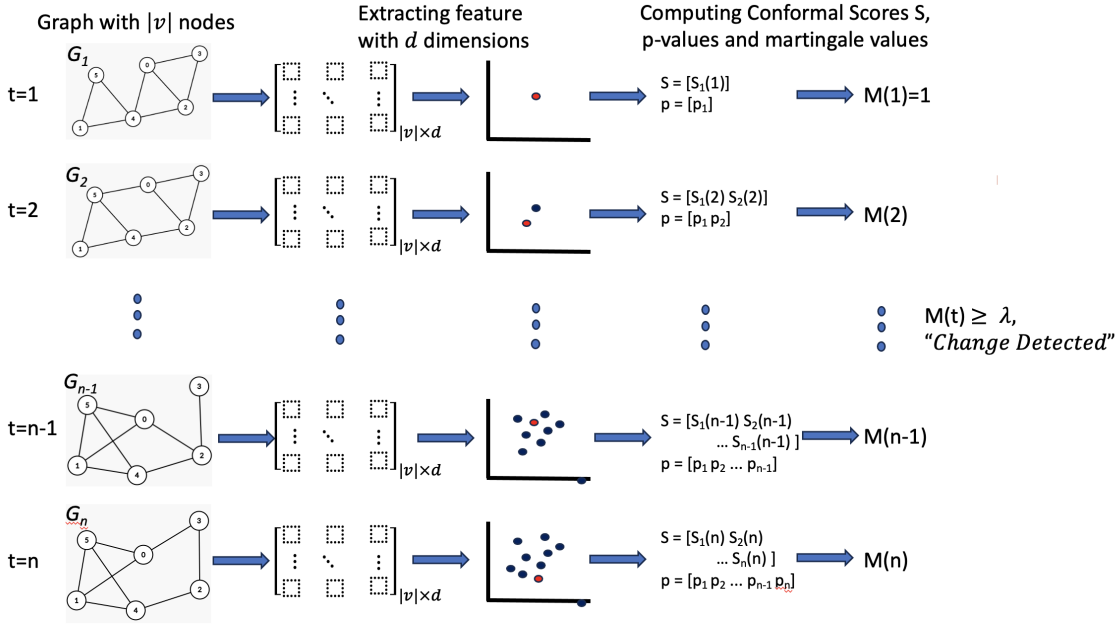
Proposed Methodology

4.1 Overview

Figure 4 shows an overview of the process of using a martingale-based change point detection approach for an evolving graph. Features are extracted from a graph snapshots over time. The extracted features are used to compute the set S of non-conformality scores (different values at different time instance t), p-values and then martingale values at time instance t. Decision to declare a change detected is based on the threshold value λ. When a change is declared, the process is reset (i.e., martingale value is reset to one at the next time instance).

Figure 4

Overview of Evolving Graph Change Detection Using Martingale Over Time



Algorithm 1 Martingale value computation at time step t

Input: Graph $\mathscr{G}_t$

1: $v_t = f(\mathscr{G}_t)$.
2: Compute conformal scores for $(\mathscr{G}_k)_{1\leq k\leq t-1}$ and $\mathscr{G}_t$ using $v_i, i = 1 \leq j \leq t$ based on (Equation 3)
3: Compute the p-value p_t using conformal scores based on (Equation 4);
4: Compute $M(t)$ using (Equation 2) with all previously computed $p_i, i = 1 \leq j \leq t-1$ and p_t;
5: **if** $M(t) \geq \lambda$ **then**
6: **Change Detected;**
7: Alert user of Change Point;
8: Break (from the test process);
9: **else**
10: **Normal (continue the test process)**
11: **end if**

4.2 Martingale-based Change Detection Procedure for Evolving Graph

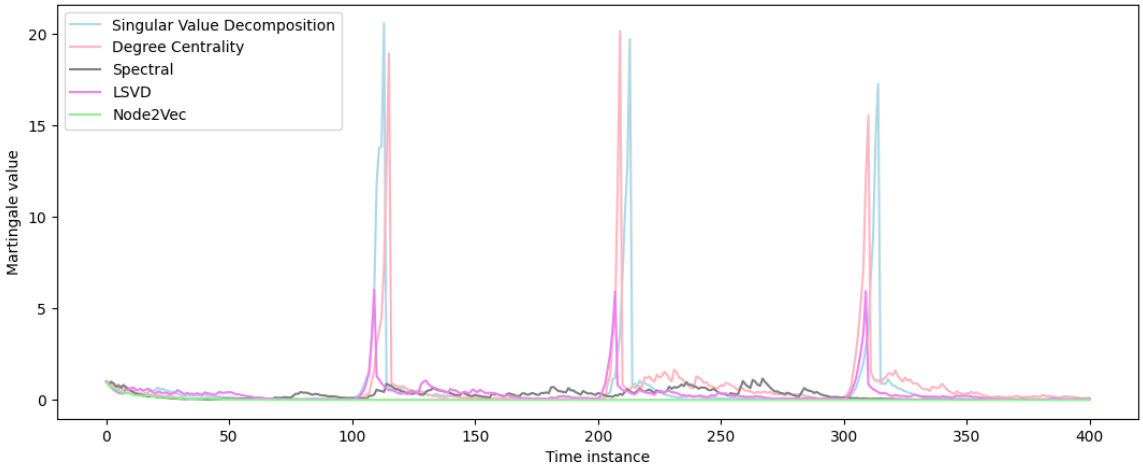
Algorithm 1 shows step-by-step the computation of martingale value for a snapshot of the evolving graph $\mathscr{G}_t$ at time instance t. Line 1 extracts the feature of interest using f on $\mathscr{G}_t$. f is one of the feature extraction methods described in section 2.4. Line 2 computes the conformal scores for all the graph snapshots observed so far using Equation 3. Line 3 computes the p-value p_t using Equation 4 with the conformal scores computed in Line 2. Line 4 computes the martingale value at time t using all previously computed p-values using Equation 2. If the martingale value is greater than a pre-defined threshold λ, the test signals a change detection, alert the user, and stop (or reset) the change detection process. If not, the process will continue to monitor the evolving graph.

4.3 Feature-based Explanation for Detected Change in Evolving Graph using Multiple Graph Features

Figure 10 shows an example of the martingale values of an evolving graph with structural changes occurring at t=100, 200, and 300 for an evolving graph generated using the Barabási–Albert model (see chapter 5 for dataset generation description). The mar-

Figure 5

Monitoring Evolving Graph Feature Distributions Change for Multiple Features Over Time.



tingale value is reset when it is greater than $\lambda = 20$. One observes from the figure that martingale-based change detection using the degree centrality vector, SVD embedding, and LSVD representations are able to detect all 3 change-points. The martingale method is unable to detect the structural change in the evolving graph using spectral embedding and Node2Vec features.

We observe that the five graph features exhibit different time-varying distribution characteristics in the evolving graph. Moreover, there exist some features whose distributions do not show a change or shift even when a structural change occurs. These features do not contain useful information about the particular structural change. Hence, by monitoring the martingale values for different features, one can provide explanation on what type of feature distribution change occurs as a structural change occurs. It provides feature-based explanation on why the change occurs at a particular time instance in the evolving graph.

Chapter 5

Experimental Results

5.1 Synthetic Graph Generation

We briefly describe four types of evolving graph of different characteristics used to validate the proposed martingale approach:

1. **Erdos-Renyi Graph**: The Erdős-Rényi (ER) random graph model, developed by Paul Erdős and Alfréd Rényi [39], is a foundational concept in random graph theory. It describes a range of random graphs where nodes are linked by edges based on a specified probability. There are two main types of ER models: $G(n,p)$ and $G(n,m)$, where n represents the number of nodes, p is the probability of an edge between any pair of nodes (for $G(n,p)$), and m is the total number of edges (for $G(n,m)$). Figure 6 shows an example of Erdős-Rényi type evolving graph with change in p from 0.2 to X at $t = 100$.

   (a) **Edge Formation:**

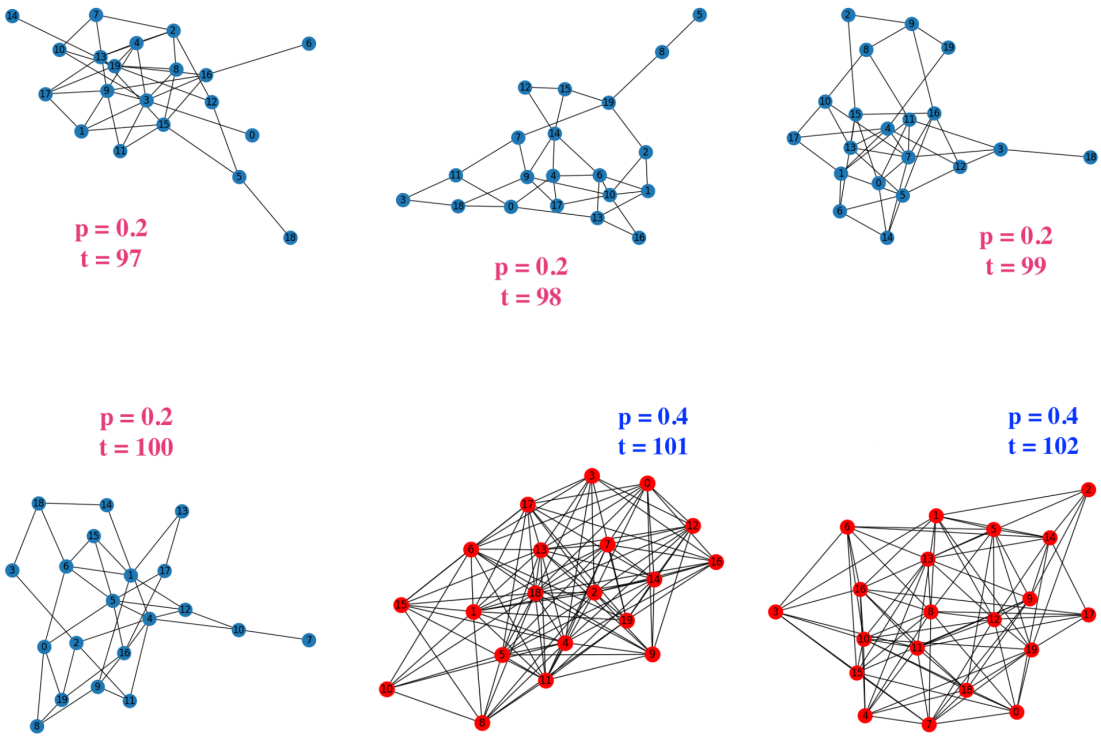
   - For the $G(n,p)$ model, edges are created independently between pairs of nodes with a fixed probability p.
   - For the $G(n,m)$ model, a set number of m edges are randomly added to the graph.

   (b) **Degree Distribution:**

   - The degree distribution of an Erdős-Rényi graph follows a binomial distribution as the number of nodes approaches infinity.
   - The distribution of degrees is typically centered around the expected degree, and most nodes have a degree close to this expected value.

Figure 6

Example of an Erdős-Rényi Type Evolving Graph with Change in p from 0.2 to 0.4 at $t = 100$.



(c) **Phase Transition:**

- For the $G(n,p)$ model, there exists a critical value of p (referred to as p_c) where, when p is below p_c, the graph is usually made up of isolated components. However, when p surpasses p_c, a large connected component emerges that includes a significant portion of the nodes.
- This phenomenon, known as a phase transition, occurs as the graph shifts abruptly from a disconnected state to a connected state as p increases beyond the critical value p_c.

2. **Barabasi-Albert Graph**: The Barabási-Albert (BA) model is a network growth model introduced by Albert-László Barabási and Réka Albert [40]. It's used to create scale-free networks, which have a unique structure where a small number of nodes

have many connections while most nodes have only a few connections. Here's how the BA model works:

(a) **Edge Formation:**

- When a new node is added, it connects to existing nodes in the network based on their degree, or the number of connections they already have.
- Nodes with higher degrees are more likely to receive new connections from the newly added node. This is known as the preferential attachment mechanism.
- The probability of connecting to an existing node v is proportional to its degree k_v. Mathematically, the probability $\Pi(v)$ of connecting to node v is given by:

$$\Pi(v) = \frac{k_v}{\sum_u k_u}$$

where k_v is the degree of node v, and the sum is taken over all existing nodes in the network.

(b) **Degree Distribution:**

- The resulting network from the BA model exhibits a scale-free degree distribution, meaning that the distribution of node degrees follows a power-law distribution.
- This means that while most nodes have only a few connections, there are a small number of nodes, known as hubs, that have a disproportionately large number of connections.

(c) **Graph Representations:**

- The BA model can be represented by the notation $G(n,m)$, where:
  - n represents the total number of nodes in the network.

- m represents the number of hubs (high degree nodes) given at the time of creation. m is typically a fixed number.

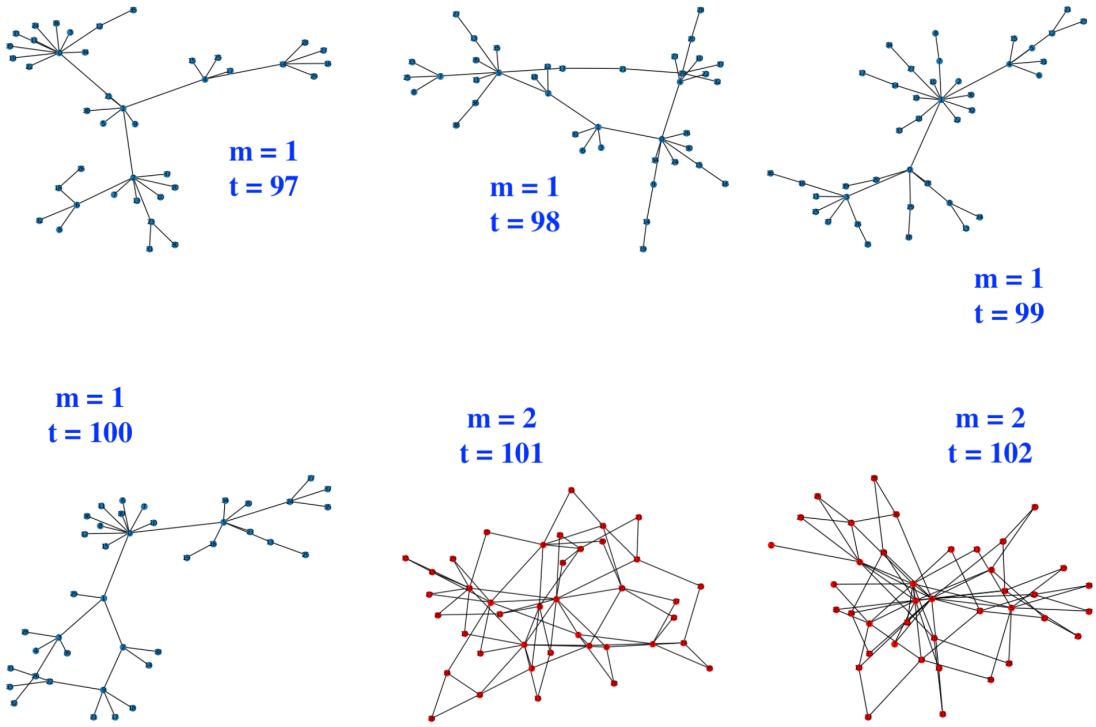
(d) **Network Evolution:**

- As more nodes are added to the network over time, the network structure evolves dynamically, with new nodes preferentially connecting to existing high-degree nodes.
- This continuous growth process leads to the formation of hubs and the emergence of the scale-free property in the network.

Figure 7 shows an example of a Barabasi-Albert type evolving graph with change in m from 1 to 2 at $t = 100$. Figure 8 shows the changes in the feature vector of the centroid of a generated Barabasi Albert-type evolving graph over time with change point at $t = 101$ in the 2D SVD graph feature space.

Figure 7

Example of a Barabási-Albert Type Evolving Graph with Change in m from 1 to 2 at $t = 100$



3. **Spectral Graph Forge (SGF) [41]**: The Spectral Graph Forge (SGF) graph generator is an algorithm designed to create random graphs $G(G_i, \alpha)$ that mimic the global properties of a given input graph G_i. It leverages concepts from spectral graph theory, particularly focusing on the eigenstructure of the graph's adjacency matrix.

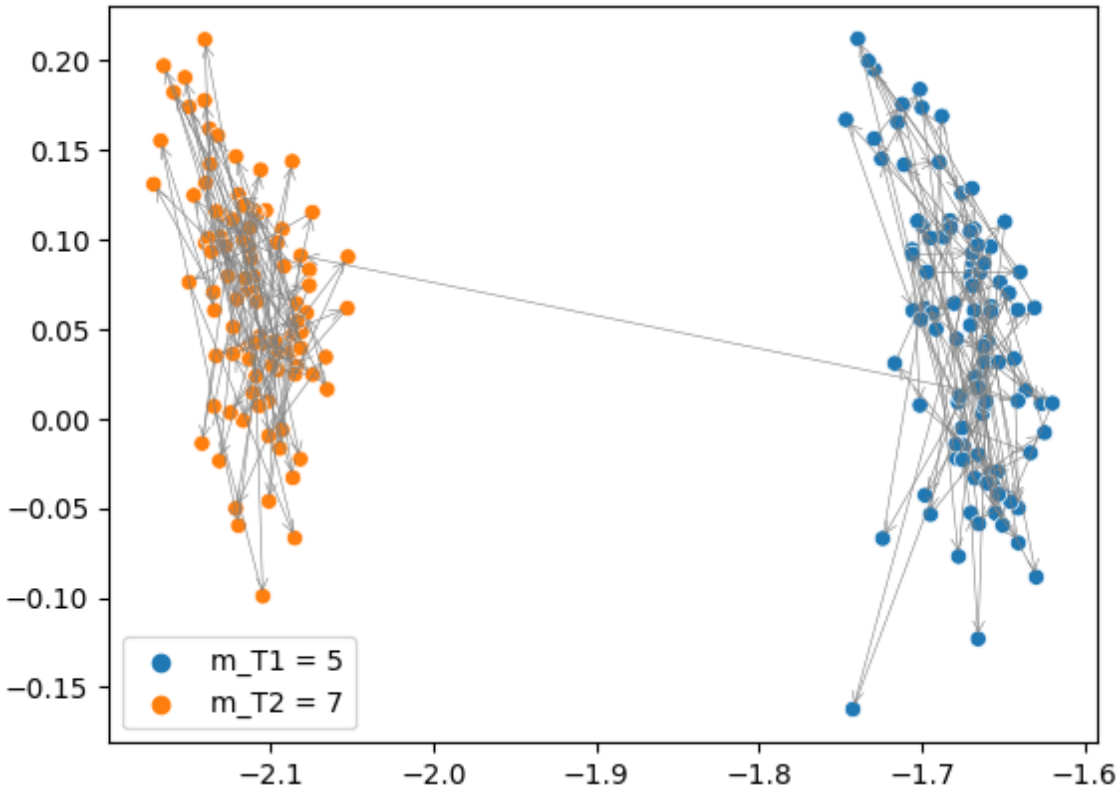
   (a) **Eigenvector Computation:**

   - First, SGF computes the eigenvectors of the adjacency matrix of the input graph.
   - These eigenvectors represent the structural properties of the graph, capturing its connectivity patterns.

     Note: The scale free topology graph is given as the input graph, replicating a social network or citation network

Figure 8

Graph Snapshot Centroids of the Barabási-Albert Evolving Graph in Figure 7 as 2D SVD Graph Feature Vectors from $t = 1$ *to* 200 *with Change Point at* $t = 101$



(b) **Filtering:**

- Next, SGF applies a filtering process to these eigenvectors. This filtering is based on a parameter called α, which controls the level of detail preserved in the eigenvectors.
- It is akin to a low-pass filtering operation, allowing SGF to selectively retain relevant structural information.

(c) **Low-Rank Approximation:**

- SGF then approximates the filtered eigenvectors using a low-rank matrix approximation technique.
- This step helps in reducing the computational complexity and memory re-

quirements while still capturing essential graph properties.

(d) **Random Graph Construction:**

- The filtered and truncated eigenvector values obtained from the low-rank approximation are utilized as inputs for Bernoulli sampling.
- This sampling process constructs a random adjacency matrix for the output graph. By sampling based on the filtered eigenvectors, SGF ensures that the generated graph maintains similar structural characteristics to the input graph.

(e) **Preservation of Properties:**

- SGF preserves the number of nodes and their ordering from the input graph.
- This ensures that the resulting graph closely resembles the properties of the input one, allowing for direct mapping of attributes and retaining the essential structural features.

(f) **Optional Transformations:**

- Additionally, SGF offers options to transform the graph adjacency matrices into other symmetric real matrices, such as identity or modularity matrices. These transformations allow users to focus on specific properties of interest, such as community structure, by altering the underlying structural representation of the graph.

4. **Random Internet Autonomous System Graph**: The Internet Autonomous System (AS) graph is an undirected graph that emulates the structure of real-world Internet AS network. Elmokashfi et al. [42] developed a graph generation model that combines network growth mechanisms and preferential attachment principles for the Internet AS graph.

(a) **Initial Graph Construction:**

- The algorithm $G(n)$ starts by constructing an initial graph structure that represents the topology of an Internet AS network. This initial graph can be generated randomly or based on existing network data. n is the number of graph nodes [100,1000]

(b) **Network Evolution:**

- The initial graph is then evolved using growth mechanisms inspired by real Internet AS networks. This involves iteratively adding nodes and edges while maintaining specific structural properties.

(c) **Preferential Attachment:**

- New nodes added during the evolution process preferentially connect to existing nodes in the graph based on their degree (number of connections). This emulates the tendency of AS networks to form connections with well-connected ASes.

(d) **Scale-Free Degree Distribution:**

- The resulting graph exhibits a scale-free degree distribution, where a small number of nodes (hubs) have many connections, while most nodes have only a few connections. This distribution closely mirrors the connectivity patterns observed in real Internet AS networks.

(e) **Validation:**

- The generated graph is validated against real-world network data to ensure that it accurately captures key structural properties of Internet AS networks, such as node degree distributions and network diameter.

5. **Newmann Watt Strogatz**: The Newman-Watts-Strogatz (NWS) graph generator, often referred to as the Newman-Watts model, is an extension of the classical Watts-Strogatz (WS) small-world network model introduced by Newman and Watts [43].

The NWS model aims to create synthetic networks $G(n,k,p)$ that exhibit both local clustering and short average path lengths, characteristic of small-world networks observed in various real-world systems.

(a) **Ring Construction:**

- The algorithm begins by constructing a ring lattice over n nodes, forming an "n-ring" structure. Each node in the ring is initially connected to its k nearest neighbors, ensuring high local clustering.

(b) **Shortcuts Addition:**

- Shortcuts are introduced to the ring lattice by adding new edges with a certain probability p.
- For each edge (u,v) in the underlying n-ring with k nearest neighbors, a new edge (u,w) is added with a probability p, where w is a randomly chosen existing node in the network.

(c) **No Edge Removal:**

- Unlike some variations of the Watts-Strogatz model, where existing edges may be rewired or removed during the shortcut addition process, the Newman-Watts-Strogatz model does not involve the removal of existing edges. Instead, only new edges are added as shortcuts.

(d) **Transition from Regular to Random Network:**

- Similar to the original Watts-Strogatz model, the Newman-Watts-Strogatz model exhibits a transition from a regular lattice structure to a more random network structure as the probability p of adding shortcuts increases.
- At low values of p, the network retains its regular lattice characteristics with high local clustering.
- As p increases, the network becomes increasingly random, resulting in

shorter average path lengths while preserving a certain level of local clustering.

5.2 Evaluation Measures

1. **False Positive Rate (FPR)**

   False Positive occurs when the system incorrectly classifies a normal instance as an anomaly or change point. FPR is the ratio of false positives to the total number of actual negatives (normal instances). Mathematically, FPR can be expressed as:

$$FPR = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

   FPR is a crucial metric for evaluating the performance of an anomaly detection or change point detection model. A high FPR indicates that the model is incorrectly flagging normal instances as anomalous or signaling a change, which can be problematic for the reliability and usability of the system.

2. **Precision**:

   Precision is the ratio of true positives to the sum of true positives and false positives. It represents the proportion of instances identified as positive (anomalies or change points) that are truly positive. Mathematically, precision can be expressed as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

   Precision is a measure of accuracy that specifically looks at how well the system performs when it claims an instance is positive. It helps answer the question: Of all the instances the system labeled as anomalies or change points, how many were actually anomalies or change points?

In applications where the cost or consequences of false positives are high, precision becomes crucial. A high precision value indicates that the system is adept at minimizing the occurrence of false positives, meaning that when it flags an anomaly or change point, it is likely to be accurate.

3. **Recall**: Recall or Sensitivity is the ratio of true positives to the sum of true positives and false negatives. It represents the proportion of actual anomalies or change points that the system successfully identifies. Mathematically, recall can be expressed as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Recall is focused on ensuring that the system doesn't miss actual anomalies or change points. It answers the question: Of all the actual anomalies or change points, how many did the system correctly identify?

Recall is often considered alongside precision when tuning a model. There is typically a trade-off between precision and recall—if you optimize for high recall, you may experience a decrease in precision, and vice versa.

4. **F1 Score**: The F1 score provides a balance between precision and recall. It considers both false positives (precision) and false negatives (recall) and aims to find a middle ground between these two metrics. Mathematically, the F1 score is calculated as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

For applications where the cost of false positives and false negatives is significant, achieving a high F1 score is desirable. This is especially true in domains such as healthcare, finance, or safety-critical systems, where maintaining a balance between precision and recall is crucial.

5. **Mean Delay Time**: The *delay time* for a detected change point can be defined as the difference in time instances from the start of the change to the time instance when it is detected and within t' time instances. If the change point cannot be detected within t' time instances, it is a *missed detection* or *false negative*.

   Mean delay time is a metric that can be relevant in the context of anomaly detection or change point detection, especially when assessing the timeliness of the detection. It provides insights into the average time it takes for the system to recognize and signal the occurrence of an anomaly or change point after it actually happens. Mathematically, mean delay time can be calculated as:

$$\text{Mean Delay Time} = \frac{\sum_{i=1}^{n}(T_i - A_i)}{n}$$

   where T_i is the time at which the detection system signals the occurrence of the i-th anomaly or change point, A_i is the actual time of occurrence of the i-th anomaly or change point, and n is the total number of anomalies or change points.

5.3 Experimental Design

We performed extensive experiments by

1. varying the difficulty of the change detection task by the amount of change to the key parameter values when change occurs. In particular, δp is the parameter value difference when change occurs for the evolving Erdos-Renyi graph. For example, Figure 9 shows that p increases from 0.4 to 0.6. Hence, $\delta p = 0.2$. For Barabasi-Albert evolving graph, we use the notation δm for m, the number of hubs. For Erdős-Rényi graph parameter δp, we vary from 0.1 to 0.4. Since δp corresponds to the change magnitude and $\delta p = 0.05$ does not registered any detection, our results start with $\delta p = 0.075$. For δm, we vary from 1 to 4.

2. comparing the performance of different graph features. We used the seven described in section 2.4. For SVD embedding, spectral embedding, and node2vec, we use $d = 2$. For degree centrality vector, the number of elements corresponds to the number of nodes in the graph. From Figure 10 and Figure 18, one observe that node2vec, spectral embedding, and betweenness centrality did not perform well in our empirical study (with majority of experiments showing high missed detection rate). Hence, their results are not presented in section 5.4. As there are many results, we only show the representative results for discussion purposes. All other results are found in the Appendix.

3. varying λ to validate the false positive bound derived from Equation 6. We vary λ from 5 to 20, corresponding to a false positive upper bound from 20% reducing to 5%.

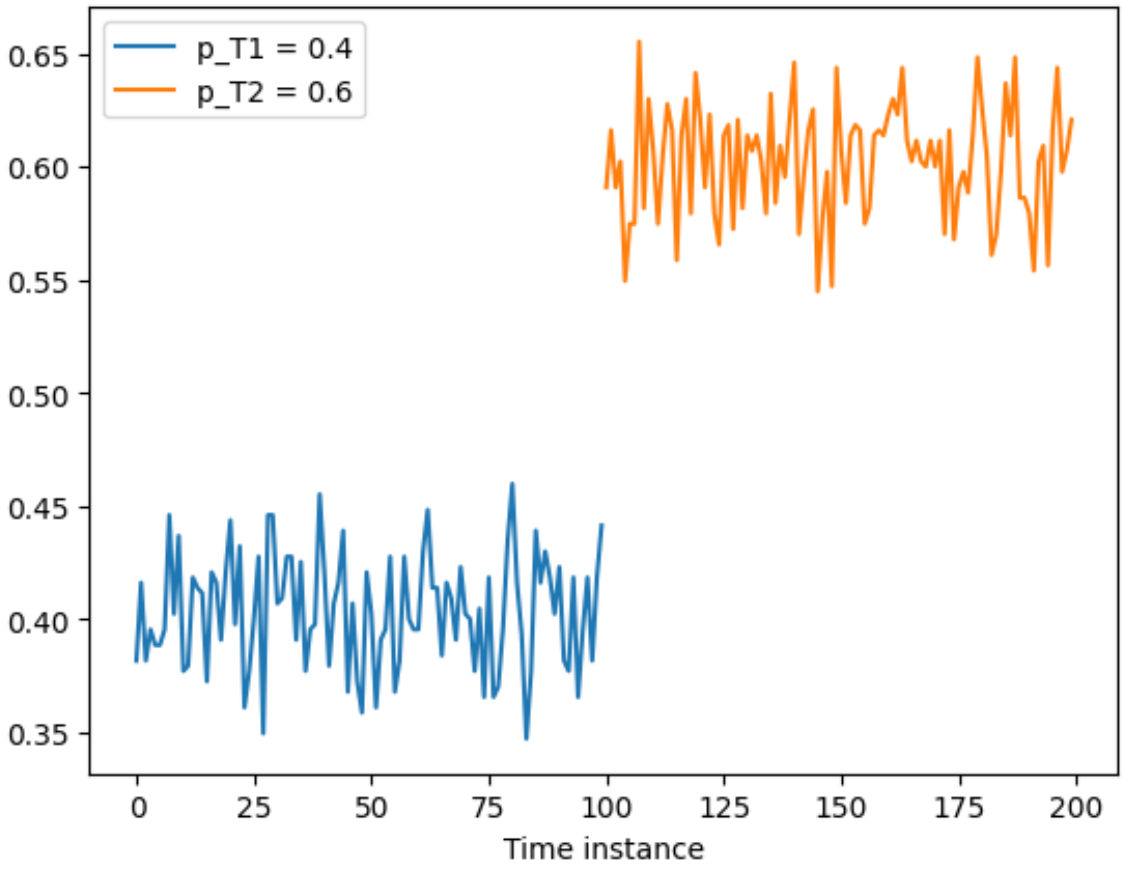
We construct different types of evolving graph with $|V|$ nodes as described in section 5.1. For the first T_1 time instances, we use a graph parameter p_1. For the next T_2 time instances, the parameter is changed to p_2. The change-point occurs at $T_1 + 1$. An experimental trial using an evolving graph with 30 nodes is shown in Figure 9. $T_1 = T_2 = 100$ with change-point at time instance 101. For each experiment with a set of specific feature representation, graph type, and threshold value, we perform 20 trials to obtain precision, recall, F1 measure, and mean delay time before detection.

5.4 Change Detection Empirical Results and Discussions

Figure 10 shows the martingale values of using the seven graph features described in chapter 3 for change detection in an experimental trial on an Erdos-Renyi type evolving graph. From the figure, we observe that not all features were able to detect the change in structure but with varying delay time. By analysis the above figure, we can say that Singular Value Decomposition (SVD), Degree Centrality, Laplacian SVD (LSVD), Eigen-

Figure 9

The Mean Values of Elements in Degree Centrality Feature Vector for $|V|$ Nodes for a Synthetic Evolving Graph Using the Erdos-Rényi Model with Change Point at $t = 101$.



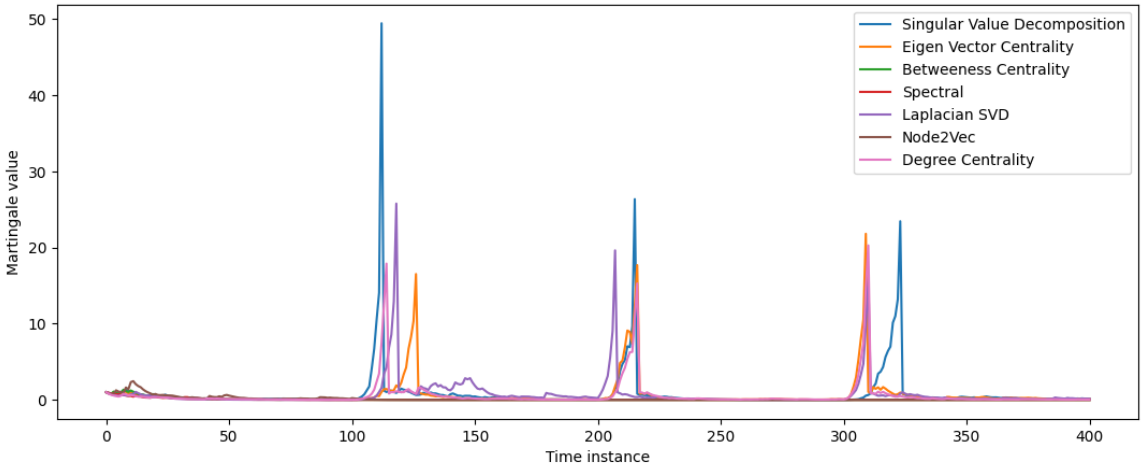
vector centrality are the features which capture changing features in a Erdos-Renyi type dynamic graph. Using SVD and LSVD features, one observes show decreasing spikes in martingale values. However, Node2vec, Betweenness Centrality, and spectral embedding were not able to capture the change points. From these observation, we suspect that the feature representation which encodes a graph property (or characteristics) is critical in the performance of the martingale test for change detection.

5.4.1 The Effect of Threshold λ on False Positive Rate

Figure 11 shows the false positive rates obtained for various thresholds for a Newman-Watts-Strogatz type evolving graph. We observe a decreasing trend in false positive rates as

Figure 10

Monitoring the Martingale Values of the Seven Features of an Erdos-Rényi Type Evolving Graph with Three Change Points



the λ value increases. Similar trend in false positive rates over different thresholds are also noted in other evolving graph types employing different graph embedding techniques. Our empirical results validates the false positive bound (Equation 5) for the martingale change detection method. More results are available in the Appendix.

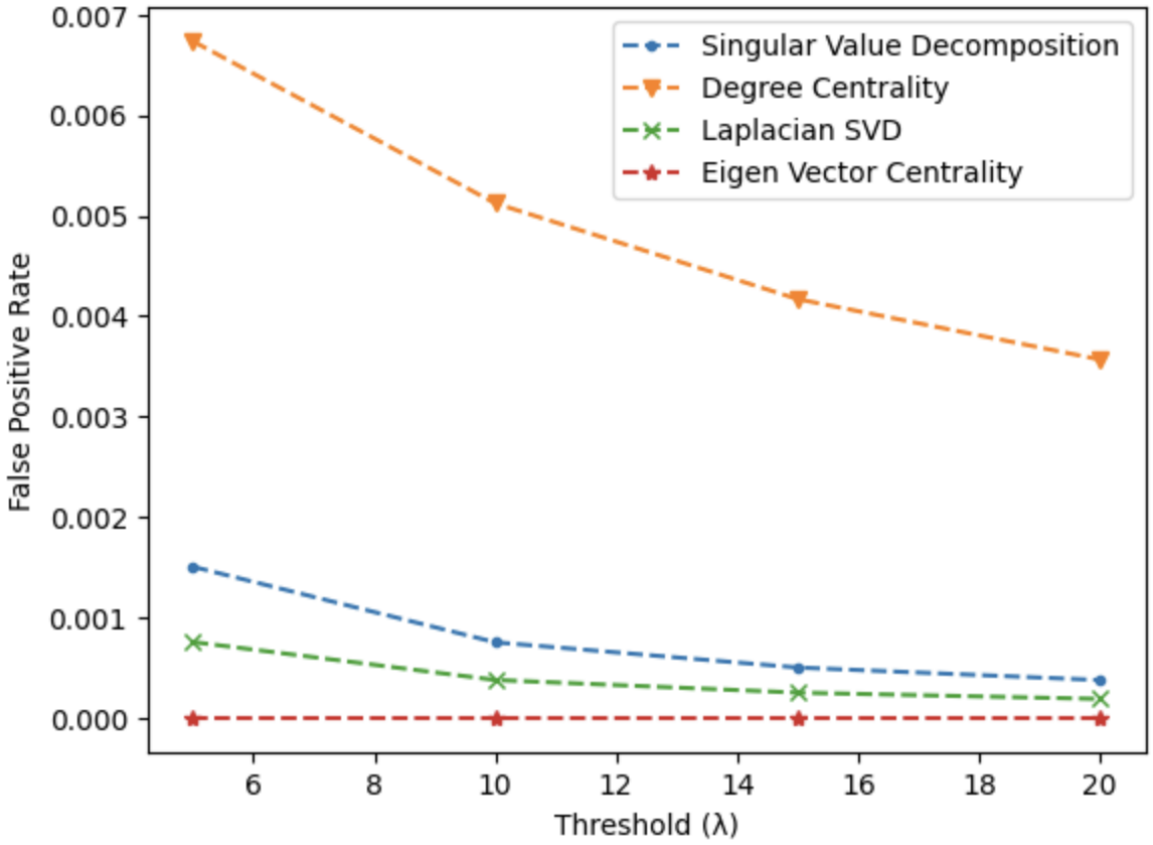
5.4.2 Performance Comparison

Figure 12 shows a multi-line plot on the precision of change detection values for Newman-Watts-Strogatz type evolving graphs utilizing SVD embedding with varying degree of structural change, $\triangle p$, based on the parameter p in an experimental trial. The higher $\triangle p$ indicates easier task as the structural change is more significant. Each curve represents a different degree of structural change with different parameter $\triangle p$. One notes that the harder task with $\triangle p = 0.1$ has the lowest precision across all λ values.

One also observes from Figure 12 that the precision increases with increase λ value. This aligns with theoretical expectations, as reducing false positives should theoretically lead to an increase in precision. The trend observed in precision further supports this theory, indicating improved in precision performance measure as false positive rate is minimized.

Figure 11

False Positive Rate Against λ on Four Graph Features on a Newman-Watts-Strogatz Type Evolving Graph



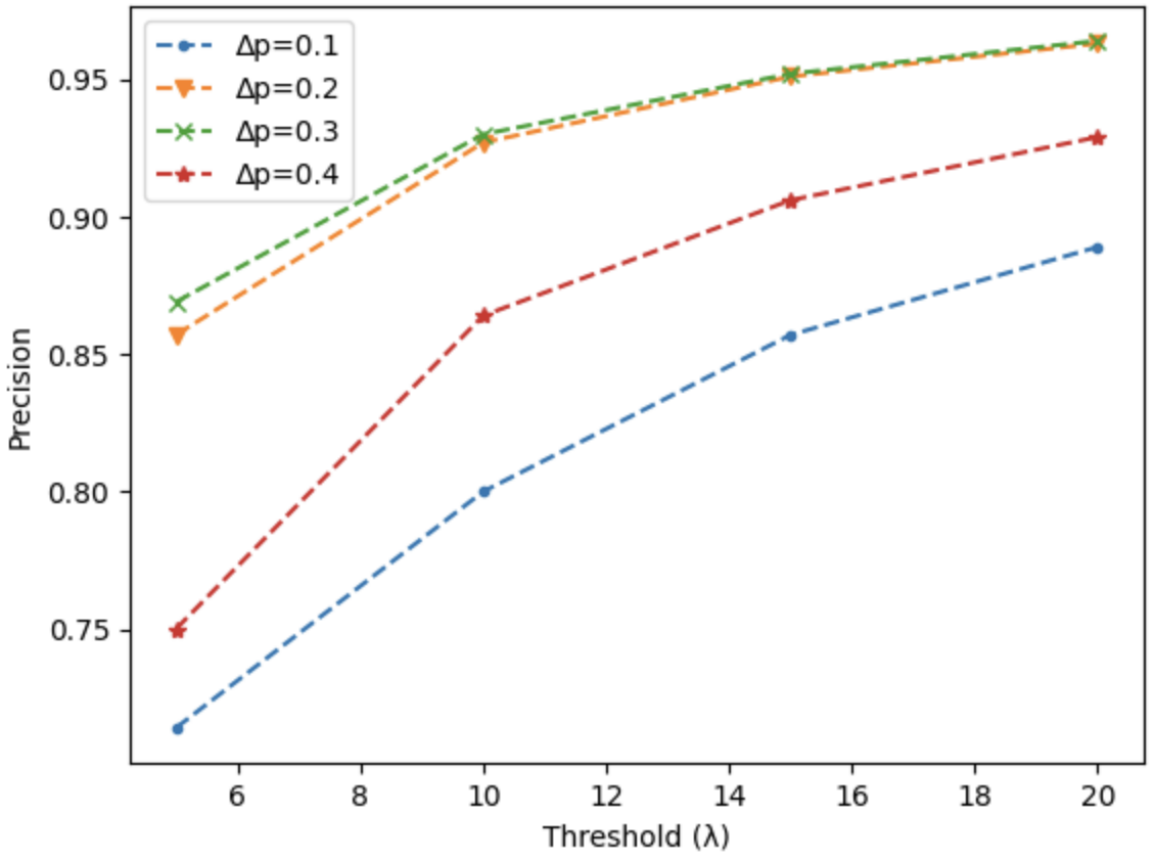
The analogous trend is evident across other evolving graph types and other feature types. More results are found in the Appendix.

We have observed a consistent trend of increasing recall in Figure 13, indicating that our model is adept at capturing change points within evolving graphs. The martingale approach effectively captures a large proportion of the true change point instances among all actual change point instances in an evolving graph. This pattern persists across different type of evolving graphs and features. These findings suggest that our approach effectively detects changes in the graph structure as they occur.

To illustrate that some features may not work well for change detection for specific type of evolving graphs, we use the SVD feature on four different types of evolving graph as

Figure 12

Precision Against λ Using SVD Feature on Newman-Watts-Strogatz (NWS) Type Evolving Graphs with Varying Degree of Structural Change.

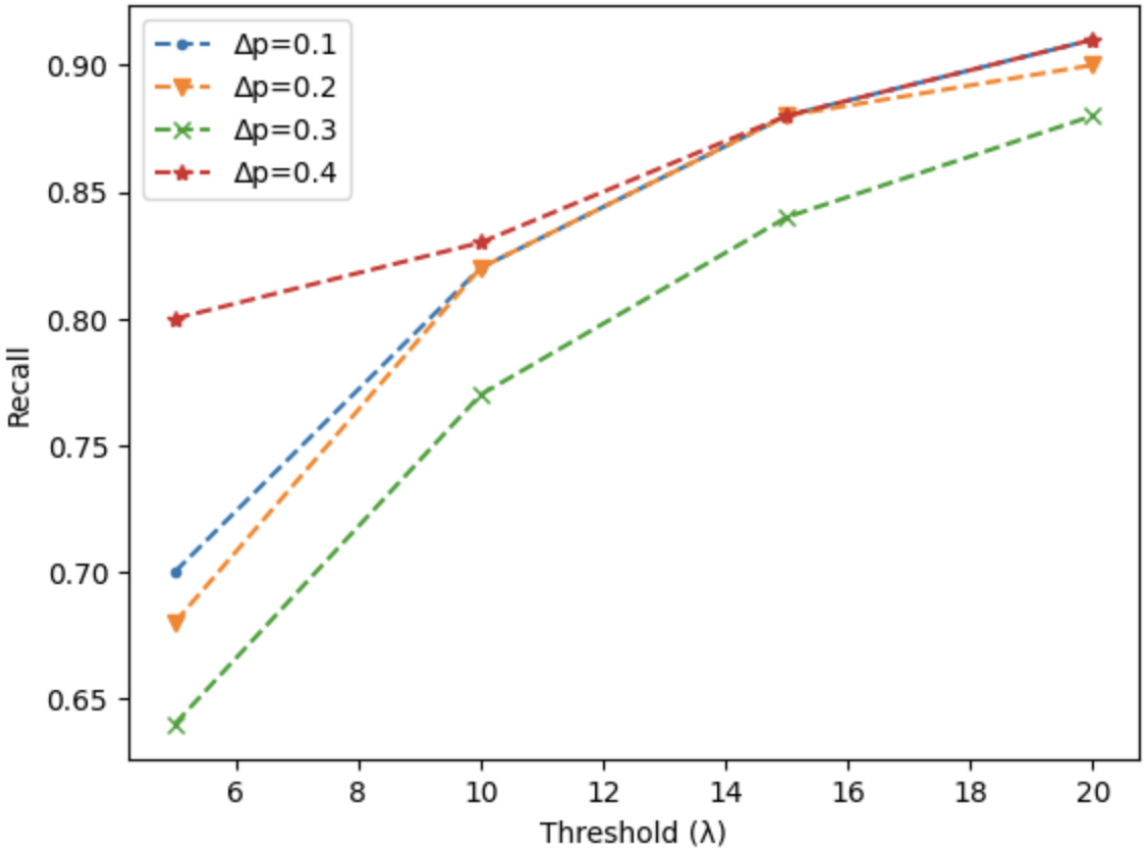


shown in Figure 14. While we observe an increasing trend in recall across different types of evolving graph, it is noteworthy that the recall for Barabasi-Albert and Erdos-Renyi model graphs consistently remain above 0.90. On the other hand, the recall for Newman-Watts-Strogatz and Barabasi-Albert-Internet-Graph are approximately half of those values. From this empirical result, one concludes that one needs to select feature type carefully for different applications to ensure good change detection performance.

We have noticed a consistent trend in F1 scores similar to that of precision and recall. This observation in Figure 15 is expected, as the F1 score is the harmonic mean of precision and recall, effectively balancing both metrics. In terms of the actual values, we found that the F1 scores for change detection on Erdos-Renyi and Barabasi-Albert type

Figure 13

Recall Against λ Using LSVD Features on Erdos-Rényi Type Evolving Graph with Varying Degree of Structural Change.

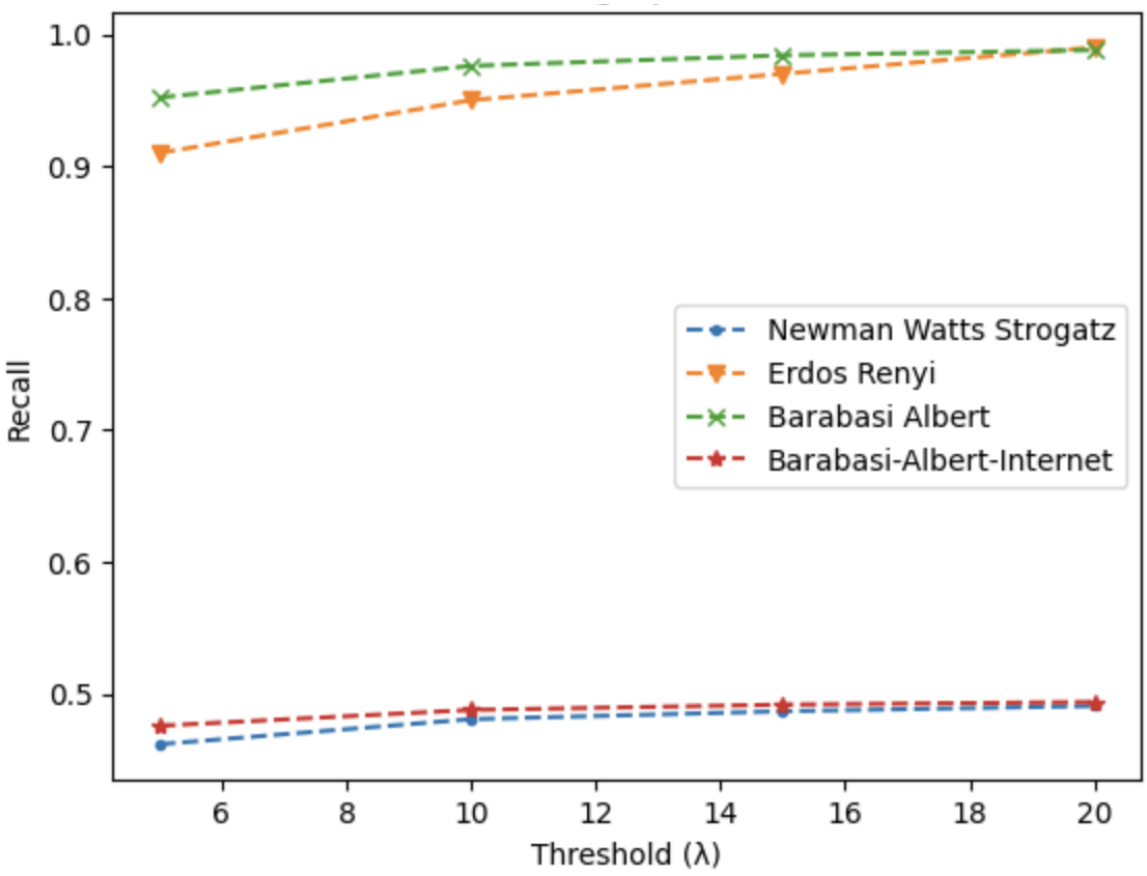


evolving graphs consistently exceeding when compared to F1 scores on Newman-Watts-Strogatz and Barabasi-Albert Internet evolving graphs . These findings underscore the significance of the feature extraction method in change detection, particularly in relation to the type of applications.

The increasing trend in mean delay time with higher threshold values aligns with the practical intuition that it takes longer time for the martingale value to reach the threshold value λ. As the threshold for change detection rises, the model necessitates more data to exceed this threshold and identify change-points. Consequently, higher thresholds lead to more delays in change detection. This underscores the critical role of threshold selection in change-point detection algorithm, as higher thresholds may result in delayed anomaly

Figure 14

Recall Against λ Threshold Using SVD Features on Different Types of Evolving Graphs.



detection which can be observed in Figure 16 and Figure 17.

5.5 Monitoring Real World Data and Change Explanations

5.5.1 MIT Social-Network Evolution Dataset Description

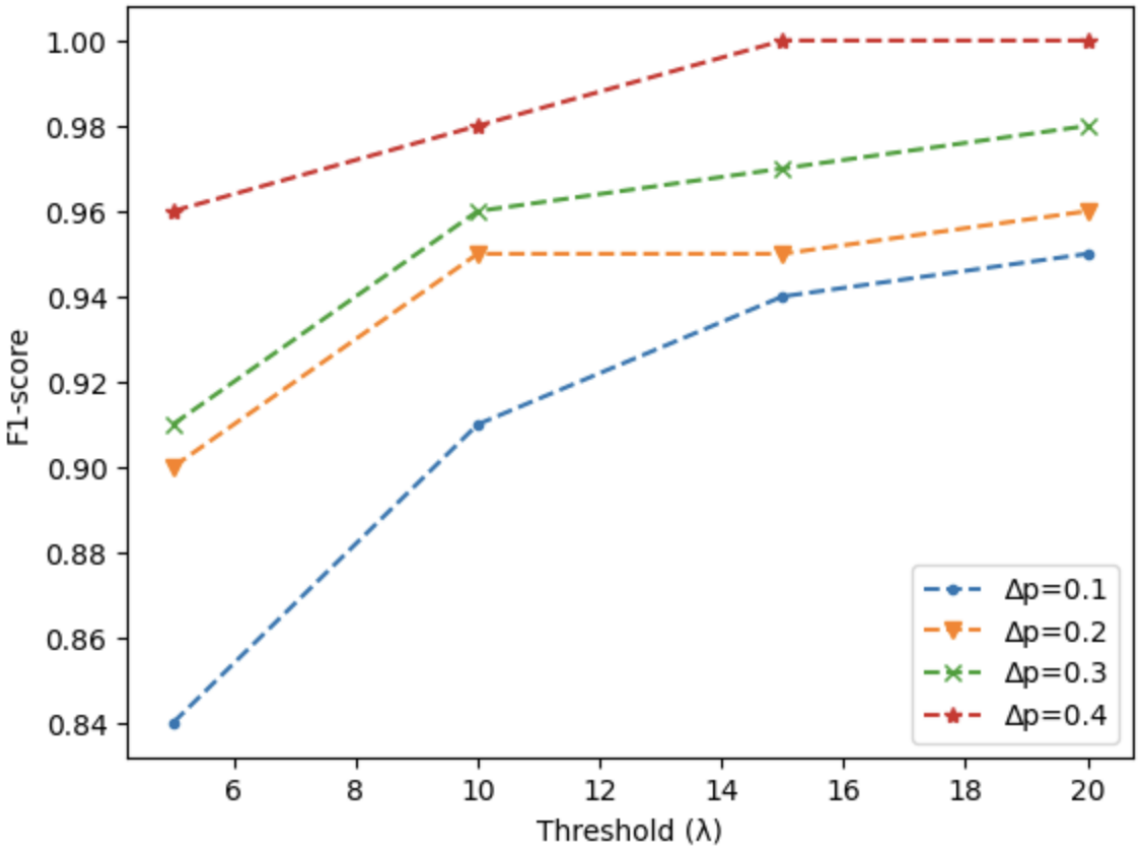
The dataset utilized in this study originates from the social evolution experiment conducted by MIT[1], designed to track everyday interactions within a closely-knit student dormitory. This experiment furnished invaluable data for validating machine learning models against spatio-temporal patterns and behavior-network co-evolution.

The dataset comprises proximity data collected via a cell-phone application, which

[1]MIT Social Evolution Experiment, http://realitycommons.media.mit.edu/socialevolution2.html

Figure 15

F1 Measure Against λ Using Degree Centrality Feature on Barabási-Albert Type Evolving Graph.

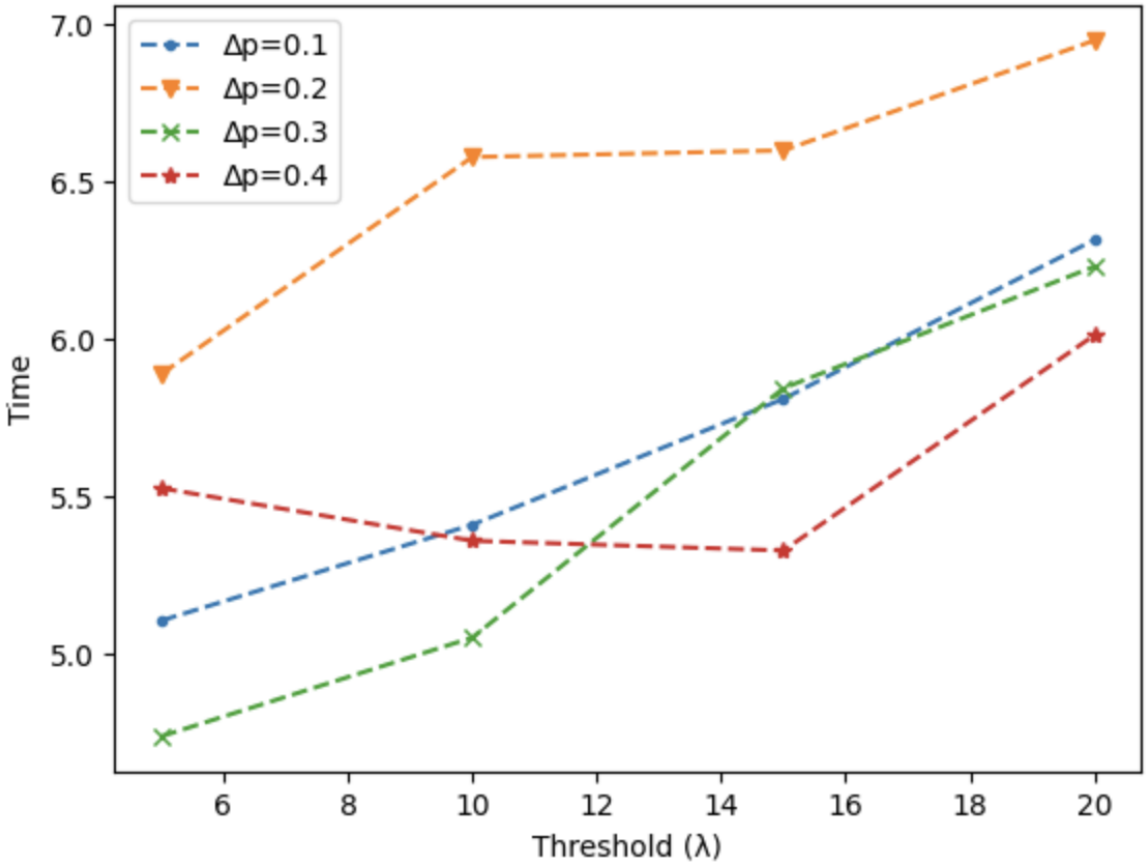


periodically scans nearby Wi-Fi access points and Bluetooth devices at six-minute intervals. This proximity data offers insights into the dynamic interactions and relationships among individuals within the dormitory community.

Interactions among students were captured through their cell phones from October 2008 to May 2009, spanning a substantial temporal duration. The dormitory under scrutiny encompassed approximately 80 members, including 30 freshmen, 20 sophomores, 10 juniors, 10 seniors, and 10 graduate student tutors. These interactions are represented as edges, with individuals serving as nodes, thereby forming an evolving graph over time—a prime scenario for our test case.

Figure 16

Mean Delay Time Measure Against λ Using Eigenvector Centrality Feature on Internet Autonomous Systems Type Evolving Graph.



5.5.2 Results and Discussions

We monitored the martingale values for each snapshot of the evolving graph and observed spikes in the martingale values for different graph features as shown in Figure 18. These spikes indicate change or abnormal points in time, where our change detection model claims that a property of the graph has changed at specific time of the year. Upon analysis, we found that many of these detected change points coincided with significant events in the MIT academic calendar, such as federal holidays like Columbus Day, Thanksgiving, New Year, Christmas, and summer break. Figure Figure 19 and Figure 20 shows the graph structure before and after change detection over the Thanksgiving day period and the New Year day period, respectively.

Figure 17

Mean Delay Time Measure Against λ Using SVD Feature on All Types of Evolving Graphs.

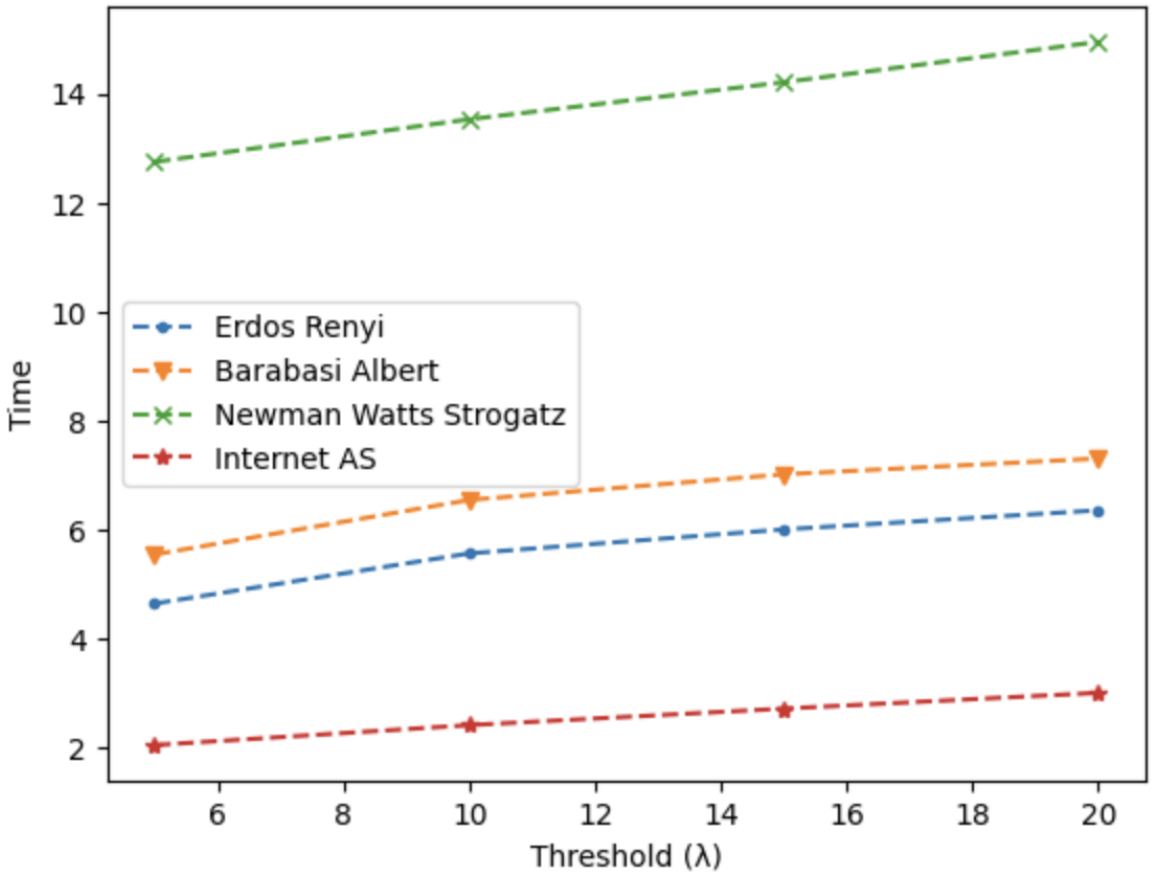
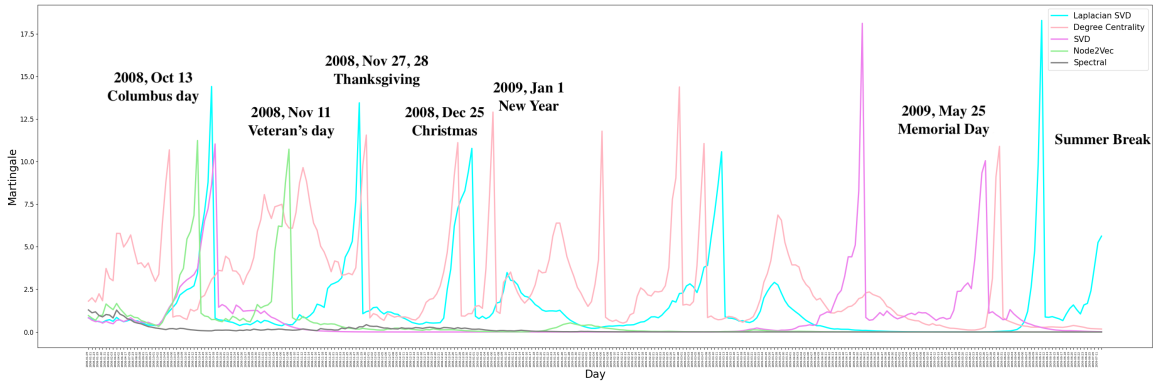


Figure 18

Monitoring Martingale Values Over Time for the MIT Social Evolution Experiment Dataset



However, there were instances where spikes occurred that could not be directly explained by the academic calendar alone. Despite this, we still posit that some of these

spikes represent genuine change points, albeit unrecorded in the calendar. This assertion is supported by the exceptional performance of certain feature embeddings, such as SVD and Degree centrality, with synthetic data, which correctly identified change points.

It is noteworthy that the Node2Vec embedding, which exhibited poor performance with synthetic data, successfully detected two change points—Columbus Day and Veterans Day—highlighting the influence of graph type on embedding behavior.

Regarding the unexplained change points, it is possible that some of them are false positives. However, further analysis is needed, and additional event or calendar data may help elucidate these occurrences. Failure to explain these change points may result in them being classified as false positives.

Figure 19

Visualization of the Graph Structure Around Thanksgiving Day for the MIT Social Evolution Experiment Dataset.

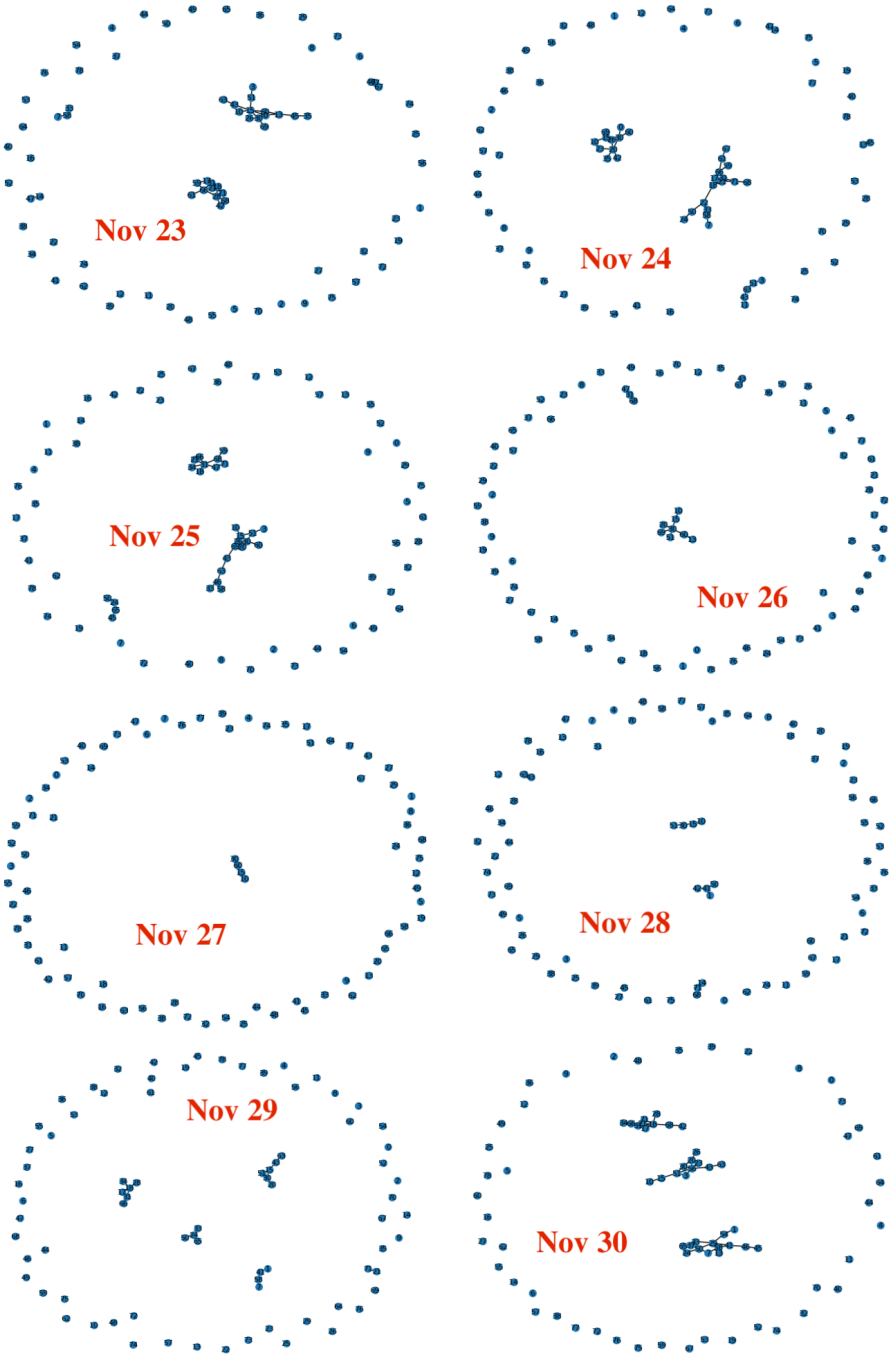
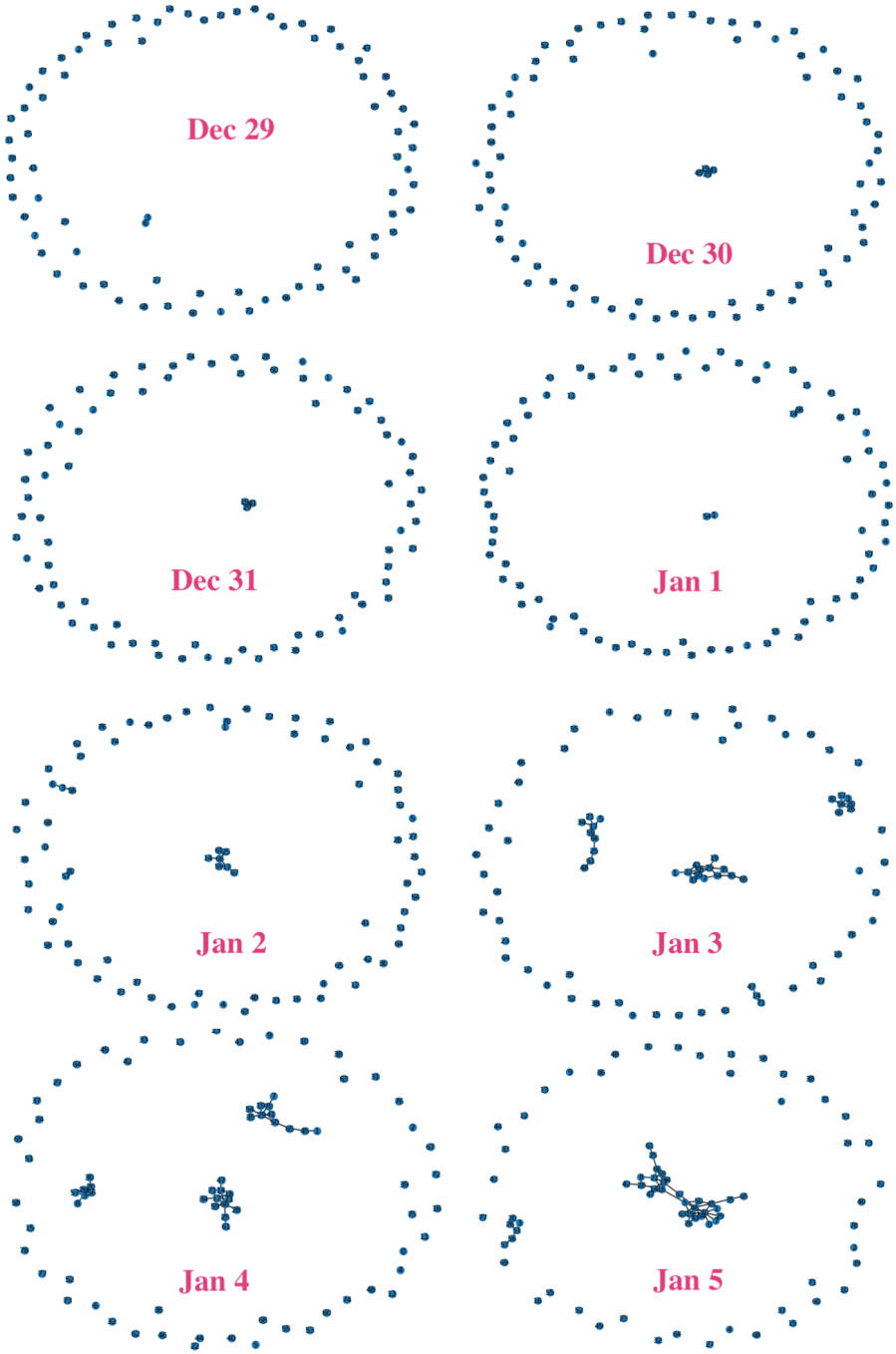


Figure 20

Visualization of the Graph Structure Around the New Year Day for the MIT Social Evolution Experiment Dataset.



Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we describe a martingale change-point detection approach for evolving graph by monitoring the martingale values derived from multiple graph features. We demonstrate empirically that the feature representation that encodes a graph property (or characteristics) is critical in the performance of the martingale test in detecting the change-point using synthetic evolving graphs created from four graph (random topology, scale-free, small-world network) generators and a real-world social-network dataset.

With the continuous growth of graph data and the increasing complexity of real-world networks, there are numerous avenues for future research and development. As the size and complexity of graph datasets continue to increase, scalability becomes a critical consideration. Future research should focus on developing algorithms and computational techniques that can efficiently process large-scale graph data without compromising accuracy or performance.

6.2 Future Work

We plan to conduct comparative studies with recently proposed approaches for change-point detection in evolving graphs. By benchmarking against state-of-the-art methods, we aim to identify strengths, weaknesses, and potential areas for improvement in existing techniques. We intend to extend our work by incorporating additional graph features beyond those considered in this study. There is immense potential for applying change-point detection techniques to a wide range of real-world applications. Future research should explore the practical applications of change-point detection in various domains.

We plan to develop graph generating algorithm that creates large size graph with varying number of nodes over time. An interesting avenue for future research involves developing predictive models to reduce delay time in detection.

References

[1] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, "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*, 2019, pp. 647–657.

[2] Y. Xie, W. Wang, M. Shao, T. Li, and Y. Yu, "Multi-view change point detection in dynamic networks," *Information Sciences*, vol. 629, pp. 344–357, 2023.

[3] S. Roberts, "Control chart tests based on geometric moving averages," *Technometrics*, vol. 42, no. 1, pp. 97–101, 2000.

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

[5] S.-S. Ho and T. T. Kairamkonda, *Change point detection in evolving graph using martingale*, 2024.

[6] M. Basseville, I. V. Nikiforov, *et al.*, *Detection of abrupt changes: theory and application*. prentice Hall Englewood Cliffs, 1993, vol. 104.

[7] S.-S. Ho, "A martingale framework for concept change detection in time-varying data streams," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 321–327.

[8] S.-S. Ho and H. Wechsler, "A martingale framework for detecting changes in data streams by testing exchangeability," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 12, pp. 2113–2127, 2010.

[9] V. Vovk, I. Nouretdinov, and A. Gammerman, "Testing exchangeability on-line," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 768–775.

[10] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. Berlin, Heidelberg: Springer-Verlag, 2005, ISBN: 0387001522.

[11] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[13] C. D. Barros, M. R. Mendonça, A. B. Vieira, and A. Ziviani, "A survey on embedding dynamic graphs," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–37, 2021.

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

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

[16] A. Wald, "Sequential analysis," *john wiley & sons, New York, NY*, 1947.

[17] E. Page, "Cumulative sum charts," *Technometrics*, vol. 3, no. 1, pp. 1–9, 1961.

[18] C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park, "Scan statistics on enron graphs," *Computational & Mathematical Organization Theory*, vol. 11, pp. 229–247, 2005.

[19] I. McCulloh and K. M. Carley, "Detecting change in longitudinal social networks," *Journal of social structure*, vol. 12, no. 1, pp. 1–37, 2011.

[20] L. Peel and A. Clauset, "Detecting change points in the large-scale structure of evolving networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[21] S. De Ridder, B. Vandermarliere, and J. Ryckebusch, "Detection and localization of change points in temporal networks with the aid of stochastic block models," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2016, no. 11, p. 113 302, 2016.

[22] S. Bhamidi, J. Jin, and A. Nobel, "Change point detection in network models: Preferential attachment and long range dependence," *The Annals of Applied Probability*, vol. 28, no. 1, pp. 35–78, 2018.

[23] S. Huang, S. Coulombe, Y. Hitti, R. Rabbany, and G. Rabusseau, "Laplacian change point detection for single and multi-view dynamic graphs," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 3, pp. 1–32, 2024.

[24] C. C. Aggarwal, Y. Li, and P. S. Yu, "On supervised change detection in graph streams," in *Proceedings of the 2020 SIAM International Conference on Data Mining*, SIAM, 2020, pp. 289–297.

[25] D. Grattarola, D. Zambon, L. Livi, and C. Alippi, "Change detection in graph streams by learning graph embeddings on constant-curvature manifolds," *IEEE Transactions on neural networks and learning systems*, vol. 31, no. 6, pp. 1856–1869, 2019.

[26] D. Koutra, N. Shah, J. T. Vogelstein, B. Gallagher, and C. Faloutsos, "Deltacon: Principled massive-graph similarity function with attribution," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 3, pp. 1–43, 2016.

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

[28] R. S. Caceres and T. Berger-Wolf, "Temporal scale of dynamic networks," in *Temporal networks*, Springer, 2013, pp. 65–94.

[29] M. Yang, Y. Feng, A. S. Rao, S. Rajasegarar, S. Tian, and Z. Zhou, "Evolving graph-based video crowd anomaly detection," *The Visual Computer*, vol. 40, no. 1, pp. 303–318, 2024.

[30] D. Sulem, H. Kenlay, M. Cucuringu, and X. Dong, "Graph similarity learning for change-point detection in dynamic networks," *arXiv preprint arXiv:2203.15470*, 2022.

[31] J. Haug, A. Braun, S. Zürn, and G. Kasneci, "Change detection for local explainability in evolving data streams," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 706–716.

[32] F. Fumagalli, M. Muschalik, E. Hüllermeier, and B. Hammer, "Incremental permutation feature importance (ipfi): Towards online explanations on data streams," *Machine Learning*, vol. 112, no. 12, pp. 4863–4903, 2023.

[33] H. Choi, D. Kim, J. Kim, J. Kim, and P. Kang, "Explainable anomaly detection framework for predictive maintenance in manufacturing systems," *Applied Soft Computing*, vol. 125, p. 109 147, 2022.

[34] Z. Liu, D. Zhou, and J. He, "Towards explainable representation of time-evolving graphs via spatial-temporal graph attention networks," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 2137–2140.

[35] Z. Han, P. Chen, Y. Ma, and V. Tresp, "Explainable subgraph reasoning for forecasting on temporal knowledge graphs," in *International Conference on Learning Representations*, 2020.

[36] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 772–10 781.

[37] D. Luo *et al.*, "Parameterized explainer for graph neural network," *Advances in neural information processing systems*, vol. 33, pp. 19 620–19 631, 2020.

[38] J. Tang, L. Xia, and C. Huang, "Explainable spatio-temporal graph neural networks," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 2432–2441.

[39] P. Erdős, A. Rényi, *et al.*, "On the evolution of random graphs," *Publ. math. inst. hung. acad. sci*, vol. 5, no. 1, pp. 17–60, 1960.

[40] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[41] L. Baldesi, C. T. Butts, and A. Markopoulou, "Spectral graph forge: Graph generation targeting modularity," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 1727–1735.

[42] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the scalability of bgp: The role of topology growth," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1250–1261, 2010.

[43] M. E. Newman and D. J. Watts, "Renormalization group analysis of the small-world network model," *Physics Letters A*, vol. 263, no. 4-6, pp. 341–346, 1999.

Appendix A

Additional Results - False Positive Rates

Figure 21, Figure 22, and Figure 23 show the false positive rates vs λ varying from 5 to 20 using different graph features for the martingale method on Erdos-Renyi type graphs, Barabasi-Albert type graphs, and Internet AS type graphs, respectively .

Figure 21

False Positive Rate vs λ for Using Different Graph Features for the Martingale Methods on Erdos-Rényi Type Graphs.

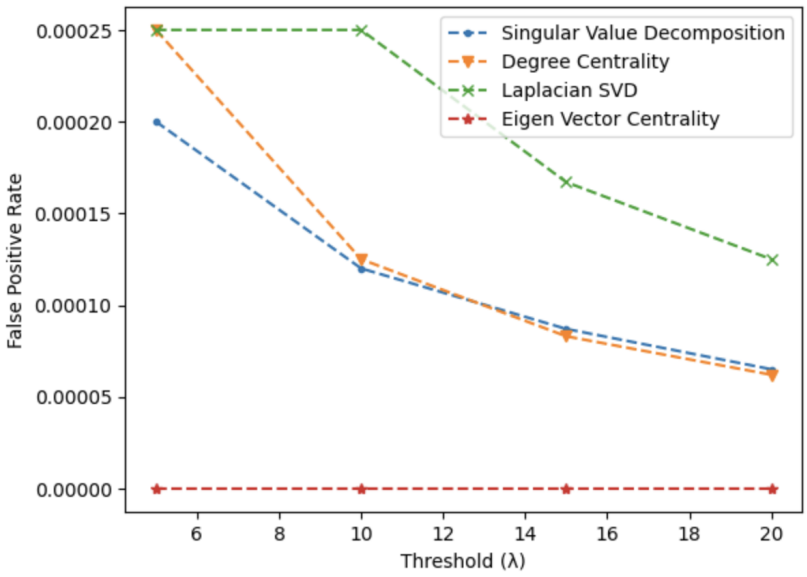


Figure 22

False Positive Rate vs λ for Using Different Graph Features for the Martingale Method on Barabási-Albert Type Graphs.

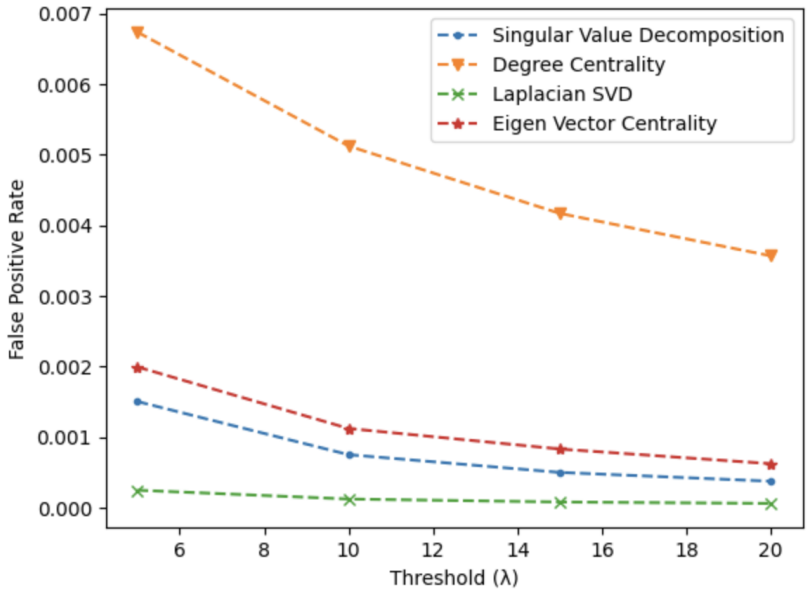
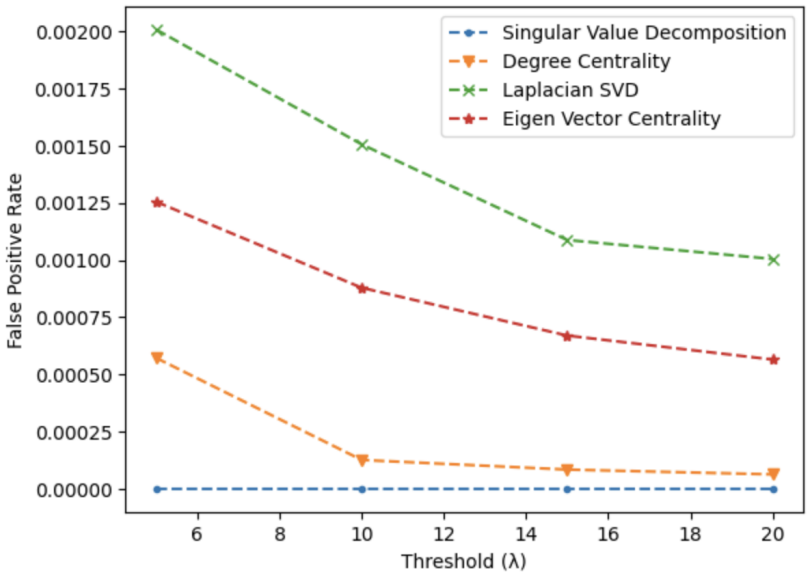


Figure 23

False Positive Rate vs λ for Using Different Graph Features for the Martingale Method on Internet AS Type Graphs.



Appendix B

Additional Results - Precision

Figure 24, Figure 25, and Figure 26 show the precision vs λ varying from 5 to 20 using (i) Eigenvector Centrality feature on Erdos-Renyi type graphs, (ii) Singular Value Decomposition feature on Barabasi-Albert type graphs, and (iii) degree centrality features on Internet AS type graphs, respectively, with varying task difficulty.

Figure 24

Precision vs λ Using Eigenvector Centrality Feature for the Martingale Method on Erdos-Rényi Type Graphs with Varying Task Difficulty.

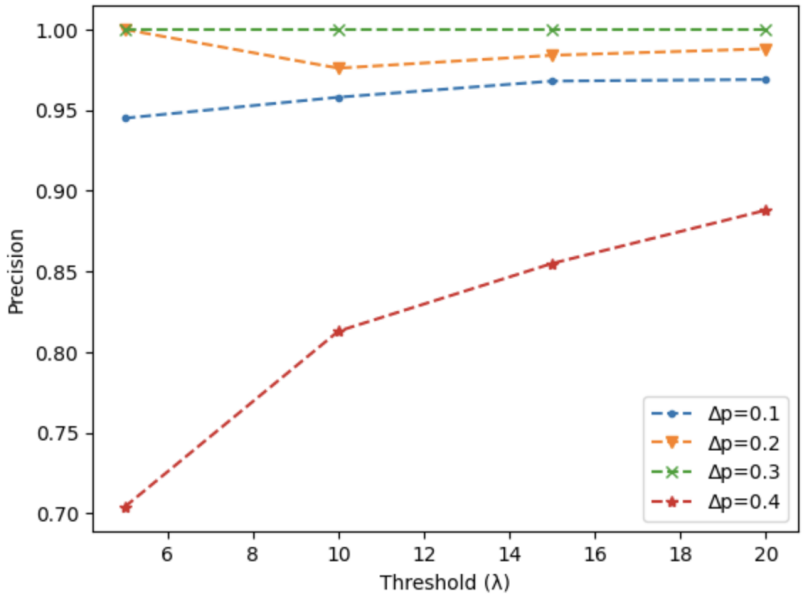


Figure 25

Precision vs λ Using Singular Value Decomposition Feature for the Martingale Method on Barabási-Albert Type Graphs with Varying Task Difficulty.

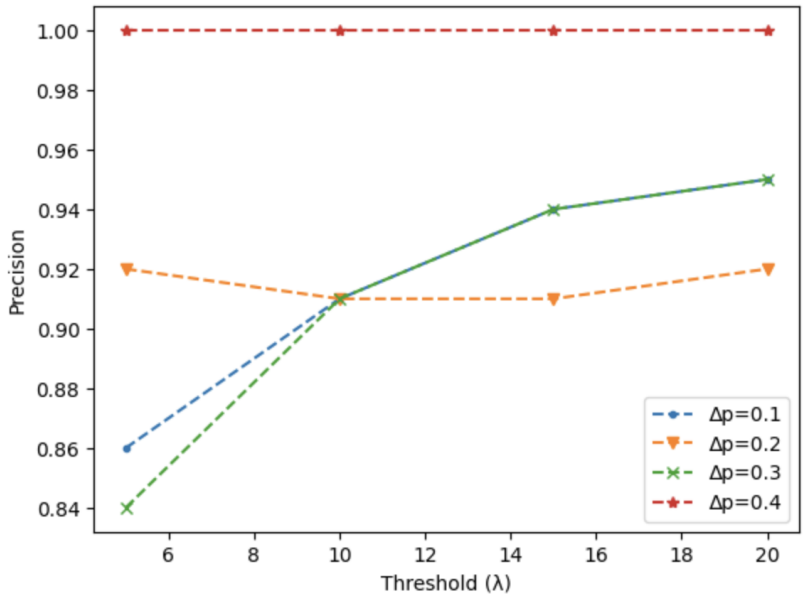
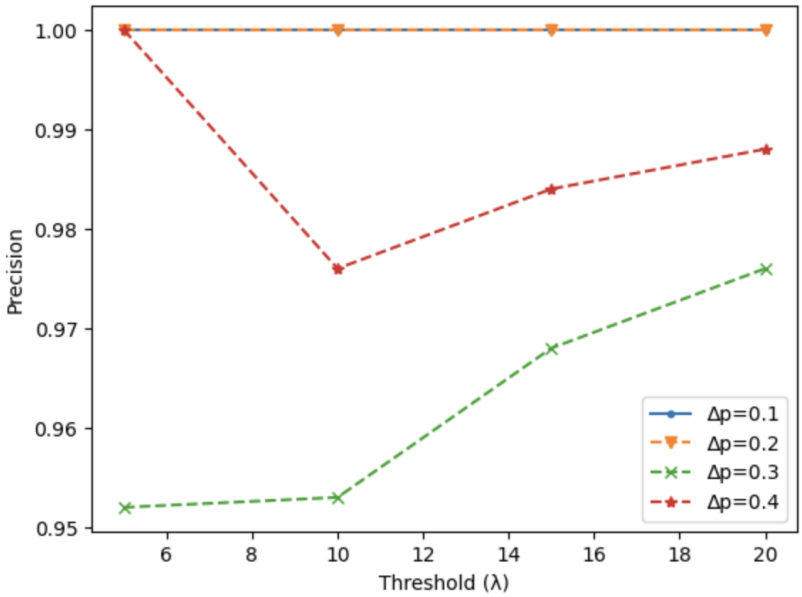


Figure 26

Precision vs λ Using Degree Centrality Feature for the Martingale Method on Internet AS Type Graphs with Varying Task Difficulty.



Appendix C

Additional Results - Recall

Figure 27 and Figure 28 show the recall vs λ varying from 5 to 20 using degree centrality feature on (i) Newman-Watt-Strogatz type graphs and (ii) Internet AS type graphs, respectively, with varying task difficulty for the martingale method.

Figure 27

Recall vs λ Using Degree Centrality Feature for the Martingale Method on Newman-Watts-Strogatz Type Graphs with Varying Task Difficulty.

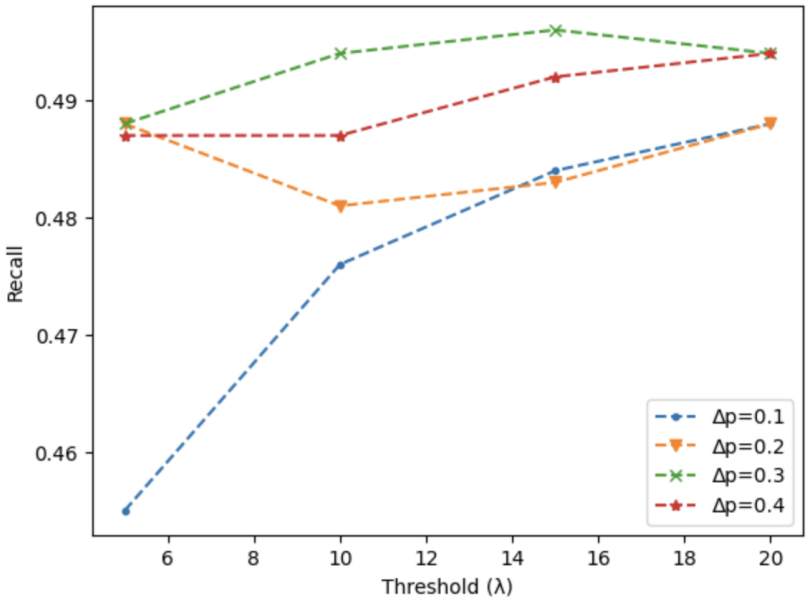
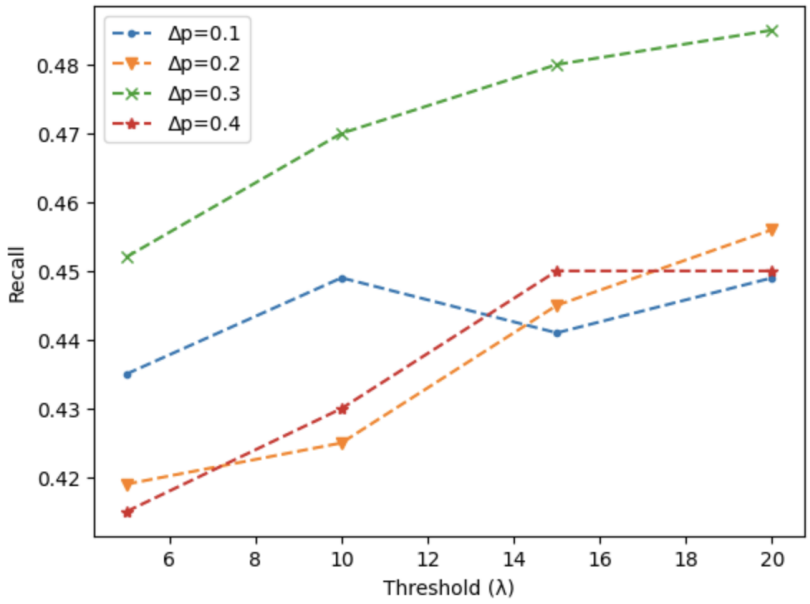


Figure 28

Recall vs λ Using Degree Centrality Feature for the Martingale Method on Internet AS Type Graphs with Varying Task Difficulty



Appendix D

Additional Results - F1-score

Figure 29, Figure 30, and Figure 31 show the F1 score vs λ varying from 5 to 20 using (i) Singular Value Decomposition feature on Erdos-Renyi type graphs, (ii) Eigenvector centrality feature on Internet AS type graphs, and (iii) Laplacian SVD feature on Newman-Watts-Strogatz type graphs, respectively, with varying task difficulty.

Figure 29

F1 Score vs λ Using Singular Value Decomposition Feature for the Martingale Method on Erdos-Rényi Type Graphs with Varying Task Difficulty

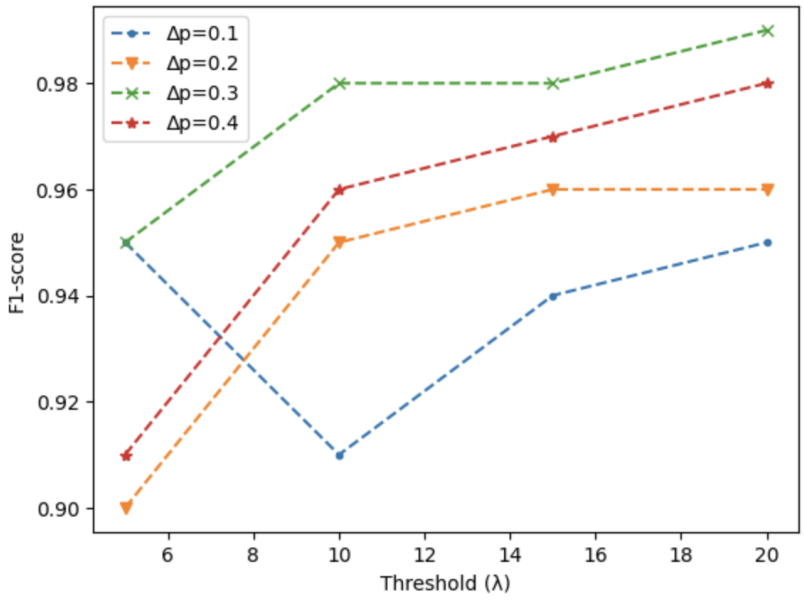


Figure 30

F1 Score vs λ Using Eigenvector Centrality Feature for the Martingale Method on Internet AS Type Graphs with Varying Task Difficulty.

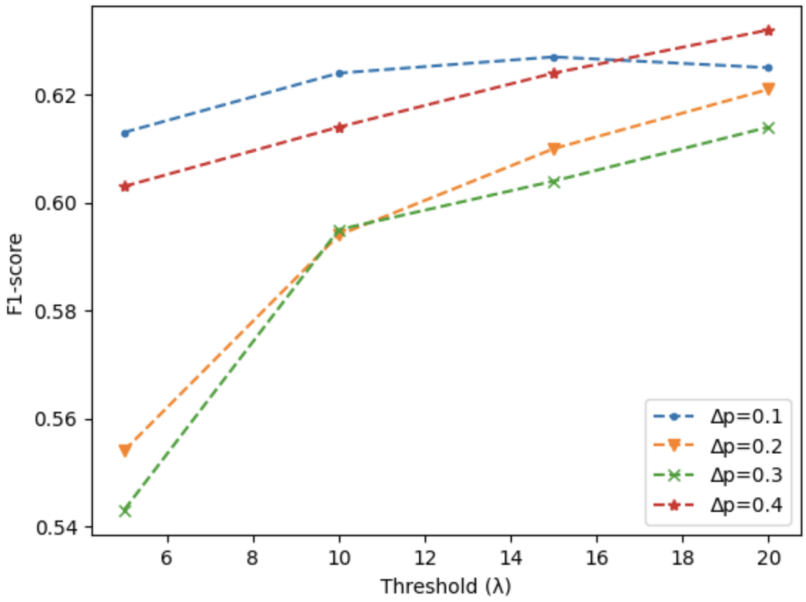


Figure 31

F1 Score vs λ Using Laplacian SVD Feature for the Martingale Method on Newman-Watts-Strogatz Type Graphs with Varying Task Difficulty.