University of Glasgow | School of Computing Science

Decentralized Change Detection Using Non-Parametric Methods for Smart Grid Anomaly Detection

Sotaro Suzuki

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at The University of Glasgow

24th August 2024

Abstract

The detection of changes in time series data, particularly in Locational Marginal Price (LMP) data, is a critical task for ensuring the stability and efficiency of energy markets. Traditional parametric change detection methods often require prior knowledge of data distributions, which may not be available or reliable in real-world scenarios. This research explores the efficacy of nonparametric change detection methods—CUSUM, GLR, Q-Q Distance, GEM, and PCA—applied to LMP data. The study further investigates two decision fusion strategies, Method A (Aggregation by Voting) and Method B (Statistical Aggregation), to enhance detection accuracy across multiple sensors (buses). Through a comprehensive experimental setup and evaluation, we demonstrate the strengths and limitations of each method, providing valuable insights into the application of nonparametric techniques in complex, high-dimensional time-series data environments.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: Sotaro Suzuki Signature: Sotaro Suzuki

Acknowledgements

I would like to express my gratitude to Dr. Christos Anagnostopoulos for his support and guidance throughout the project. Also, I would like to express my gratitude to my family for their support during this time.

Contents

Chapter 1: Introduction

Change detection in time series data is a fundamental problem with applications across various domains, including energy markets and Locational Marginal Price (LMP) analysis. As energy markets continue to evolve and face new challenges, the need for robust and efficient change detection methods has become increasingly critical. This research focuses on evaluating and comparing nonparametric change detection algorithms for LMP data, addressing the limitations of traditional parametric methods in complex, high-dimensional settings.

Traditional parametric methods such as Cumulative Sum (CUSUM) and Generalized Likelihood Ratio (GLR) tests have long been the cornerstone of change detection [1–3]. However, these methods often struggle in scenarios where data distributions are unknown or cannot be easily modeled, which is often the case in energy market data. To overcome these limitations, nonparametric approaches have gained significant attention in recent years.

This study investigates five nonparametric change detection algorithms: CUSUM, GLR, Quantile-Quantile (Q-Q) Distance, Geometric Entropy Minimization (GEM), and Principal Component Analysis (PCA). Each method offers unique advantages in handling complex, high-dimensional data without relying on specific distributional assumptions. For instance, the Q-Q distance method has shown effectiveness in decentralized settings [4], while GEM has demonstrated robustness in high-dimensional scenarios [5].

In the context of energy markets, change detection plays a crucial role in identifying market anomalies, sudden shifts in supply-demand dynamics, and potential manipulations. Recent work by Kurt et al. [6] has highlighted the potential of adaptive learning methods in smart grid monitoring, further emphasizing the need for advanced change detection techniques in this domain.

Our research aims to address the gap in comprehensive comparisons of multiple nonparametric methods in the specific context of LMP data and multi-bus power systems. We introduce novel decision fusion strategies to aggregate local detection decisions across multiple buses, contributing to the development of more robust and adaptive change detection frameworks for energy market monitoring.

Chapter 2: Related Work

Change detection in time series data has been a fundamental problem across various domains, including signal processing, statistical analysis, and machine learning. This section provides an overview of traditional and modern approaches to change detection, with a specific focus on their applications in energy markets and Locational Marginal Price (LMP) data analysis.

2.1 Traditional Parametric Methods

Parametric methods such as the Cumulative Sum (CUSUM) and Generalized Likelihood Ratio (GLR) tests have long been the cornerstone of change detection due to their effectiveness in scenarios where the underlying data distributions are known or can be reasonably estimated [1]. These methods operate under the assumption that the pre-change and post-change distributions are either fully known or can be parameterized, making them powerful in structured environments [2, 3].

CUSUM, introduced by Page [2], has been widely applied in quality control and, more recently, in cybersecurity and network anomaly detection. The method accumulates the log-likelihood ratio between two probability distributions, declaring a change when this sum exceeds a predefined threshold. GLR, on the other hand, extends this concept to scenarios where the post-change distribution parameters are unknown, estimating these parameters as part of the detection process [3].

Recent applications of CUSUM and GLR in smart grid cybersecurity have demonstrated their continued relevance in modern contexts. For instance, Kurt et al. [7] proposed a method for detecting hybrid and stealthy cyber-attacks in smart grids, showcasing the adaptability of these traditional methods to contemporary challenges.

However, these parametric methods often struggle in complex, high-dimensional settings where data distributions are unknown or cannot be easily modeled, which is often the case in energy market data.

2.2 Nonparametric Approaches

To overcome the limitations of parametric approaches, nonparametric methods have gained significant attention in recent years. These methods do not rely on specific distributional assumptions, making them more flexible and adaptable to diverse data characteristics.

2.2.1 Quantile-Quantile (Q-Q) Distance

The Quantile-Quantile (Q-Q) distance method compares the quantiles of two data distributions to detect changes without assuming any specific distributional form. This approach has been effectively applied in decentralized settings where data is gathered from multiple distributed sensors [4]. The Q-Q distance method is particularly advantageous in environments where maintaining low false alarm rates is critical and where the data exhibits significant variability.

Yang and Qi [4] demonstrated the effectiveness of Q-Q distance in decentralized quickest detection scenarios, showing its computational efficiency and robustness to outliers. Their approach, which combines Q-Q distance with data fusion techniques, has shown promise in handling complex, multi-sensor environments similar to those found in energy market monitoring systems.

2.2.2 Geometric Entropy Minimization (GEM)

The Geometric Entropy Minimization (GEM) approach focuses on the distribution of distances to the k-nearest neighbors in a transformed feature space. GEM detects changes by identifying outliers in this distribution, making it well-suited for high-dimensional data where traditional parametric methods may struggle [5].

GEM's strength lies in its ability to adapt to the underlying data structure without relying on predefined distributional assumptions. However, the computational complexity of GEM can be a concern for very large datasets. Recent work by Sricharan and Hero [8] has addressed this issue by proposing efficient implementations using bipartite k-NN graphs, making GEM more applicable to real-time scenarios in energy market monitoring.

2.2.3 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) has been widely used for change detection, particularly in scenarios involving high-dimensional data. PCA-based methods monitor variations in the principal components of the data, which can indicate shifts in the underlying data distribution.

Recent advancements in PCA-based change detection include the work of Xie et al. [9], who proposed a method for change-point detection in high-dimensional time series with missing data, a common scenario in energy market data. Their approach demonstrates the adaptability of PCA-based methods to challenging real-world scenarios in energy market monitoring.

2.3 Change Detection in Energy Markets

In the context of energy markets, change detection plays a crucial role in identifying market anomalies, sudden shifts in supply-demand dynamics, and potential manipulations. The application of nonparametric change detection methods to Locational Marginal Price (LMP) data has shown promise in early detection of price spikes and unusual market behaviors.

Energy market data presents unique challenges for change detection algorithms. The complex, often non-Gaussian nature of price data, coupled with the need for rapid, reliable detection of significant changes, makes this domain particularly challenging. Furthermore, the high-dimensional nature of multi-bus systems and the potential for localized anomalies necessitate approaches that can handle spatial and temporal correlations effectively.

Recent work by Kurt et al. [6] proposed a reinforcement learning approach for online cyber-attack detection in smart grids. Their method demonstrated improved performance in detecting both abrupt and gradual changes in smart grid data compared to traditional techniques, highlighting the potential of adaptive learning methods in energy market monitoring.

2.4 Conclusion and Research Gap

While significant progress has been made in developing nonparametric change detection methods and applying them to energy market data, several challenges remain. The high-

dimensional, non-stationary nature of LMP data, coupled with the need for real-time detection and low false alarm rates, presents a unique set of requirements that are not fully addressed by existing methods.

Furthermore, while these nonparametric methods offer robust alternatives to traditional parametric approaches, they are often evaluated independently of each other. There is a lack of comprehensive comparisons of multiple nonparametric methods in the specific context of LMP data and multi-bus power systems.

Our research aims to address this gap by providing a comprehensive comparison of CUSUM, GLR, Q-Q distance, GEM, and PCA methods in the context of LMP data. Additionally, we introduce novel decision fusion strategies to aggregate local detection decisions across multiple buses, contributing to the development of more robust and adaptive change detection frameworks for energy market monitoring.

Chapter 3: Methodology

This chapter outlines the methodologies employed in our research for detecting changes in Locational Marginal Price (LMP) data using various nonparametric change detection algorithms. We implemented and compared five algorithms: CUSUM (Cumulative Sum), GLR (Generalized Likelihood Ratio), Q-Q Distance, GEM (Geometric Entropy Minimization), and PCA (Principal Component Analysis). Additionally, two decision fusion strategies, referred to as Method A and Method B, were applied to aggregate the results from multiple sensors (buses).

3.1 CUSUM Algorithm

The Cumulative Sum (CUSUM) algorithm is a sequential analysis technique used for detecting changes in a process. In our nonparametric implementation, we use the following decision statistic:

$$g_k = \max(0, g_{k-1} + s_k) \tag{3.1}$$

where s_k is a score function based on the observed data at time k. The stopping rule for declaring a change is:

$$t_a = \inf k : g_k \geq h \tag{3.2}$$

where h is a predefined threshold. The pseudo-code for the nonparametric CUSUM algorithm is as follows: In our implementation, we apply the CUSUM algorithm to each bus in

Algorithm 1 Nonparametric CUSUM

Input: Data $\mathbf{y} = (y_1, \ldots, y_n)$, mean μ, threshold h
Initialize: $g_0 \leftarrow 0$, $\mathbf{g} \leftarrow \mathbf{0}^n$
for $k \leftarrow 1$ to n **do**
    $s_k \leftarrow y_k - \mu$
    $g_k \leftarrow \max(0, g_{k-1} + s_k)$
    $\mathbf{g}[k] \leftarrow g_k$
    if $g_k \geq h$ **then**
        return k, $\mathbf{g}$
    end if
end for
return -1, $\mathbf{g}$

the power system independently. The mean μ for each bus is calculated from its historical data.

3.2 GLR Algorithm

The Generalized Likelihood Ratio (GLR) algorithm is an extension of the CUSUM algorithm that can handle unknown post-change parameters. In our nonparametric implementation, we use a rank-based approach. The decision statistic for the GLR algorithm is:

$$g_k = \max_{1 \leq j \leq k} \sum_{i=j}^{k} (R_i^{(j-)} - R_i^{(j+)})^2 \tag{3.3}$$

where $R_i^{(j-)}$ and $R_i^{(j+)}$ are the ranks of the i-th observation in the windows before and after the potential change point j, respectively. The pseudo-code for the nonparametric GLR algorithm is as follows: In our implementation, we apply the GLR algorithm to each bus in

Algorithm 2 Nonparametric GLR

Input: Data $\mathbf{y} = (y_1, \ldots, y_n)$, window size w, threshold h
Initialize: $\mathbf{g} \leftarrow \mathbf{0}^n$, $t_{hat} \leftarrow 0$
Precompute ranks $\mathbf{R}$ for all windows of size w
for $k \leftarrow 2w$ to n **do**
    $max_statistic \leftarrow -\infty$
    $max_j \leftarrow 0$
    for $j \leftarrow k - w + 1$ to k **do**
        $before_rank \leftarrow \mathbf{R}[j - w]$
        $after_rank \leftarrow \mathbf{R}[j]$
        $statistic \leftarrow \sum_{i=1}^{w} (before_rank[i] - after_rank[i])^2$
        if $statistic > max_statistic$ **then**
            $max_statistic \leftarrow statistic$
            $max_j \leftarrow j$
        end if
    end for
    $\mathbf{g}[k-1] \leftarrow max_statistic$
    if $\mathbf{g}[k-1] > h$ **then**
        $t_{hat} \leftarrow max_j$
        break
    end if
end for
return t_{hat}, $\mathbf{g}$

the power system independently. The window size w and threshold h are tuning parameters that can be adjusted based on the specific characteristics of the data and the desired sensitivity of the detection.

3.3 QQ-Distance Algorithm

The QQ-distance algorithm is a nonparametric approach for detecting changes in data distributions. It compares the quantiles of two data windows to measure the dissimilarity between their underlying distributions.

3.3.1 QQ-Distance Calculation

For two data windows W_1 and W_2 of equal size s, the QQ-distance is defined as:

$$d_{QQ}(W_1, W_2) = \frac{1}{s} \sum_{i=1}^{s} \frac{\sqrt{2}}{2} \left| Q_{W_1}\left(\frac{i}{s}\right) - Q_{W_2}\left(\frac{i}{s}\right) \right| \tag{3.4}$$

where Q_{W_1} and Q_{W_2} are the quantile functions of W_1 and W_2, respectively.

3.3.2 Change Detection Procedure

The change detection procedure uses two sliding windows:

- A reference window W_1 representing the pre-change distribution

- A detection window W_2 containing the most recent observations

The algorithm computes the QQ-distance between these windows at each time step. A change is declared when the QQ-distance exceeds a predefined threshold.

Algorithm 3 QQ-Distance Change Detection

Input: Data stream $\mathbf{y} = (y_1, \ldots, y_n)$, window size w, threshold h
Initialize: $\mathbf{g} \leftarrow \mathbf{0}^n$
for $k \leftarrow 2w$ to n **do**
    $W_1 \leftarrow (y_{k-2w+1}, \ldots, y_{k-w})$
    $W_2 \leftarrow (y_{k-w+1}, \ldots, y_k)$
    $d_{QQ} \leftarrow$ calculate_qq_distance(W_1, W_2)
    $\mathbf{g}[k - 2w + 1] \leftarrow d_{QQ}$
    if $d_{QQ} > h$ **then**
        return $k - w + 1$, $\mathbf{g}$
    end if
end for
return -1, $\mathbf{g}$

Algorithm 4 Calculate QQ-Distance

function CALCULATE_QQ_DISTANCE(W_1, W_2)
    $s \leftarrow \min(|W_1|, |W_2|)$
    $q \leftarrow (1, 2, \ldots, s)/s$
    $Q_1 \leftarrow$ quantile(W_1, q)
    $Q_2 \leftarrow$ quantile(W_2, q)
    $d_{QQ} \leftarrow \frac{1}{s} \sum_{i=1}^{s} \frac{\sqrt{2}}{2} |Q_1[i] - Q_2[i]|$
    return d_{QQ}
end function

In our implementation, we apply the QQ-distance algorithm to each bus in the power system independently. The window size w and threshold h are tuning parameters that can be adjusted based on the specific characteristics of the data and the desired sensitivity of the detection.

3.3.3 Geometric Entropy Minimization (GEM) Algorithm

The GEM algorithm is a nonparametric approach for anomaly detection based on nearest neighbor statistics. The algorithm consists of offline and online phases.

Offline Phase

Given a set of nominal data points $\mathbf{X} = \mathbf{x}_i : i = 1, 2, \ldots, N$, the offline phase proceeds as follows:

Algorithm 5 GEM Offline Phase

1: Partition $\mathbf{X}$ into two subsets $\mathbf{S}_1$ and $\mathbf{S}_2$ with sizes N_1 and N_2
2: Estimate intrinsic dimension d of the data
3: **for** each $\mathbf{x}_j \in \mathbf{S}_2$ **do**
4: Find k nearest neighbors of $\mathbf{x}_j$ in $\mathbf{S}1$
5: Compute $d_j = \sum i = 1^k e_j(i)$, where $e_j(i)$ is the distance to the i-th nearest neighbor
6: **end for**
7: Sort $d_j : \mathbf{x}_j \in \mathbf{S}_2$ in ascending order

Online Phase

For each new data point $\mathbf{x}_t$, the online phase computes:

$$d_t = \sum_{i=1}^{k} e_t(i) \tag{3.5}$$

where $e_t(i)$ is the distance to the i-th nearest neighbor in $\mathbf{S}_1$. The tail probability is estimated as:

$$\hat{p}t = \frac{1}{N_2} \sum \mathbf{x}_j \in \mathbf{S}_2 \mathbf{1} d_j > d_t \tag{3.6}$$

The statistical evidence is then computed:

$$\hat{s}_t = \log\left(\frac{\alpha}{\hat{p}_t}\right) \tag{3.7}$$

Finally, the decision statistic is updated:

$$g_t = \max 0, g_{t-1} + \hat{s}_t \tag{3.8}$$

An anomaly is declared if $g_t \geq h$, where h is a predefined threshold.

3.3.4 Principal Component Analysis (PCA) Algorithm

The PCA algorithm is used when the observed data exhibits low intrinsic dimensionality. It consists of offline and online phases.

Offline Phase

Given a set of nominal data points $\mathbf{S}_1 = \mathbf{x}_i : i = 1, 2, \ldots, N_1$, the offline phase proceeds as follows:

Algorithm 6 PCA Offline Phase

Compute the sample mean $\bar{\mathbf{x}} = \frac{1}{N_1} \sum_{i=1}^{N_1} \mathbf{x}i$
Compute the sample covariance matrix $\mathbf{Q} = \frac{1}{N_1} \sum i = 1^{N_1} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}i - \bar{\mathbf{x}})^T$
Compute eigenvalues $\lambda_j : j = 1, 2, \ldots, p$ and eigenvectors $\mathbf{v}j : j = 1, 2, \ldots, p$ of $\mathbf{Q}$
Choose r based on desired fraction γ of retained variance: $\frac{\sum j=1^r \lambda_j}{\sum j=1^p \lambda_j} \geq \gamma$
Form matrix $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r]$
for each $\mathbf{x}_j \in \mathbf{S}_2$ **do**
    Compute residual $\mathbf{r}_j = (\mathbf{I}_p - \mathbf{V}\mathbf{V}^T)(\mathbf{x}_j - \bar{\mathbf{x}})$
    Compute $|\mathbf{r}_j|_2$
end for
Sort $|\mathbf{r}_j|_2 : \mathbf{x}_j \in \mathbf{S}_2$ in ascending order

Online Phase

For each new data point $\mathbf{x}_t$, the online phase computes:

$$\mathbf{r}_t = (\mathbf{I}_p - \mathbf{V}\mathbf{V}^T)(\mathbf{x}_t - \bar{\mathbf{x}}) \tag{3.9}$$

The tail probability is estimated as:

$$\hat{p}t = \frac{1}{N_2} \sum \mathbf{x}_j \in \mathbf{S}_2 \mathbf{1} |\mathbf{r}_j|_2 > |\mathbf{r}_t|_2 \tag{3.10}$$

The statistical evidence and decision statistic are computed and updated as in the GEM algorithm. Both the GEM and PCA algorithms use the same CUSUM-like detection mechanism in their online phases, allowing for efficient and adaptive anomaly detection in high-dimensional data streams.

3.4 Fusion Strategies

In this research, we developed and employed two fusion strategies, Method A and Method B, to effectively aggregate the change detection results across multiple buses. These strategies are designed to enhance the robustness and accuracy of the change detection process by leveraging the strengths of different decision-making and statistical aggregation techniques.

3.4.1 Method A: Voting-based Fusion

Method A assumes a star-like topology where all buses send binary decisions (0 for no change, 1 for change) to a central 'sink' node. The algorithm is as follows:

where N is the total number of buses and $p \in \{0.1, 0.2, 0.5, 0.7, 0.9\}$ is the percentage threshold for Rule 3.

3.4.2 Method B: Statistic-based Fusion

Method B involves buses sending their local statistics $s_k(t)$ to the sink node, which then applies various aggregation functions:

The global threshold H can be set as the average, minimum, maximum, or median of the local thresholds $\{h_1, \ldots, h_N\}$. Both methods aim to improve detection accuracy by combining information from multiple buses, with Method A using a simple voting scheme and Method B leveraging more detailed statistical information.

Algorithm 7 Method A: Voting-based Fusion

Each bus k determines its optimal local threshold h_k
for each time step t **do**
  for each bus k **do**
    if local change detected **then**
      Send 1 to sink
    else
      Send 0 to sink
    end if
  end for
  Sink records votes $v_t = \sum_{k=1}^{N} \text{vote}_k(t)$
  Apply one of the following decision rules:
  if Rule 1: At least one bus **then**
    if $v_t \geq 1$ **then**
      Declare change, set detection time $T_d = t$
    end if
  else if Rule 2: All buses **then**
    if $v_t = N$ **then**
      Declare change, set detection time $T_d = t$
    end if
  else if Rule 3: At least $p\%$ of buses **then**
    if $v_t \geq p \cdot N$ **then**
      Declare change, set detection time $T_d = t$
    end if
  end if
end for

Algorithm 8 Method B: Statistic-based Fusion

Each bus k determines its optimal local threshold h_k
Sink computes global threshold H based on $\{h_1, \ldots, h_N\}$
for each time step t **do**
  for each bus k **do**
    Send statistic $s_k(t)$ to sink
  end for
  Sink aggregates statistics using one of:
  if Aggregation 1: Average **then**
    $S_t = \frac{1}{N} \sum_{k=1}^{N} s_k(t)$
  else if Aggregation 2: Median **then**
    $S_t = \text{median}\{s_1(t), \ldots, s_N(t)\}$
  else if Aggregation 3: Outlier-robust Average **then**
    $\text{med} = \text{median}\{s_1(t), \ldots, s_N(t)\}$
    $\text{MAD} = \text{median}\{|s_k(t) - \text{med}|\}$
    $z_k = 0.6745(s_k(t) - \text{med})/\text{MAD}$
    $S_t = \text{mean}\{s_k(t) : |z_k| \leq 3.5\}$
  end if
  if $S_t > H$ **then**
    Declare change, set detection time $T_d = t$
  end if
end for

Chapter 4: Experimental Setup

This chapter details the experimental setup used to evaluate the performance of nonparametric change detection methods on Locational Marginal Price (LMP) data.

4.1 Dataset and Preprocessing

The dataset used in this study consists of Locational Marginal Price (LMP) data from the file `'LMP.csv'`, containing 8785 rows and 142 columns. Each row represents a weekly observation, with columns for Week, Label (a binary indicator of significant change), and LMP values for 140 buses. Importantly, the dataset is complete, with no missing values in any of the rows or columns.

The preprocessing steps included:

- Selecting eight specific buses (Bus115, Bus116, Bus117, Bus118, Bus119, Bus121, Bus135, and Bus139) for analysis.
- Using the last 855 samples of the dataset to balance computational efficiency and statistical significance.

4.2 Algorithm Implementation and Evaluation

We implemented and evaluated five nonparametric change detection algorithms: CUSUM, GLR, QQ Distance, GEM, and PCA. The evaluation process included:

- Performance analysis across all buses with fixed parameters.
- Sensitivity analysis of key parameters for specific buses.
- Visualization of detection performance across buses and parameter settings.

Table 4.1 summarizes the configurations for each algorithm:

Table 4.1: Algorithm Configurations

Algorithm	**Initial Analysis**	**Bus-specific Analysis**
CUSUM	Threshold = 2.0 for all buses	Bus 116, varying Threshold
GLR	Threshold = 2000, Window size = 24 for all buses	Bus 135, fixed Window size = 24, varying Threshold Bus 135, fixed Threshold = 2000, varying Window size
QQ Distance	Threshold = 0.1, Window size = 24 for all buses	Bus 119, fixed Window size = 24, varying Threshold Bus 119, fixed Threshold = 0.1, varying Window size
GEM	Alpha = 0.2, Threshold = 4, K = 16 for all buses	Bus 115, fixed Alpha = 0.2, K = 16, varying Threshold Bus 115, fixed Threshold = 4, K = 16, varying Alpha Bus 115, fixed Threshold = 4, Alpha = 0.2, varying K
PCA	Gamma = 0.9, Threshold = 4, Alpha = 0.2 for all buses	Bus 115, fixed Gamma = 0.9, Alpha = 0.2, varying Threshold Bus 115, fixed Threshold = 4, Alpha = 0.2, varying Gamma Bus 115, fixed Threshold = 4, Gamma = 0.9, varying Alpha

4.3 Fusion Strategies Evaluation

To evaluate the effectiveness of multi-bus change detection, we implemented and analyzed two fusion strategies: Method A (Voting-based Fusion) and Method B (Statistic-based Fusion). These strategies were applied to the results of each individual change detection algorithm (CUSUM, GLR, QQ Distance, GEM, and PCA) with fixed parameters as follows:

Algorithm	**Fixed Parameters**
CUSUM	Threshold = 2.0
GLR	Threshold = 2000, Window size = 24
QQ Distance	Threshold = 0.1, Window size = 24
GEM	Alpha = 0.2, Threshold = 4, K = 16
PCA	Gamma = 0.9, Threshold = 4, Alpha = 0.2

Table 4.2: Fixed parameters for each algorithm in fusion strategy evaluation

We then analyzed how the performance metrics change when varying the parameters within Method A and Method B. These fusion strategies aim to combine information from multiple buses to improve overall detection accuracy and robustness.

4.3.1 Method A: Aggregation by Voting

Method A is a voting-based fusion strategy that aggregates binary decisions from individual buses. This approach is intuitive and computationally efficient, making it suitable for real-time applications. In our experiments, Method A was implemented with the following voting schemes:

- **At Least One Bus**: A change was declared if any bus signaled a change, maximizing sensitivity to detect even minor disturbances. This scheme is highly sensitive but may lead to increased false alarms.
- **All Buses**: A change was declared only if all buses signaled a change, minimizing false positives but potentially missing small or localized changes. This scheme is the most conservative and may result in delayed detections.
- **Percentage-Based Voting**: The voting scheme was tested with percentages p set to $10\%, 20\%, 50\%, 70\%$, and 90%, to analyze how different levels of consensus affected detection performance. The detection time was recorded as the time when the required number of buses reached consensus. This scheme provides a flexible middle ground between the two extremes.

4.3.2 Method B: Statistical Aggregation

Method B is a more sophisticated approach that aggregates statistical information from individual buses rather than binary decisions. This method potentially captures more nuanced information about the detected changes. For Method B, the following statistical aggregation techniques were tested:

- **Average**: The average of the statistical values across all buses was computed and compared against a global threshold H. This method is simple but can be sensitive to outliers.

- **Median**: The median of the statistical values was used to determine the change detection decision, providing robustness against extreme values. This method is less sensitive to outliers compared to the average.
- **Outlier-Detection Aggregation**: Outliers were identified and removed using the Median Absolute Deviation (MAD) method with a threshold of 3.5. The average of the remaining non-outlier values was then used for change detection. This method aims to combine the benefits of both average and median approaches while mitigating the impact of potential outliers.

The global threshold H was determined using various strategies to explore different levels of sensitivity:

- **Average of Local Thresholds**: H was calculated as the average of the individual thresholds $h[k]$ for each bus.
- **Minimum of Local Thresholds**: H was set as the minimum of the local thresholds to capture early changes, potentially increasing sensitivity.
- **Maximum of Local Thresholds**: H was set as the maximum of the local thresholds, requiring a higher consensus among buses and potentially reducing false alarms.
- **Median of Local Thresholds**: H was set as the median of the local thresholds, providing a balance between early detection and robustness.

These configurations were systematically varied to identify the optimal settings for each fusion strategy in the context of LMP data change detection. By comparing the performance of Method A and Method B under various settings, we aimed to determine which fusion strategy is most effective for detecting changes in multi-bus LMP data.

4.4 Evaluation Metrics

To assess the performance of each algorithm and fusion strategy, we used the following metrics:

- **Accuracy**: The ratio of correct detections (both true positives and true negatives) to the total number of cases.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

  where TP is True Positives, TN is True Negatives, FP is False Positives, and FN is False Negatives.

- **Precision**: The ratio of true positives to the total number of positive detections.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

- **Recall**: The ratio of true positives to the total number of actual positives.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

- **F1 Score**: The harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

- **False Alarm Rate (FAR)**: The proportion of false positives among all negative instances.

$$\text{FAR} = \frac{FP}{FP + TN} \tag{4.5}$$

- **Expected Delay (ED)**: The average time between the occurrence of a change and its detection.

$$\text{ED} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \max(0, \hat{t}_c - t_c) \tag{4.6}$$

  where $\mathcal{C}$ is the set of true change points, t_c is the time of the true change point, and $\hat{t}_c$ is the time of the closest detected change point after t_c.

- **Detection Time**: The time elapsed from the start of the time series to the detection of a change.

$$\text{Detection Time} = \min\{t : \hat{y}_t = 1\} \tag{4.7}$$

  where $\hat{y}_t$ is the predicted label at time t, with 1 indicating a detected change.

These metrics provide a comprehensive evaluation of the change detection performance, balancing between the ability to accurately identify changes (precision, recall, F1 score), the overall correctness of the algorithm (accuracy), the tendency to falsely identify changes (FAR), and the timeliness of change detection (ED and Detection Time).

Chapter 5: Results

This chapter presents the results of our experimental evaluation of nonparametric change detection algorithms on Locational Marginal Price (LMP) data. We begin by analyzing the performance of the Cumulative Sum (CUSUM) algorithm, followed by detailed examinations of the Generalized Likelihood Ratio (GLR), QQ Distance, Geometric Entropy Minimization (GEM), and Principal Component Analysis (PCA) methods.

5.1 CUSUM Algorithm Performance

The CUSUM algorithm's performance was evaluated across different buses and threshold values. Table 5.1 presents the results for all buses with a fixed threshold of 2.0, which was chosen as a representative mid-range value.

Bus	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus115	2.0	0.200	0.197	1.000	0.329	0.996	-1.0	1
Bus116	2.0	0.725	0.411	0.923	0.569	0.323	-1.0	394
Bus117	2.0	0.202	0.198	1.000	0.330	0.993	-1.0	2
Bus118	2.0	0.202	0.198	1.000	0.330	0.993	-1.0	3
Bus119	2.0	0.202	0.198	1.000	0.330	0.993	-1.0	2
Bus121	2.0	0.804	0.000	0.000	0.000	0.000	0.0	-1
Bus135	2.0	0.502	0.283	1.000	0.441	0.620	-1.0	138
Bus139	2.0	0.367	0.200	0.738	0.314	0.723	-1.0	16

Table 5.1: CUSUM performance metrics for all buses with a fixed threshold of 2.0

The results show significant variations in performance across different buses. Bus116 and Bus135 demonstrate notably higher accuracy and F1 scores compared to other buses, suggesting that these buses may have more distinct change patterns that are easier for the CUSUM algorithm to detect. Conversely, Bus121 shows no detection capability, indicating that the chosen threshold might be too high for this particular bus or that it experiences minimal changes. Figure 5.1 provides a visual representation of the CUSUM algorithm's performance across all buses with fixed parameters.

To further investigate the CUSUM algorithm's sensitivity to parameter changes, we conducted a detailed analysis on Bus116, varying the threshold while keeping other parameters constant. Table 5.2 presents these results.

Bus	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus116	0.5	0.537	0.295	0.976	0.453	0.571	-1.0	8
Bus116	1.0	0.671	0.370	0.958	0.534	0.399	-1.0	11
Bus116	2.0	0.725	0.411	0.923	0.569	0.323	-1.0	394
Bus116	4.0	0.723	0.405	0.881	0.555	0.316	9.5	490
Bus116	8.0	0.694	0.362	0.732	0.484	0.316	22.0	515

Table 5.2: CUSUM performance metrics for Bus116 with varying thresholds

The results for Bus116 demonstrate the trade-off between different performance metrics as the threshold varies. Lower thresholds (0.5 and 1.0) result in higher recall but also higher false alarm rates. As the threshold increases, we observe improved accuracy and precision, but at the cost of reduced recall and increased detection time. The optimal threshold appears to be around 2.0, where we achieve a balance between accuracy, precision, and recall.

Figure 5.2 illustrates the impact of varying the threshold on CUSUM performance for Bus116.

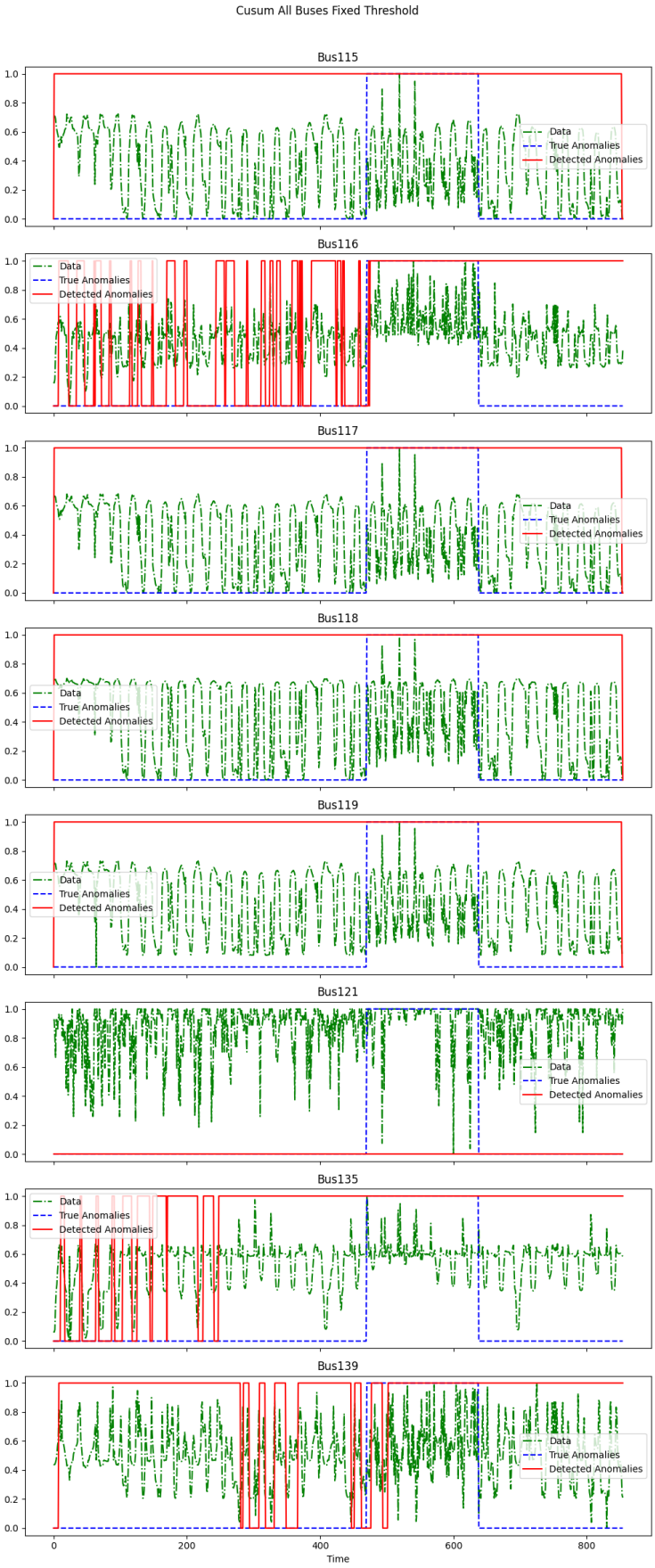

Figure 5.1: CUSUM performance metrics for all buses with fixed threshold

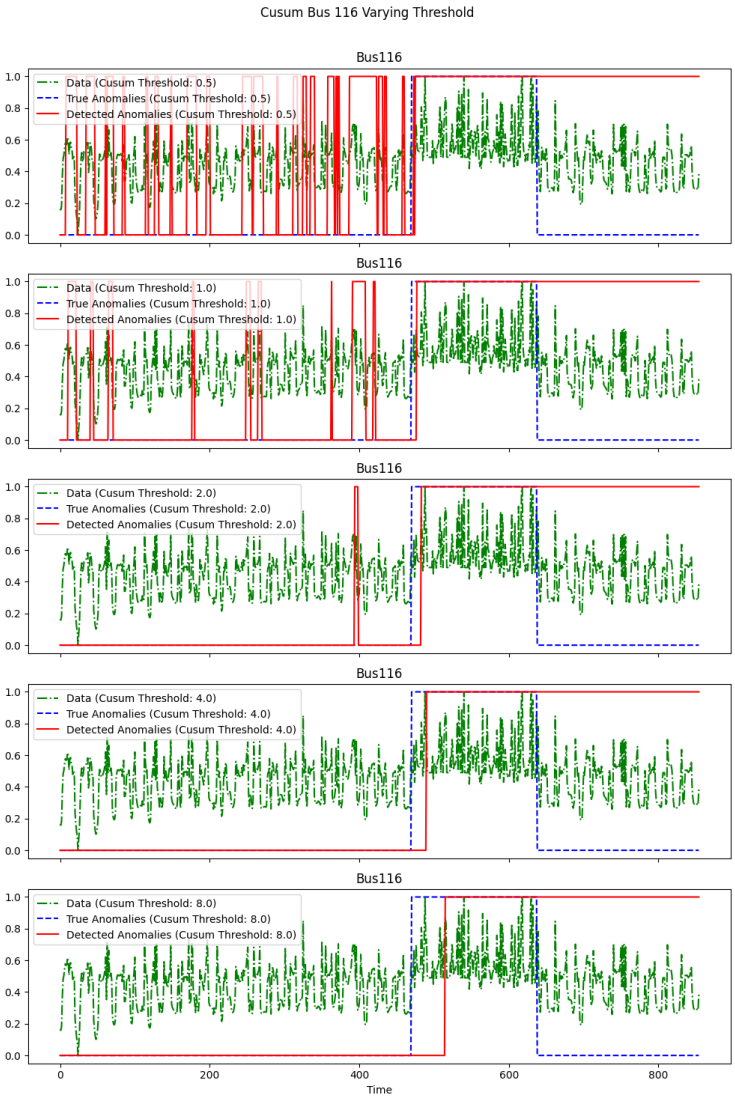

Figure 5.2: CUSUM performance metrics for Bus116 with varying thresholds

5.2 Generalized Likelihood Ratio (GLR) Algorithm Performance

The Generalized Likelihood Ratio (GLR) algorithm was evaluated across different buses with fixed parameters and then analyzed for sensitivity to parameter changes on a specific bus.

5.2.1 Performance Across Buses

Table 5.3 presents the GLR algorithm's performance across all buses with a fixed window size of 24 and threshold of 2000.

Bus	Window Size	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus115	24	2000	0.804	0.000	0.000	0.000	0.000	0.0	-1
Bus116	24	2000	0.303	0.220	1.000	0.361	0.868	-1.0	91
Bus117	24	2000	0.804	0.000	0.000	0.000	0.000	0.0	-1
Bus118	24	2000	0.804	0.000	0.000	0.000	0.000	0.0	-1
Bus119	24	2000	0.804	0.000	0.000	0.000	0.000	0.0	-1
Bus121	24	2000	0.240	0.205	1.000	0.341	0.946	-1.0	37
Bus135	24	2000	0.736	0.423	0.946	0.585	0.316	4.0	479
Bus139	24	2000	0.299	0.219	1.000	0.359	0.872	-1.0	88

Table 5.3: GLR performance metrics for all buses with fixed window size (24) and threshold (2000)

The results show significant variations in GLR performance across different buses. Bus135 demonstrates the best overall performance with high accuracy (0.736) and F1 score (0.585), suggesting that the chosen parameters are well-suited for detecting changes on this bus. In contrast, several buses (Bus115, Bus117, Bus118, and Bus119) show no detection capability, indicating that the threshold might be too high for these buses or that they experience minimal changes. Figure A.1 in Appendix A.2 provides a visual representation of the GLR algorithm's performance across all buses with fixed parameters.

5.2.2 Parameter Sensitivity Analysis

To investigate the GLR algorithm's sensitivity to parameter changes, we conducted detailed analyses on Bus135, varying the threshold and window size independently.

Threshold Sensitivity

Table 5.4 presents the results for Bus135 with varying thresholds and a fixed window size of 24.

Bus	Window Size	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus135	24	1000	0.496	0.280	1.000	0.438	0.627	-1.0	256
Bus135	24	1500	0.508	0.285	1.000	0.444	0.613	-1.0	266
Bus135	24	2000	0.736	0.423	0.946	0.585	0.316	4.0	479
Bus135	24	2500	0.735	0.421	0.940	0.582	0.316	4.5	480
Bus135	24	3000	0.635	0.252	0.435	0.319	0.316	47.0	565

Table 5.4: GLR performance metrics for Bus135 with varying thresholds and fixed window size (24)

The results demonstrate the impact of threshold variation on GLR performance. Lower thresholds (1000 and 1500) result in higher recall but also higher false alarm rates. As the threshold increases, we observe improved accuracy and precision, but at the cost of reduced recall and increased detection time. The optimal threshold appears to be around 2000, where we achieve a balance between accuracy, precision, and recall. Figure A.2 in Appendix A.2 illustrates the impact of varying the threshold on GLR performance for Bus135.

Window Size Sensitivity

Table 5.5 presents the results for Bus135 with varying window sizes and a fixed threshold of 2000.

Bus	Window Size	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus135	6	2000	0.804	0.000	0.000	0.000	0.000	0.0	-1
Bus135	12	2000	0.804	0.000	0.000	0.000	0.000	0.0	-1
Bus135	24	2000	0.736	0.423	0.946	0.585	0.316	4.0	479
Bus135	48	2000	0.305	0.220	1.000	0.361	0.865	-1.0	93
Bus135	96	2000	0.315	0.223	1.000	0.364	0.853	-1.0	101

Table 5.5: GLR performance metrics for Bus135 with varying window sizes and fixed threshold (2000)

The window size significantly affects GLR performance. Smaller window sizes (6 and 12) result in no detection, possibly due to insufficient data for reliable change detection. As the window size increases, we observe improved detection capability, with the best performance at a window size of 24. However, further increases in window size lead to higher false alarm rates and reduced accuracy. Figure A.3 in Appendix A.2 illustrates the impact of varying the window size on GLR performance for Bus135.

5.3 QQ Distance Algorithm Performance

The QQ Distance algorithm was evaluated across different buses with fixed parameters and then analyzed for sensitivity to parameter changes on a specific bus.

5.3.1 Performance Across Buses

Table 5.6 presents the QQ Distance algorithm's performance across all buses with a fixed window size of 24 and threshold of 0.1.

Bus	Window Size	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus115	24	0.1	0.230	0.213	1.000	0.351	0.972	-1.0	0
Bus116	24	0.1	0.782	0.420	0.125	0.193	0.045	8.5	441
Bus117	24	0.1	0.265	0.204	0.875	0.331	0.895	-1.0	11
Bus118	24	0.1	0.347	0.173	0.565	0.265	0.711	-1.0	54
Bus119	24	0.1	0.236	0.200	0.893	0.327	0.936	-1.0	10
Bus121	24	0.1	0.792	0.000	0.000	0.000	0.000	0.0	-1
Bus135	24	0.1	0.688	0.000	0.000	0.000	0.131	-1.0	60
Bus139	24	0.1	0.650	0.086	0.071	0.078	0.198	-1.0	89

Table 5.6: QQ Distance performance metrics for all buses with fixed window size (24) and threshold (0.1)

The results show considerable variations in QQ Distance performance across different buses. Bus116 demonstrates the highest accuracy (0.782), but with low recall (0.125), indicating a tendency to miss true changes. Bus119 shows a good balance between precision and recall, resulting in the highest F1 score (0.327) among all buses. Bus121 and Bus135 show no detection capability, suggesting that the chosen parameters might not be suitable for these buses. Figure A.4 in Appendix A.2 provides a visual representation of the QQ Distance algorithm's performance across all buses with fixed parameters.

5.3.2 Parameter Sensitivity Analysis

To investigate the QQ Distance algorithm's sensitivity to parameter changes, we conducted detailed analyses on Bus119, varying the threshold and window size independently.

Threshold Sensitivity

Table 5.7 presents the results for Bus119 with varying thresholds and a fixed window size of 24.

Bus	Window Size	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus119	24	0.01	0.208	0.208	1.000	0.344	1.000	0.0	0
Bus119	24	0.05	0.208	0.208	1.000	0.344	1.000	0.0	0
Bus119	24	0.1	0.236	0.200	0.893	0.327	0.936	-1.0	10
Bus119	24	0.2	0.358	0.121	0.333	0.177	0.636	-1.0	55
Bus119	24	0.4	0.537	0.036	0.048	0.041	0.334	-1.0	58

Table 5.7: QQ Distance performance metrics for Bus119 with varying thresholds and fixed window size (24)

The results demonstrate the significant impact of threshold variation on QQ Distance performance. Lower thresholds (0.01 and 0.05) result in perfect recall but also the highest false alarm rates. As the threshold increases, we observe improved accuracy and reduced false alarm rates, but at the cost of significantly reduced recall. The threshold of 0.1 appears to provide the best balance between accuracy, precision, and recall for Bus119. Figure A.5 in Appendix A.2 illustrates the impact of varying the threshold on QQ Distance performance for Bus119.

Window Size Sensitivity

Table 5.8 presents the results for Bus119 with varying window sizes and a fixed threshold of 0.1.

Bus	Window Size	Threshold	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay	Detection Time
Bus119	6	0.1	0.229	0.203	0.982	0.336	0.959	-1.0	0
Bus119	12	0.1	0.207	0.202	0.994	0.336	0.992	-1.0	1
Bus119	24	0.1	0.236	0.200	0.893	0.327	0.936	-1.0	10
Bus119	48	0.1	0.237	0.221	0.970	0.360	0.971	-1.0	0
Bus119	96	0.1	0.322	0.264	0.940	0.413	0.887	-1.0	0

Table 5.8: QQ Distance performance metrics for Bus119 with varying window sizes and fixed threshold (0.1)

The window size has a noticeable effect on QQ Distance performance. Smaller window sizes (6 and 12) result in higher recall but also higher false alarm rates. As the window size increases, we observe improved accuracy and F1 scores, with the best overall performance at a window size of 96. However, the differences in performance across different window sizes are relatively small, suggesting that the QQ Distance algorithm is somewhat robust to changes in window size for Bus119. Figure A.6 in Appendix A.2 illustrates the impact of varying the window size on QQ Distance performance for Bus119.

5.4 Geometric Entropy Minimization (GEM) Algorithm Performance

The Geometric Entropy Minimization (GEM) algorithm was evaluated across different buses with fixed parameters and then analyzed for sensitivity to parameter changes on a specific bus.

5.4.1 Performance Across Buses

Table 5.9 presents the GEM algorithm's performance across all buses with fixed parameters: K=16, Alpha=0.2, and Threshold=4.

Bus	K	Alpha	Threshold	d	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	16	0.2	4	3	0.855	0.927	0.685	0.788	0.035	1.5
Bus116	16	0.2	4	3	0.393	0.393	1.000	0.565	1.000	0.0
Bus117	16	0.2	4	3	0.862	0.958	0.679	0.794	0.019	11.0
Bus118	16	0.2	4	3	0.836	0.962	0.607	0.745	0.015	8.5
Bus119	16	0.2	4	3	0.859	0.950	0.679	0.792	0.023	11.5
Bus121	16	0.2	4	3	0.393	0.393	1.000	0.565	1.000	0.0
Bus135	16	0.2	4	3	0.766	0.895	0.458	0.606	0.035	-1.0
Bus139	16	0.2	4	3	0.735	0.772	0.464	0.580	0.089	8.0

Table 5.9: GEM performance metrics for all buses with fixed parameters (K=16, Alpha=0.2, Threshold=4)

The results show significant variations in GEM performance across different buses. Bus117 and Bus119 demonstrate the best overall performance with high accuracy, precision, and F1 scores. Bus116 and Bus121 show perfect recall but low precision, indicating a tendency to over-detect changes. Bus135 and Bus139 show moderate performance, suggesting that the chosen parameters might not be optimal for these buses. Figure A.7 in Appendix A.2 provides a visual representation of the GEM algorithm's performance across all buses with fixed parameters.

5.4.2 Parameter Sensitivity Analysis

To investigate the GEM algorithm's sensitivity to parameter changes, we conducted detailed analyses on Bus115, varying Alpha, K, and Threshold independently.

Alpha Sensitivity

Table 5.10 presents the results for Bus115 with varying Alpha values and fixed K=16 and Threshold=4.

Bus	K	Alpha	Threshold	d	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	16	0.01	4	3	0.609	1.000	0.006	0.012	0.000	25.0
Bus115	16	0.05	4	3	0.628	1.000	0.054	0.102	0.000	24.5
Bus115	16	0.1	4	3	0.649	1.000	0.107	0.194	0.000	11.5
Bus115	16	0.2	4	3	0.855	0.927	0.685	0.788	0.035	1.5
Bus115	16	0.4	4	3	0.487	0.433	0.988	0.603	0.838	0.5

Table 5.10: GEM performance metrics for Bus115 with varying Alpha values

The results demonstrate the significant impact of Alpha on GEM performance. Lower Alpha values result in higher precision but very low recall, while higher Alpha values increase recall at the cost of precision and increased false alarm rates. An Alpha value of 0.2 appears to provide the best balance for Bus115. Figure A.8 in Appendix A.2 illustrates the impact of varying Alpha on GEM performance for Bus115.

K Sensitivity

Table 5.11 presents the results for Bus115 with varying K values and fixed Alpha=0.2 and Threshold=4.

Bus	K	Alpha	Threshold	d	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	2	0.2	4	3	0.883	0.883	0.810	0.845	0.069	11.5
Bus115	4	0.2	4	3	0.904	0.926	0.821	0.871	0.042	11.0
Bus115	8	0.2	4	3	0.878	0.939	0.738	0.827	0.031	11.5
Bus115	16	0.2	4	3	0.855	0.927	0.685	0.788	0.035	1.5
Bus115	32	0.2	4	3	0.810	0.931	0.560	0.699	0.027	1.0

Table 5.11: GEM performance metrics for Bus115 with varying K values

The K value affects GEM performance, with the best overall performance achieved at K=4. As K increases, we observe a general trend of decreasing recall and F1 score, although precision remains relatively stable. Figure A.9 in Appendix A.2 illustrates the impact of varying K on GEM performance for Bus115.

Threshold Sensitivity

Table 5.12 presents the results for Bus115 with varying Threshold values and fixed K=16 and Alpha=0.2.

Bus	K	Alpha	Threshold	d	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	16	0.2	2	3	0.888	0.900	0.804	0.849	0.058	0.0
Bus115	16	0.2	4	3	0.855	0.927	0.685	0.788	0.035	1.5
Bus115	16	0.2	8	3	0.756	0.957	0.399	0.563	0.012	12.0
Bus115	16	0.2	16	3	0.651	1.000	0.113	0.203	0.000	25.0
Bus115	16	0.2	32	3	0.607	0.000	0.000	0.000	0.000	0.0

Table 5.12: GEM performance metrics for Bus115 with varying Threshold values

The Threshold significantly impacts GEM performance. Lower thresholds result in higher recall but increased false alarm rates, while higher thresholds improve precision at the cost of recall. A threshold of 2 appears to provide the best balance for Bus115. Figure A.10 in Appendix A.2 illustrates the impact of varying the Threshold on GEM performance for Bus115.

5.5 Principal Component Analysis (PCA) Algorithm Performance

The Principal Component Analysis (PCA) algorithm was evaluated across different buses with fixed parameters and then analyzed for sensitivity to parameter changes on a specific bus.

5.5.1 Performance Across Buses

Table 5.13 presents the PCA algorithm's performance across all buses with fixed parameters: Gamma=0.9, Threshold=4, and Alpha=0.2.

Bus	d	Gamma	Threshold	Alpha	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	3	0.9	4	0.2	0.742	0.983	0.351	0.518	0.004	2.5
Bus116	3	0.9	4	0.2	0.681	0.682	0.357	0.469	0.108	31.0
Bus117	3	0.9	4	0.2	0.703	0.815	0.315	0.455	0.046	11.5
Bus118	3	0.9	4	0.2	0.660	0.695	0.244	0.361	0.069	9.0
Bus119	3	0.9	4	0.2	0.696	0.806	0.298	0.435	0.046	11.5
Bus121	3	0.9	4	0.2	0.696	0.667	0.452	0.539	0.147	11.5
Bus135	3	0.9	4	0.2	0.754	1.000	0.375	0.545	0.000	0.0
Bus139	3	0.9	4	0.2	0.796	0.879	0.560	0.684	0.050	7.5

Table 5.13: PCA performance metrics for all buses with fixed parameters (Gamma=0.9, Threshold=4, Alpha=0.2)

The results show variations in PCA performance across different buses. Bus139 demonstrates the best overall performance with high accuracy, precision, and F1 score. Bus135 shows perfect precision but lower recall, indicating a conservative detection approach. Other buses show moderate performance, suggesting that the chosen parameters might not be optimal for all buses. Figure A.11 in Appendix A.2 provides a visual representation of the PCA algorithm's performance across all buses with fixed parameters.

5.5.2 Parameter Sensitivity Analysis

To investigate the PCA algorithm's sensitivity to parameter changes, we conducted detailed analyses on Bus115, varying Alpha, Gamma, and Threshold independently.

Alpha Sensitivity

Table 5.14 presents the results for Bus115 with varying Alpha values and fixed Gamma=0.9 and Threshold=4.

Bus	d	Gamma	Threshold	Alpha	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	3	0.9	4	0.01	0.607	0.000	0.000	0.000	0.000	0.0
Bus115	3	0.9	4	0.05	0.623	1.000	0.042	0.080	0.000	25.0
Bus115	3	0.9	4	0.1	0.649	1.000	0.107	0.194	0.000	11.5
Bus115	3	0.9	4	0.2	0.742	0.983	0.351	0.518	0.004	2.5
Bus115	3	0.9	4	0.4	0.471	0.426	0.994	0.596	0.869	-1.0

Table 5.14: PCA performance metrics for Bus115 with varying Alpha values

The results demonstrate the significant impact of Alpha on PCA performance. Lower Alpha values result in higher precision but very low recall, while higher Alpha values increase recall at the cost of precision and increased false alarm rates. An Alpha value of 0.2 appears to provide the best balance for Bus115. Figure A.12 in Appendix A.2 illustrates the impact of varying Alpha on PCA performance for Bus115.

Gamma Sensitivity

Table 5.15 presents the results for Bus115 with varying Gamma values and fixed Alpha=0.2 and Threshold=4.

Bus	d	Gamma	Threshold	Alpha	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	3	0.7	4	0.2	0.742	0.983	0.351	0.518	0.004	2.5
Bus115	3	0.8	4	0.2	0.742	0.983	0.351	0.518	0.004	2.5
Bus115	3	0.9	4	0.2	0.742	0.983	0.351	0.518	0.004	2.5
Bus115	3	0.95	4	0.2	0.696	0.806	0.298	0.435	0.046	11.5
Bus115	3	0.99	4	0.2	0.492	0.262	0.161	0.199	0.293	-1.0

Table 5.15: PCA performance metrics for Bus115 with varying Gamma values

The Gamma value shows little effect on performance for lower values (0.7 to 0.9), but performance degrades significantly for very high values (0.95 and 0.99). This suggests that retaining too much variance in the PCA transformation may lead to overfitting. Figure A.13 in Appendix A.2 illustrates the impact of varying Gamma on PCA performance for Bus115.

Threshold Sensitivity

Table 5.16 presents the results for Bus115 with varying Threshold values and fixed Gamma=0.9 and Alpha=0.2.

Bus	d	Gamma	Threshold	Alpha	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Bus115	3	0.9	2	0.2	0.759	0.857	0.464	0.602	0.050	-1.0
Bus115	3	0.9	4	0.2	0.742	0.983	0.351	0.518	0.004	2.5
Bus115	3	0.9	8	0.2	0.674	1.000	0.173	0.294	0.000	15.0
Bus115	3	0.9	16	0.2	0.607	0.000	0.000	0.000	0.000	0.0
Bus115	3	0.9	32	0.2	0.607	0.000	0.000	0.000	0.000	0.0

Table 5.16: PCA performance metrics for Bus115 with varying Threshold values

The Threshold significantly impacts PCA performance. Lower thresholds result in higher recall but increased false alarm rates, while higher thresholds improve precision at the cost of recall. A threshold of 2 appears to provide the best balance between precision and recall for Bus115. Figure A.14 in Appendix A.2 illustrates the impact of varying the Threshold on PCA performance for Bus115.

5.6 Fusion Strategies Performance

We evaluated the performance of two fusion strategies, Method A and Method B, on the CUSUM algorithm results across multiple buses. These strategies aim to improve detection accuracy by combining information from multiple sources.

5.6.1 CUSUM Fusion Strategies Results

Table 5.17 presents the performance metrics for both Method A and Method B fusion strategies applied to CUSUM results with a threshold of 2.0. Figure 5.3 provides a visual comparison of the fusion strategies' performance metrics.

Method	Detection Time	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Method A (At least one bus)	1	0.198	0.197	1.000	0.329	0.999	-1.0
Method A (All buses)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method A (p=0.1)	0	0.196	0.196	1.000	0.328	1.000	0.0
Method A (p=0.2)	1	0.198	0.197	1.000	0.329	0.999	-1.0
Method A (p=0.5)	3	0.202	0.198	1.000	0.330	0.993	-1.0
Method A (p=0.7)	16	0.253	0.208	1.000	0.345	0.930	-1.0
Method A (p=0.9)	394	0.692	0.362	0.738	0.485	0.319	-1.0
Method B (average, average)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (average, minimum)	1	0.198	0.197	1.000	0.329	0.999	-1.0
Method B (average, maximum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (average, median)	82	0.641	0.000	0.000	0.000	0.202	-1.0
Method B (median, average)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (median, minimum)	1	0.199	0.197	1.000	0.329	0.997	-1.0
Method B (median, maximum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (median, median)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, average)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, minimum)	1	0.199	0.197	1.000	0.329	0.997	-1.0
Method B (outlier_detection, maximum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, median)	82	0.641	0.000	0.000	0.000	0.202	-1.0

Table 5.17: CUSUM Fusion Strategies Performance Metrics (Threshold = 2.0)

The results show significant variations in performance across different fusion strategies:

- **Method A**: The "At least one bus" and low percentage (p=0.1, 0.2, 0.5) strategies achieve perfect recall but suffer from very high false alarm rates. As the percentage increases, we observe improved accuracy and reduced false alarm rates, with the best balance achieved at p=0.9.
- **Method B**: Most Method B strategies either fail to detect any changes (accuracy of 0.804 with no true positives) or suffer from very high false alarm rates. The exceptions are the "average, median" and "outlier_detection, median" strategies, which show improved accuracy but fail to detect any true positives.

5.6.2 GLR Fusion Strategies Results

We applied both Method A and Method B fusion strategies to the Generalized Likelihood Ratio (GLR) algorithm results across multiple buses. The performance metrics for these fusion strategies are presented in Table A.1 in Appendix A.1, with a window size of 24 and GLR threshold of 2000. The performance comparison of these fusion strategies is visually represented in Figure A.15 in Appendix A.2.

The results reveal interesting patterns in the performance of different fusion strategies for the GLR algorithm:

- **Method A**: The "At least one bus" and low percentage (p=0.1, 0.2) strategies achieve perfect recall but suffer from high false alarm rates and low accuracy. The strategy with

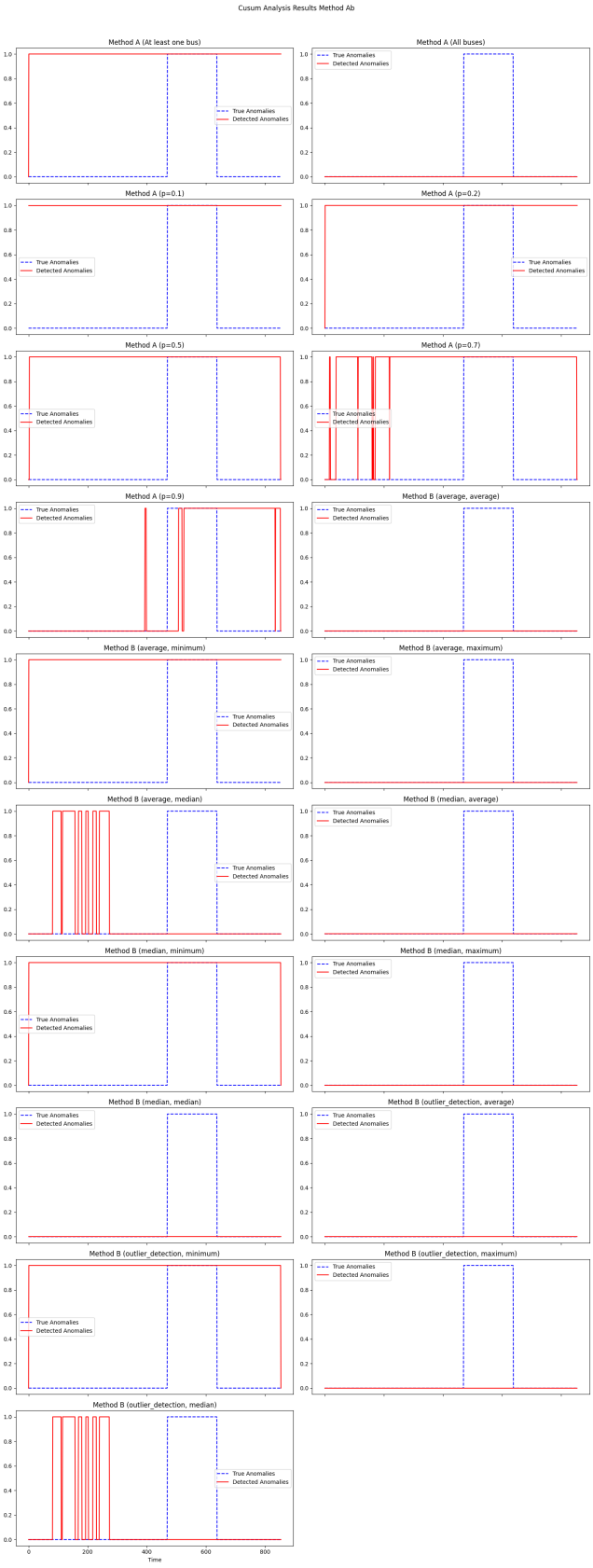

Figure 5.3: CUSUM Fusion Strategies Performance Comparison

p=0.5 shows the best overall performance, with high accuracy (0.736), good precision (0.423), and high recall (0.946), resulting in the highest F1 score (0.585) among all strategies. Interestingly, strategies with higher percentages (p=0.7, 0.9) fail to detect any changes, suggesting that requiring agreement from a large majority of buses may be too stringent for effective change detection.

- **Method B**: Surprisingly, all variations of Method B fail to detect any changes, resulting in an accuracy of 0.804 but with no true positives. This suggests that the statistical aggregation methods employed in Method B may be too conservative for the GLR algorithm, potentially smoothing out the signals that indicate changes.

These results highlight several key insights:

- The GLR algorithm appears to be more sensitive to the choice of fusion strategy compared to the CUSUM algorithm.
- Method A with a moderate consensus requirement (p=0.5) provides the best balance between detection accuracy and false alarm rate for the GLR algorithm.
- Method B strategies, which performed moderately well for CUSUM, are ineffective for GLR. This suggests that the choice of fusion strategy should be tailored to the specific change detection algorithm being used.
- The stark contrast in performance between different parameter settings (e.g., p=0.5 vs. p=0.7 in Method A) emphasizes the critical importance of careful parameter tuning in fusion strategies.

5.6.3 QQ Distance Fusion Strategies Results

We applied both Method A and Method B fusion strategies to the QQ Distance algorithm results across multiple buses. The performance metrics for these fusion strategies are presented in Table A.2 in Appendix A.1, with a window size of 24 and threshold of 0.1. The performance comparison of these fusion strategies is visually represented in Figure A.16 in Appendix A.2.

Analysis of the results reveals several interesting patterns in the performance of different fusion strategies for the QQ Distance algorithm:

- **Method A**: The "At least one bus" and low percentage (p=0.1, 0.2) strategies achieve perfect recall but suffer from very low accuracy and high false alarm rates. As the percentage increases, we observe improved accuracy but decreased recall. The strategy with p=0.5 shows a balanced performance with moderate accuracy (0.339) and recall (0.565), resulting in the highest F1 score (0.262) among Method A strategies. Strategies with higher percentages (p=0.7, 0.9) show further improved accuracy but at the cost of significantly reduced recall.
- **Method B**: Most Method B strategies either fail to detect any changes or suffer from very high false alarm rates. However, two strategies stand out:
  - "Method B (median, minimum)" achieves perfect recall with improved accuracy (0.275) compared to similar Method A strategies.

  - "Method B (average, median)" and "Method B (outlier_detection, median)" show high accuracy (0.686) but fail to detect any true positives, resulting in a long expected delay.

These results highlight several key insights for QQ Distance fusion strategies:

- The QQ Distance algorithm shows high sensitivity to the choice of fusion strategy, with performance varying widely across different methods and parameters.
- There is a clear trade-off between accuracy and recall, with strategies that achieve perfect recall suffering from low accuracy and high false alarm rates.
- Method A with moderate consensus (p=0.5) provides a balanced performance, suggesting that requiring agreement from about half of the buses may be optimal for QQ Distance.
- Most Method B strategies struggle with QQ Distance, either failing to detect changes or producing excessive false alarms. However, the "median, minimum" strategy shows promise, achieving perfect recall with improved accuracy compared to similar Method A strategies.
- The poor performance of many Method B strategies suggests that statistical aggregation methods may need to be carefully tailored for QQ Distance to be effective.

5.6.4 GEM Fusion Strategies Results

We applied both Method A and Method B fusion strategies to the Geometric Entropy Minimization (GEM) algorithm results across multiple buses. The performance metrics for these fusion strategies are presented in Table A.3 in Appendix A.1. The performance comparison of these fusion strategies is visually represented in Figure A.17 in Appendix A.2.

Analysis of the results reveals several interesting patterns in the performance of different fusion strategies for the GEM algorithm:

- **Method A**: The "At least one bus" and low percentage (p=0.1, 0.2) strategies achieve perfect recall but suffer from low accuracy (0.393) and high false alarm rates. As the percentage increases, we observe a significant improvement in performance:
  - The strategy with p=0.5 shows the best overall performance, with high accuracy (0.888), precision (0.935), and recall (0.768), resulting in the highest F1 score (0.843) among all strategies.
  - Strategies with higher percentages (p=0.7, 0.9) maintain high accuracy and precision but show a gradual decrease in recall.
- **Method B**: Most Method B strategies either fail to detect any changes or suffer from very high false alarm rates. However, a few strategies show interesting results:
  - "Method B (average, minimum)" and "Method B (average, median)" achieve perfect recall but with low accuracy (0.417 and 0.424 respectively) and high false alarm rates.
  - "Method B (median, minimum)" shows high precision (1.0) and improved accuracy (0.637) but very low recall (0.077).

  - Outlier detection-based strategies perform poorly, with low accuracy and high false alarm rates.

These results highlight several key insights for GEM fusion strategies:

- The GEM algorithm shows high sensitivity to the choice of fusion strategy, with performance varying widely across different methods and parameters.
- Method A with moderate to high consensus (p=0.5 to p=0.7) provides the best balance between accuracy, precision, and recall. This suggests that requiring agreement from about half to two-thirds of the buses is optimal for GEM.
- There is a clear trade-off between recall and precision/accuracy as the consensus requirement increases in Method A.
- Most Method B strategies struggle with GEM, either failing to detect changes or producing excessive false alarms. The "median, minimum" strategy shows some promise with high precision but suffers from low recall.
- The poor performance of outlier detection-based strategies in Method B suggests that this approach may not be suitable for GEM without further refinement.

5.6.5 PCA Fusion Strategies Results

We applied both Method A and Method B fusion strategies to the Principal Component Analysis (PCA) algorithm results across multiple buses. The performance metrics for these fusion strategies are presented in Table A.4 in Appendix A.1, with Gamma=0.9, threshold $h = 4$, and Alpha=0.2. The performance comparison of these fusion strategies is visually represented in Figure A.18 in Appendix A.2.

Analysis of the results reveals several interesting patterns in the performance of different fusion strategies for the PCA algorithm:

- **Method A**:
  - The "At least one bus" and $p = 0.2$ strategies show the best overall performance, with high accuracy (0.850), precision (0.743), recall (0.946), and F1 score (0.832). This suggests that PCA benefits from a more sensitive detection approach.
  - As the percentage increases from $p = 0.5$ to $p = 0.9$, we observe increasing precision but decreasing recall and F1 score. This indicates a trade-off between precision and recall as the consensus requirement becomes stricter.
  - The "All buses" strategy fails to detect any changes, highlighting the challenge of achieving unanimous agreement across all buses.
- **Method B**:
  - Most Method B strategies either fail to detect any changes or suffer from very high false alarm rates.
  - The "average, minimum" and "median, minimum" strategies, as well as the "outlier_detection, minimum" strategy, achieve perfect recall but with low accuracy (0.393) and high false alarm rates.

  - The "average, median" and "outlier_detection, median" strategies show moderate accuracy but fail to detect any true positives, resulting in long expected delays.

These results highlight several key insights for PCA fusion strategies:

- PCA shows high sensitivity to the choice of fusion strategy, with performance varying significantly across different methods and parameters.
- Method A with low consensus requirements ($p = 0.2$ or "At least one bus") provides the best overall performance for PCA. This suggests that PCA benefits from a more sensitive detection approach, where changes detected by even a small number of buses are considered significant.
- There is a clear trade-off between precision and recall in Method A as the consensus requirement increases. This allows for flexibility in choosing the strategy based on whether false positives or false negatives are more costly in a given application.
- Most Method B strategies struggle with PCA, either failing to detect changes or producing excessive false alarms. This indicates that statistical aggregation methods may not be well-suited for PCA-based change detection without further refinement.
- The poor performance of outlier detection-based strategies in Method B suggests that this approach may not be effective for PCA in its current form.

Chapter 6: Discussion

The results of our comprehensive evaluation of nonparametric change detection algorithms on Locational Marginal Price (LMP) data provide valuable insights into the performance and applicability of these methods in the context of energy markets. In this section, we discuss the key findings, their implications, and the broader context of our research.

6.1 Algorithm Performance Comparison

6.1.1 Individual Algorithm Performance

Our analysis of CUSUM, GLR, QQ Distance, GEM, and PCA algorithms reveals significant variations in performance across different buses and parameter settings:

- **CUSUM**: Demonstrated moderate performance with high sensitivity to threshold selection. The optimal threshold varied across buses, highlighting the need for bus-specific tuning.
- **GLR**: Showed high accuracy and F1 scores for some buses (e.g., Bus135) but struggled with others, indicating sensitivity to local data characteristics. The window size parameter significantly influenced performance.
- **QQ Distance**: Exhibited balanced performance across multiple metrics but was highly sensitive to both threshold and window size parameters. This suggests a need for careful parameter tuning in practical applications.
- **GEM**: Demonstrated strong overall performance, particularly in terms of accuracy and F1 score. The algorithm showed robustness across different buses, suggesting good generalization capabilities.
- **PCA**: Achieved high precision but lower recall compared to other methods. Performance was notably sensitive to the Alpha parameter, indicating the importance of proper feature selection in PCA-based change detection.

These findings underscore the importance of algorithm selection and parameter tuning in change detection tasks. No single algorithm consistently outperformed the others across all scenarios, emphasizing the need for context-specific algorithm selection in real-world applications.

6.1.2 Fusion Strategies Effectiveness

The evaluation of fusion strategies (Method A and Method B) revealed several important insights:

- **Method A (Voting-based)**: Generally outperformed Method B, with the optimal percentage threshold varying by algorithm. This suggests that simple voting mechanisms can effectively leverage information from multiple buses.

- **Method B (Statistical Aggregation)**: Showed mixed results, often struggling to balance false alarm rates and detection accuracy. The median-based strategies occasionally showed promise, indicating potential for refinement.
- **Algorithm-Specific Patterns**: Different algorithms benefited from different fusion approaches. For instance, GEM performed well with moderate consensus in Method A, while PCA favored more sensitive detection strategies.

These results highlight the potential of fusion strategies to enhance change detection performance in multi-bus systems. However, they also emphasize the need for careful strategy selection and parameter tuning based on the specific algorithm and application context.

6.2 Implications for Energy Market Monitoring

Our findings have several important implications for the application of nonparametric change detection in energy market monitoring:

- **Algorithm Selection**: The varying performance of algorithms across different buses suggests that a one-size-fits-all approach may not be optimal. Energy market operators may need to employ multiple algorithms or carefully select algorithms based on specific bus characteristics.
- **Parameter Tuning**: The high sensitivity of algorithm performance to parameter settings underscores the importance of adaptive parameter tuning mechanisms in real-world deployments. This could involve periodic re-calibration or the development of online parameter optimization techniques.
- **Multi-Bus Fusion**: The effectiveness of fusion strategies, particularly voting-based methods, suggests that leveraging information from multiple buses can enhance change detection reliability. This approach could be particularly valuable in identifying system-wide anomalies or market-wide events.
- **Early Warning Systems**: The high precision achieved by some algorithms (e.g., PCA) could be leveraged to develop early warning systems for potential market anomalies, allowing operators to investigate subtle changes before they become widespread issues.

6.3 Limitations and Future Work

While our study provides comprehensive insights into nonparametric change detection in LMP data, several limitations and areas for future research should be acknowledged:

- **Data Limitations**: Our analysis focused on a specific dataset of LMP data. Future work should explore the generalizability of these findings to other energy markets and time periods.
- **Computational Efficiency**: We did not extensively evaluate the computational costs of these algorithms. Future research should consider the trade-offs between detection performance and computational requirements, particularly for real-time applications.

- **Adaptive Algorithms**: Developing adaptive versions of these algorithms that can automatically adjust parameters based on changing data characteristics could significantly enhance their practical applicability.
- **Fusion Strategy Refinement**: While our fusion strategies showed promise, there is room for further refinement, particularly in the statistical aggregation methods. Exploring more sophisticated fusion techniques, such as machine learning-based aggregation, could yield improved results.
- **Interpretability**: Future work should focus on enhancing the interpretability of change detection results, particularly for complex algorithms like GEM and PCA. This could involve developing visualization tools or explanatory mechanisms to help operators understand the nature of detected changes.

In conclusion, our comprehensive evaluation of nonparametric change detection algorithms on LMP data provides valuable insights for both researchers and practitioners in the field of energy market monitoring. The varying performance of algorithms across different scenarios underscores the complexity of change detection in this domain and highlights the need for careful algorithm selection, parameter tuning, and potentially the use of fusion strategies in practical applications. As energy markets continue to evolve and face new challenges, the development of robust, adaptive, and interpretable change detection methods will remain a critical area of research.

Chapter 7: Conclusion

This study presents a comprehensive evaluation of nonparametric change detection algorithms applied to Locational Marginal Price (LMP) data in energy markets. We investigated five algorithms—CUSUM, GLR, QQ Distance, GEM, and PCA—and explored two fusion strategies for multi-bus systems. Our research provides several key contributions to the field of change detection in energy market monitoring:

1. **Algorithm Performance**: We demonstrated that the performance of nonparametric change detection algorithms varies significantly across different buses and parameter settings. GEM showed strong overall performance and robustness, while PCA achieved high precision but lower recall. CUSUM, GLR, and QQ Distance exhibited sensitivity to parameter tuning, highlighting the importance of adaptive parameter selection in practical applications.
2. **Fusion Strategies**: Our evaluation of fusion strategies revealed that voting-based methods (Method A) generally outperformed statistical aggregation methods (Method B). This suggests that leveraging consensus across multiple buses can enhance change detection reliability in energy market monitoring systems.
3. **Parameter Sensitivity**: We identified critical parameters for each algorithm and quantified their impact on performance metrics. This knowledge is essential for developing adaptive parameter tuning mechanisms for real-world deployments.
4. **Multi-Bus Analysis**: Our research underscores the potential benefits of multi-bus analysis in change detection tasks. By combining information from multiple buses, we can potentially identify system-wide anomalies and improve overall detection accuracy.
5. **Practical Implications**: We discussed the implications of our findings for energy market monitoring, including the need for algorithm-specific optimization, the potential for early warning systems, and the importance of interpretable results for market operators.

The limitations of our study, including the focus on a specific LMP dataset and the need for further investigation of computational efficiency, provide clear directions for future research. Developing adaptive algorithms, refining fusion strategies, and enhancing result interpretability are promising areas for advancing the field of change detection in energy markets.

In conclusion, our research contributes to the understanding and application of nonparametric change detection methods in the context of energy market monitoring. As energy markets continue to evolve and face new challenges, the insights and methodologies presented in this study can serve as a foundation for developing more robust, efficient, and adaptive monitoring systems. These advancements will be crucial in ensuring the stability, efficiency, and fairness of energy markets in an increasingly complex and dynamic environment.

The field of change detection in energy markets remains rich with opportunities for further research and innovation. Future work should focus on:

- Extending the analysis to diverse energy market datasets to validate the generalizability of our findings.
- Investigating the integration of domain-specific knowledge to enhance detection accuracy and interpretability.
- Exploring the potential of machine learning and deep learning approaches in complementing traditional change detection methods.
- Developing real-time, adaptive systems that can automatically select and tune algorithms based on changing market conditions.

As the energy sector continues its transition towards more sustainable and decentralized systems, the role of accurate and reliable change detection in market monitoring will only grow in importance. The methodologies and insights presented in this study contribute to this critical aspect of energy market management, paving the way for more resilient and efficient energy systems in the future.

Appendix A: Appendix

A.1 Tables

Table A.1: GLR Fusion Strategies Performance Metrics (Window Size = 24, Threshold = 2000)

Method	Detection Time	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Method A (At least one bus)	37	0.240	0.205	1.000	0.341	0.946	-1.0
Method A (All buses)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method A (p=0.1)	0	0.196	0.196	1.000	0.328	1.000	0.0
Method A (p=0.2)	37	0.240	0.205	1.000	0.341	0.946	-1.0
Method A (p=0.5)	479	0.736	0.423	0.946	0.585	0.316	4.0
Method A (p=0.7)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method A (p=0.9)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (average, average)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (average, minimum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (average, maximum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (average, median)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (median, average)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (median, minimum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (median, maximum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (median, median)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, average)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, minimum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, maximum)	-1	0.804	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, median)	-1	0.804	0.000	0.000	0.000	0.000	0.0

Table A.2: QQ Distance Fusion Strategies Performance Metrics (Window Size = 24, Threshold = 0.1)

Method	Detection Time	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Method A (At least one bus)	0	0.223	0.211	1.000	0.349	0.981	-1.0
Method A (All buses)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method A (p=0.1)	0	0.208	0.208	1.000	0.344	1.000	0.0
Method A (p=0.2)	0	0.223	0.211	1.000	0.349	0.981	-1.0
Method A (p=0.5)	54	0.339	0.171	0.565	0.262	0.720	-1.0
Method A (p=0.7)	60	0.595	0.137	0.179	0.155	0.295	-1.0
Method A (p=0.9)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (average, average)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (average, minimum)	0	0.208	0.208	1.000	0.344	1.000	0.0
Method B (average, maximum)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (average, median)	722	0.686	0.000	0.000	0.000	0.134	215.0
Method B (median, average)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (median, minimum)	54	0.275	0.223	1.000	0.364	0.916	-1.0
Method B (median, maximum)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (median, median)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, average)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, minimum)	0	0.208	0.208	1.000	0.344	1.000	0.0
Method B (outlier_detection, maximum)	-1	0.792	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, median)	722	0.686	0.000	0.000	0.000	0.134	215.0

A.2 Figures

Table A.3: GEM Fusion Strategies Performance Metrics

Method	Detection Time	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Method A (At least one bus)	0	0.393	0.393	1.000	0.565	1.000	0.0
Method A (All buses)	91	0.649	1.000	0.107	0.194	0.000	24.0
Method A (p=0.1)	0	0.393	0.393	1.000	0.565	1.000	0.0
Method A (p=0.2)	0	0.393	0.393	1.000	0.565	1.000	0.0
Method A (p=0.5)	46	0.888	0.935	0.768	0.843	0.035	1.5
Method A (p=0.7)	61	0.864	0.951	0.690	0.800	0.023	9.0
Method A (p=0.9)	66	0.766	0.959	0.423	0.587	0.012	11.5
Method B (median, average)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (median, minimum)	116	0.637	1.000	0.077	0.144	0.000	36.5
Method B (median, maximum)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (median, median)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, average)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, minimum)	11	0.169	0.057	0.071	0.063	0.768	-1.0
Method B (outlier_detection, maximum)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, median)	13	0.169	0.048	0.060	0.053	0.761	-1.0

Table A.4: PCA Fusion Strategies Performance Metrics (Gamma = 0.9, h = 4, Alpha = 0.2)

Method	Detection Time	Accuracy	Precision	Recall	F1 Score	False Alarm Rate	Expected Delay
Method A (At least one bus)	43	0.850	0.743	0.946	0.832	0.212	0.0
Method A (All buses)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method A (p=0.1)	0	0.393	0.393	1.000	0.565	1.000	0.0
Method A (p=0.2)	43	0.850	0.743	0.946	0.832	0.212	0.0
Method A (p=0.5)	66	0.724	0.829	0.375	0.516	0.050	11.5
Method A (p=0.7)	66	0.707	0.939	0.274	0.424	0.012	11.5
Method A (p=0.9)	115	0.618	1.000	0.030	0.058	0.000	36.0
Method B (average, average)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (average, minimum)	0	0.393	0.393	1.000	0.565	1.000	0.0
Method B (average, maximum)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (average, median)	416	0.581	0.000	0.000	0.000	0.042	290.0
Method B (median, average)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (median, minimum)	0	0.393	0.393	1.000	0.565	1.000	0.0
Method B (median, maximum)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (median, median)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, average)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, minimum)	0	0.393	0.393	1.000	0.565	1.000	0.0
Method B (outlier_detection, maximum)	-1	0.607	0.000	0.000	0.000	0.000	0.0
Method B (outlier_detection, median)	416	0.595	0.000	0.000	0.000	0.019	290.0

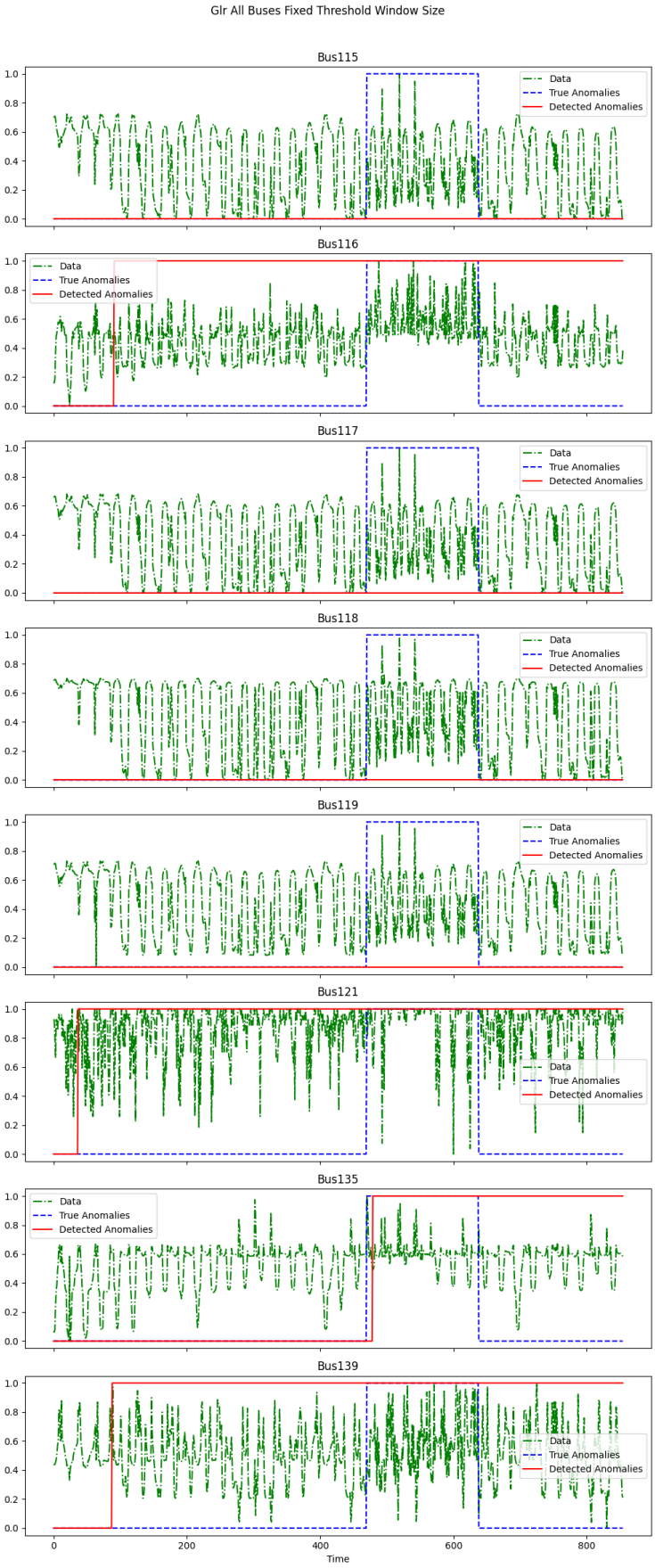

Figure A.1: GLR performance metrics for all buses with fixed window size and threshold

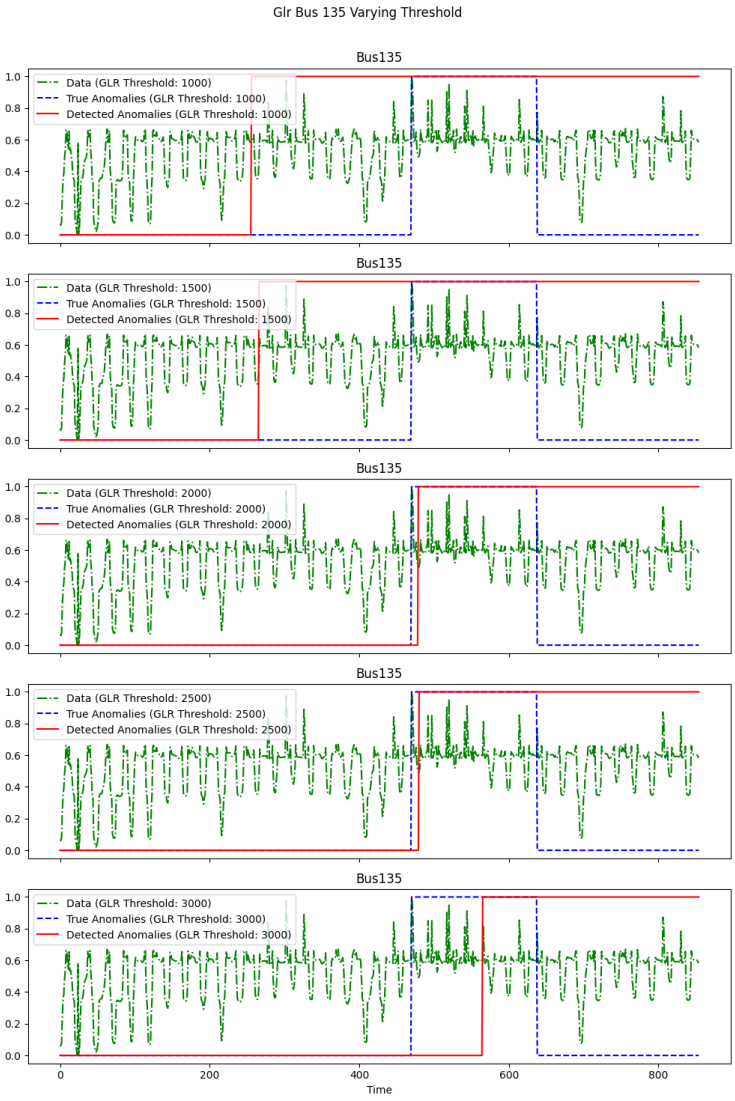

Figure A.2: GLR performance metrics for Bus135 with varying thresholds

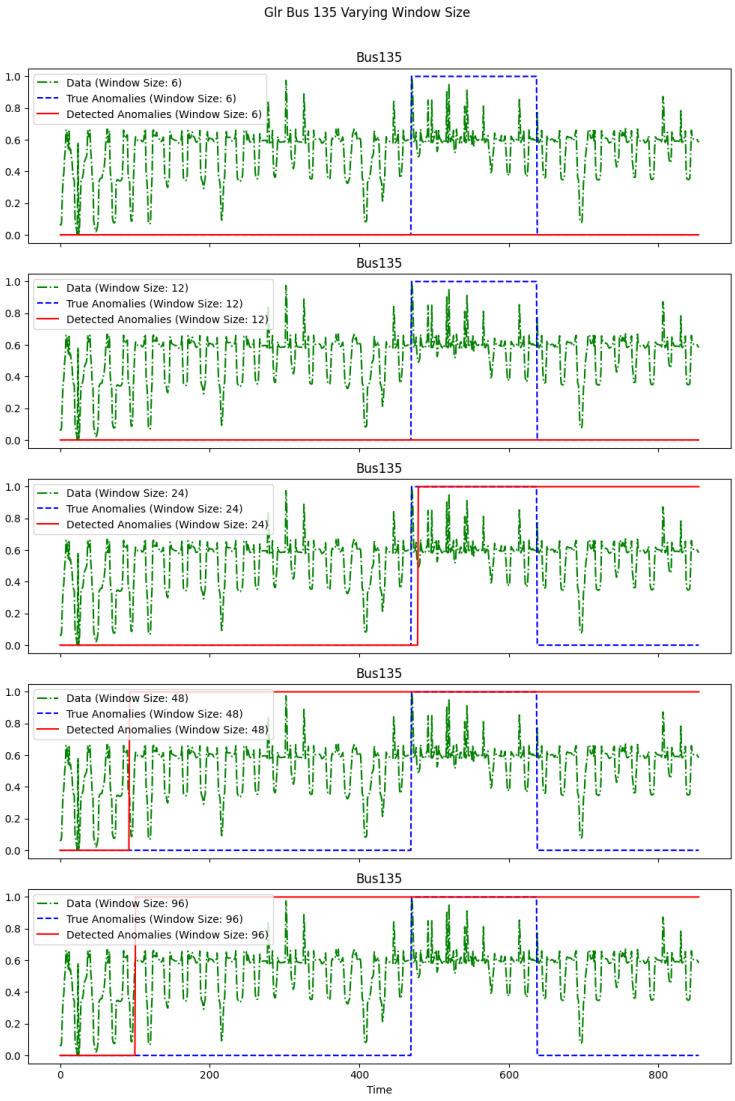

Figure A.3: GLR performance metrics for Bus135 with varying window sizes

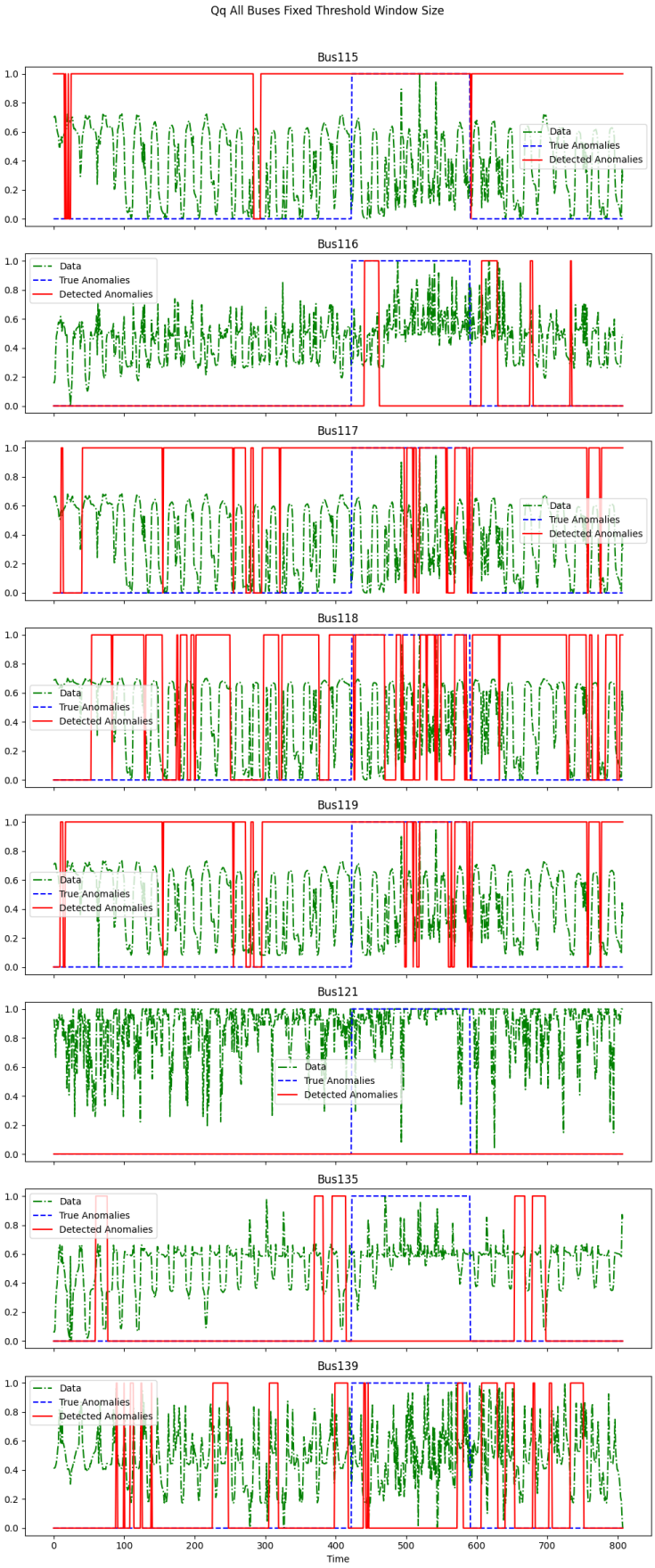

Figure A.4: QQ distance performance metrics for all buses with fixed window size and threshold

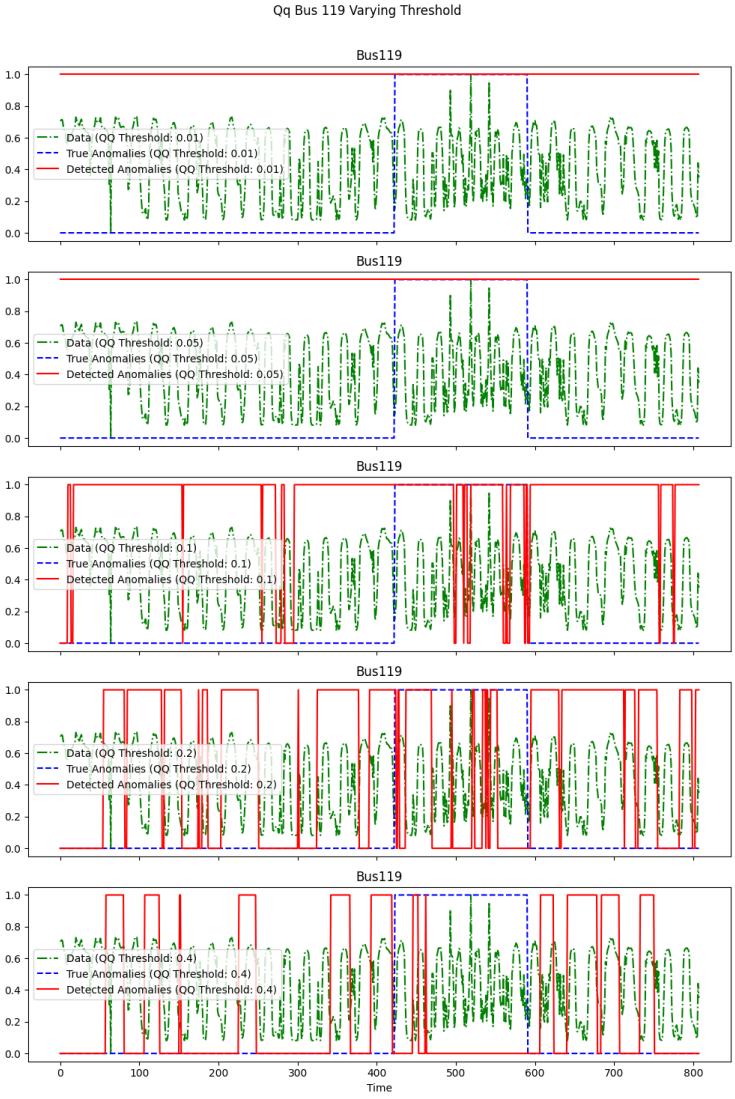

Figure A.5: QQ distance performance metrics for Bus119 with varying thresholds

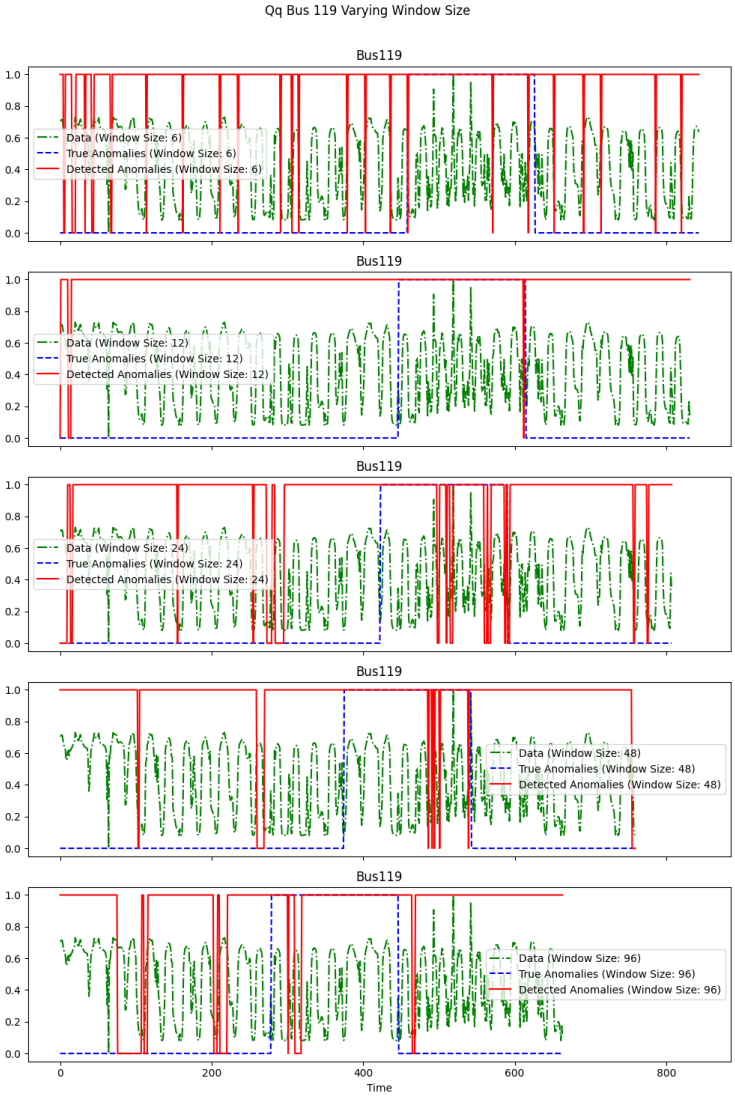

Figure A.6: QQ distance performance metrics for Bus119 with varying window sizes

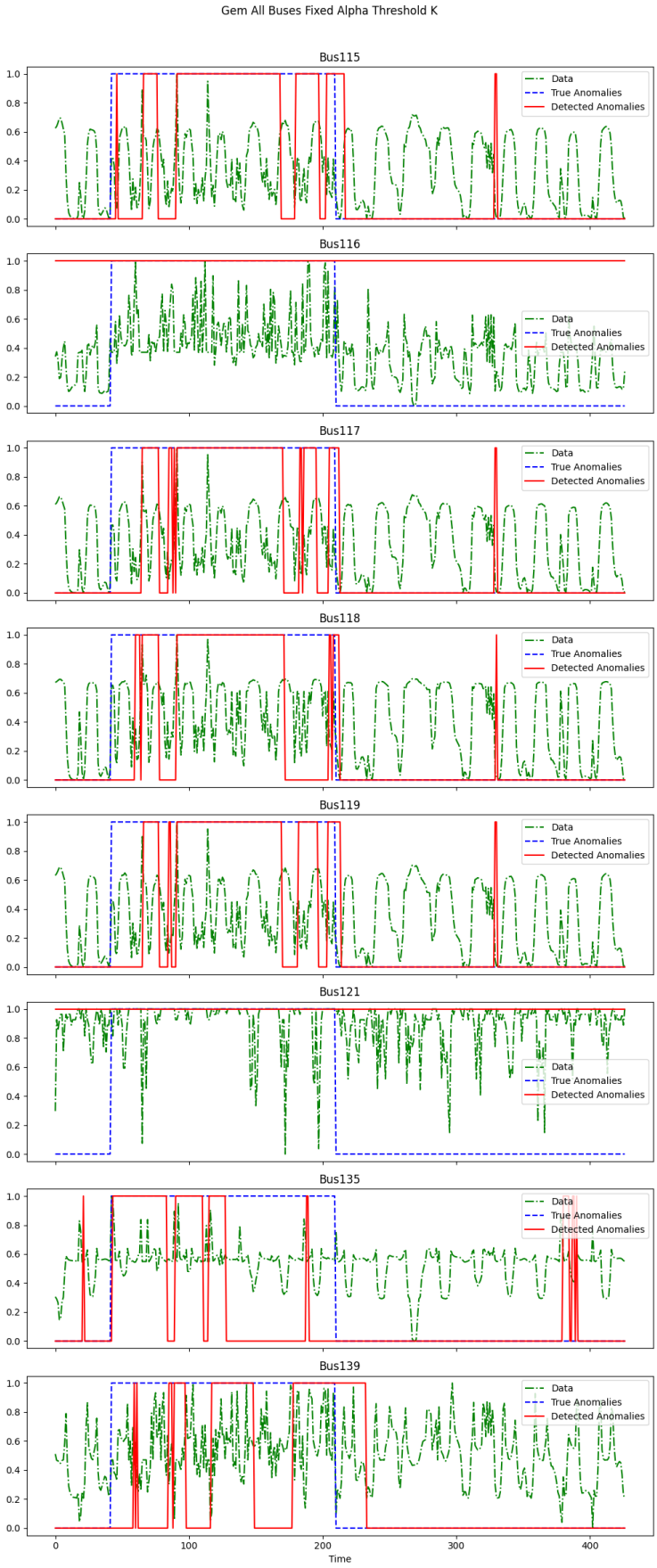

Figure A.7: GEM performance metrics for all buses with fixed alpha, threshold, and k

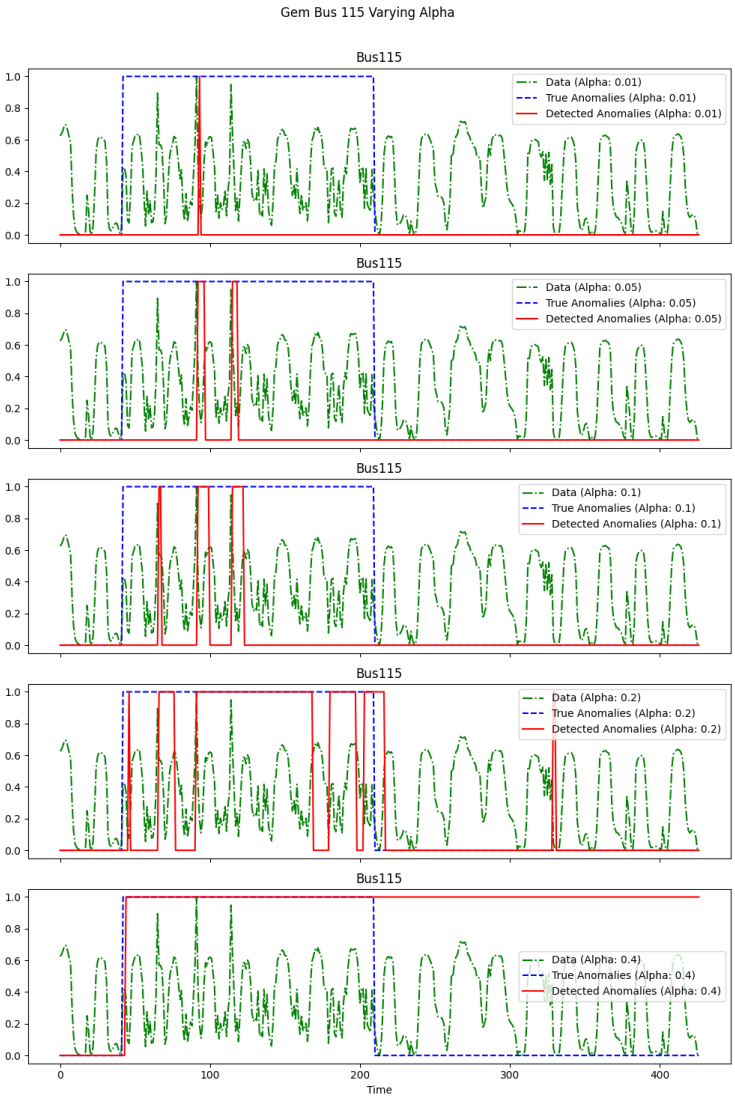

Figure A.8: GEM performance metrics for Bus115 with varying alpha

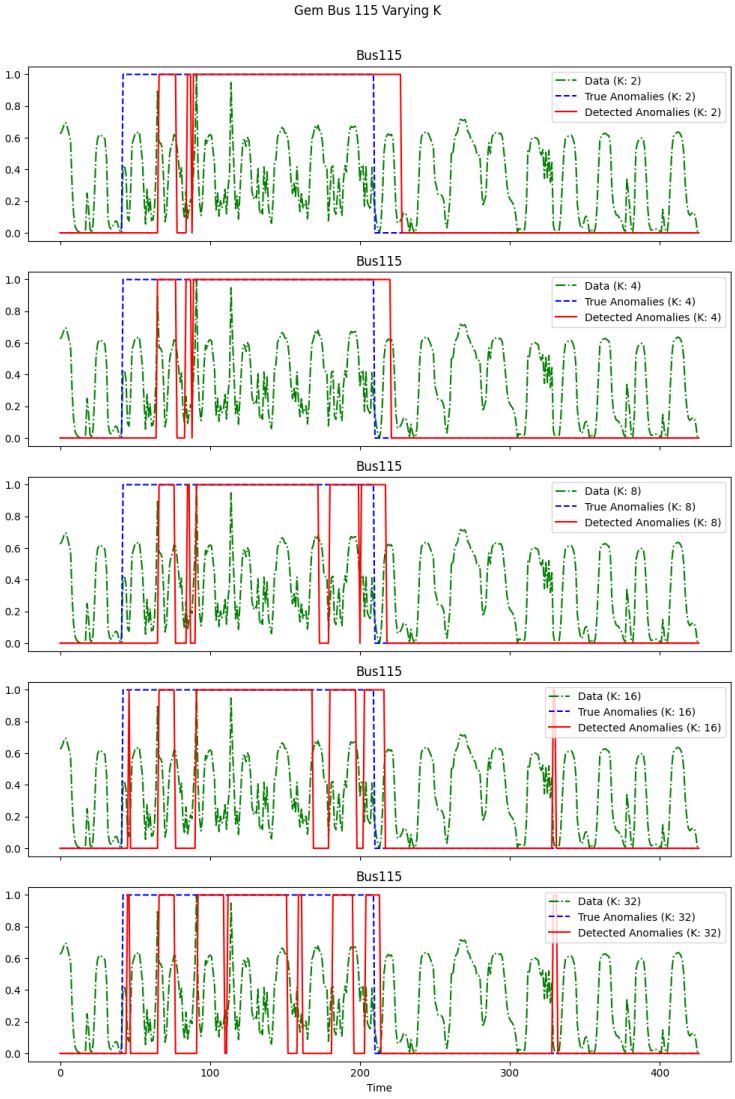

Figure A.9: GEM performance metrics for Bus115 with varying k

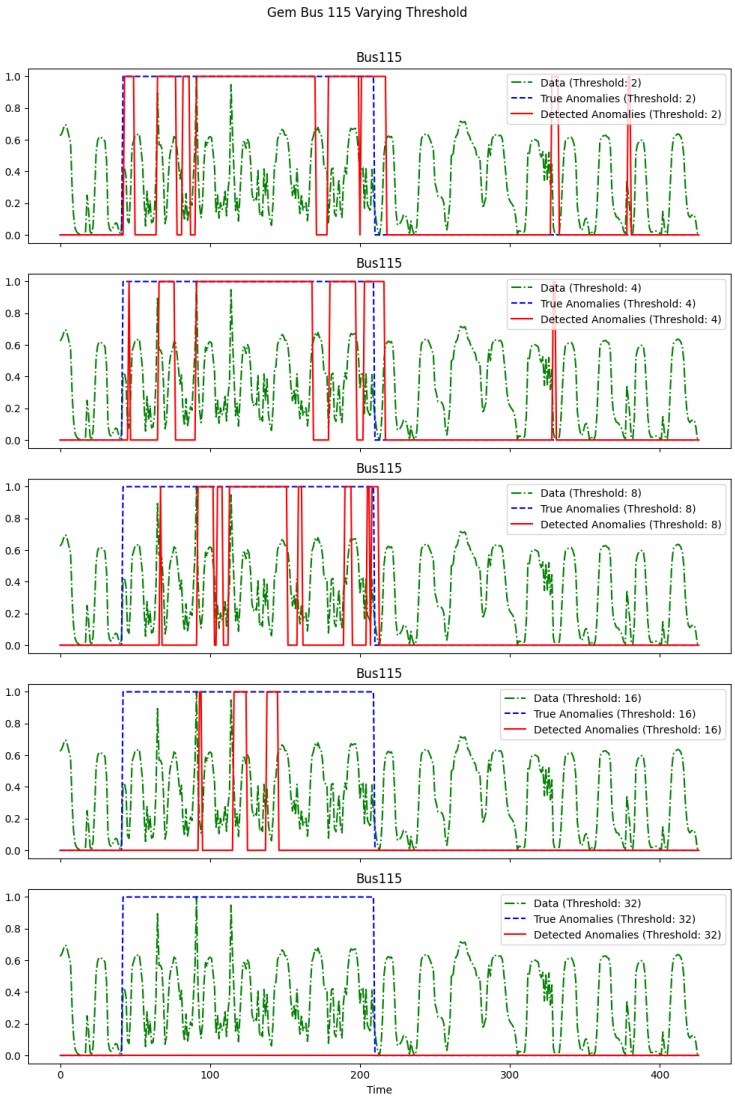

Figure A.10: GEM performance metrics for Bus115 with varying threshold

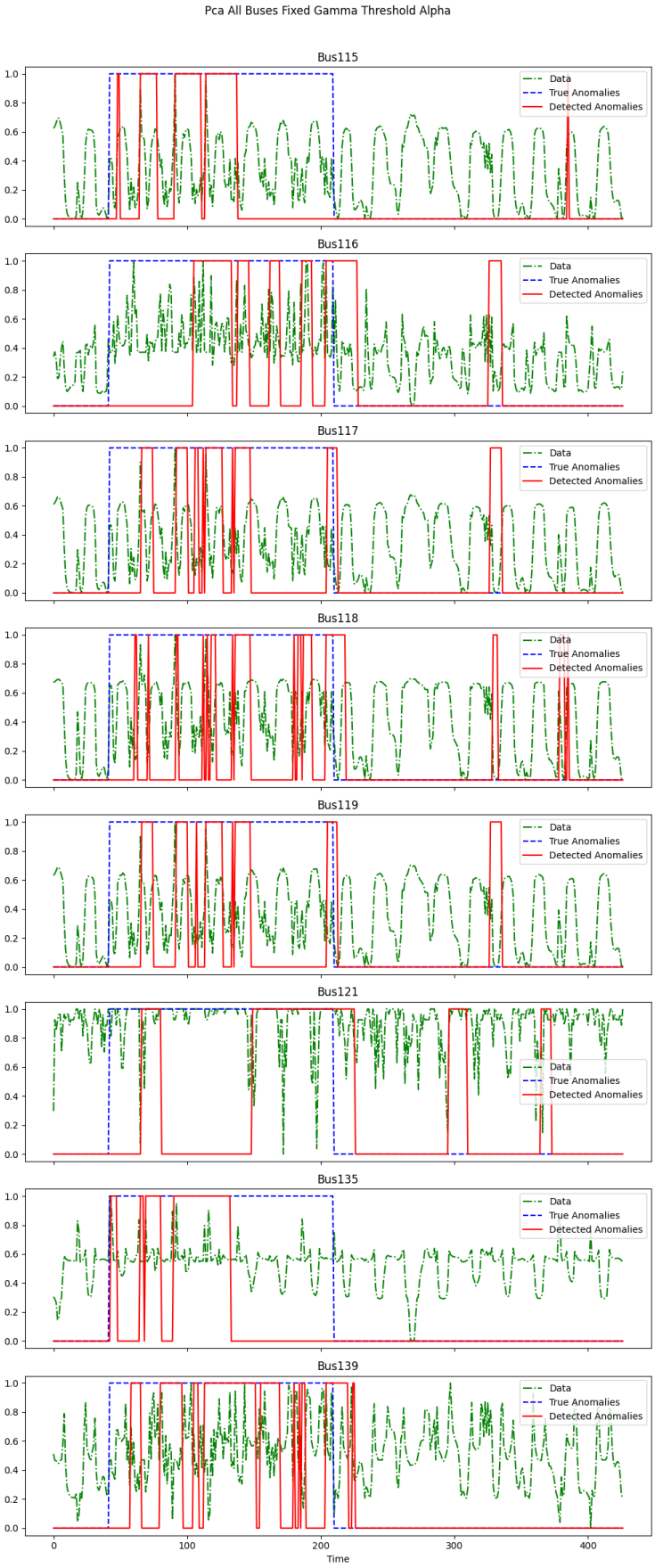

Figure A.11: PCA performance metrics for all buses with fixed gamma, threshold, and alpha

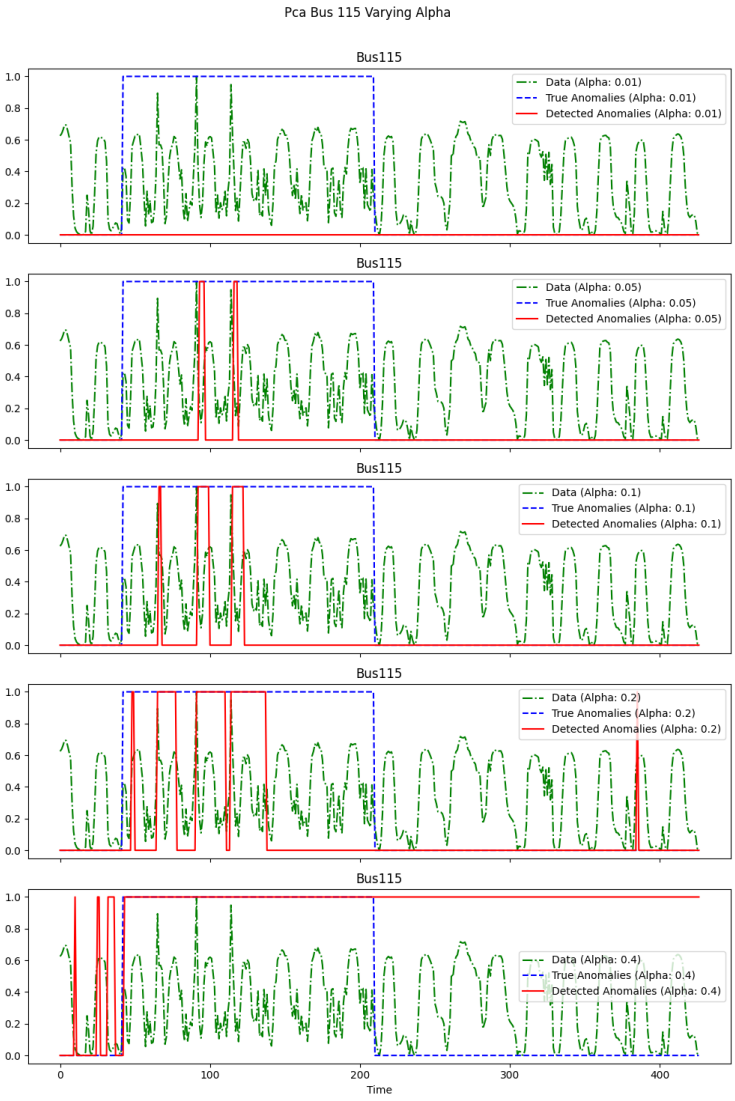

Figure A.12: PCA performance metrics for Bus115 with varying alpha

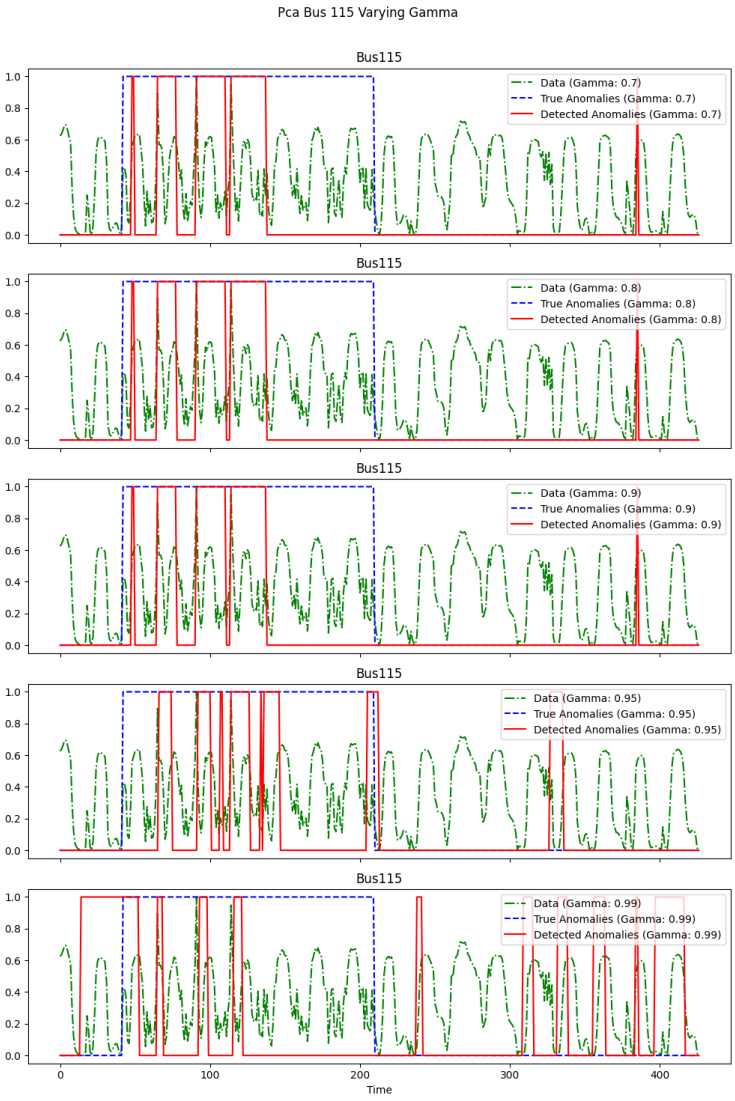

Figure A.13: PCA performance metrics for Bus115 with varying gamma

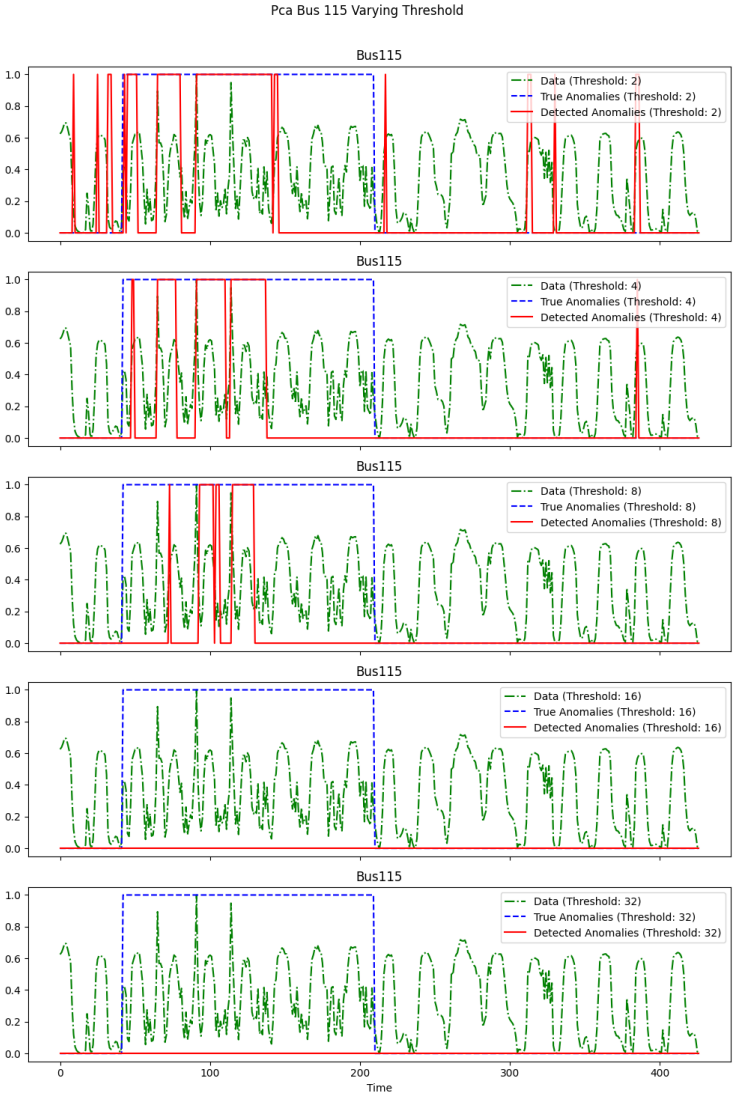

Figure A.14: PCA performance metrics for Bus115 with varying threshold

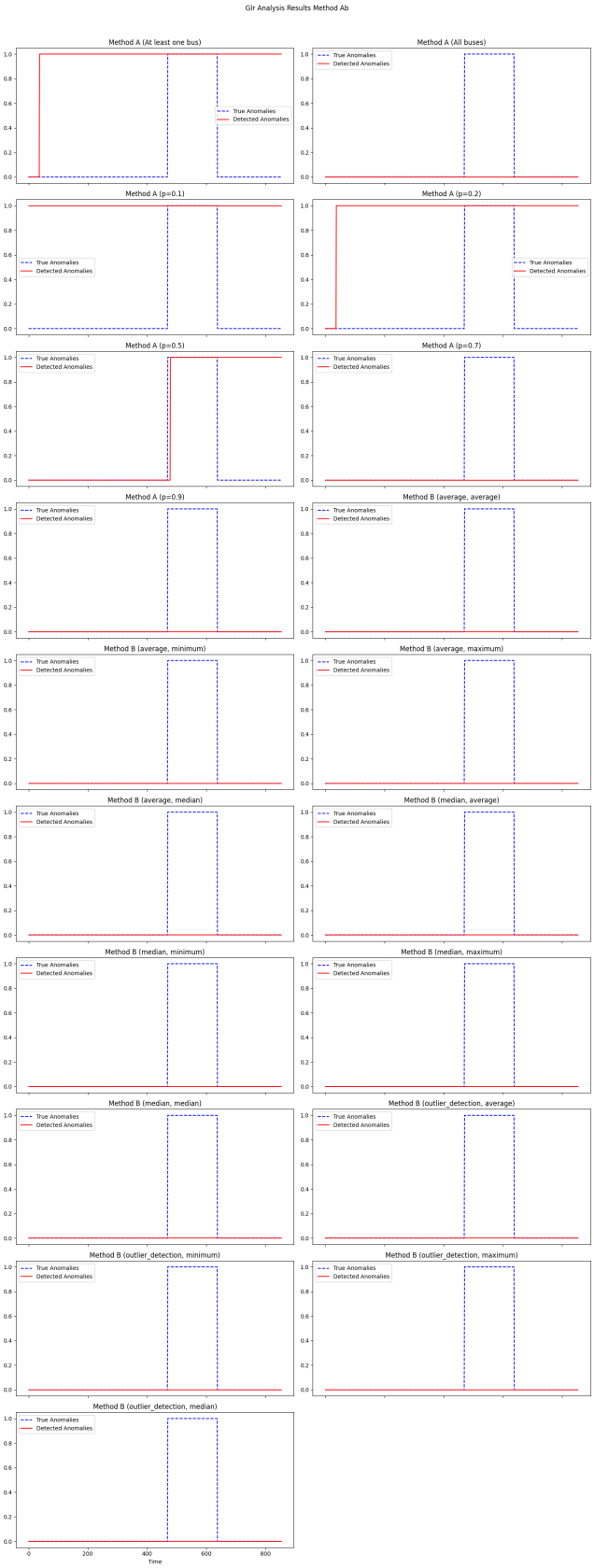

Figure A.15: GLR Fusion Strategies Performance Comparison

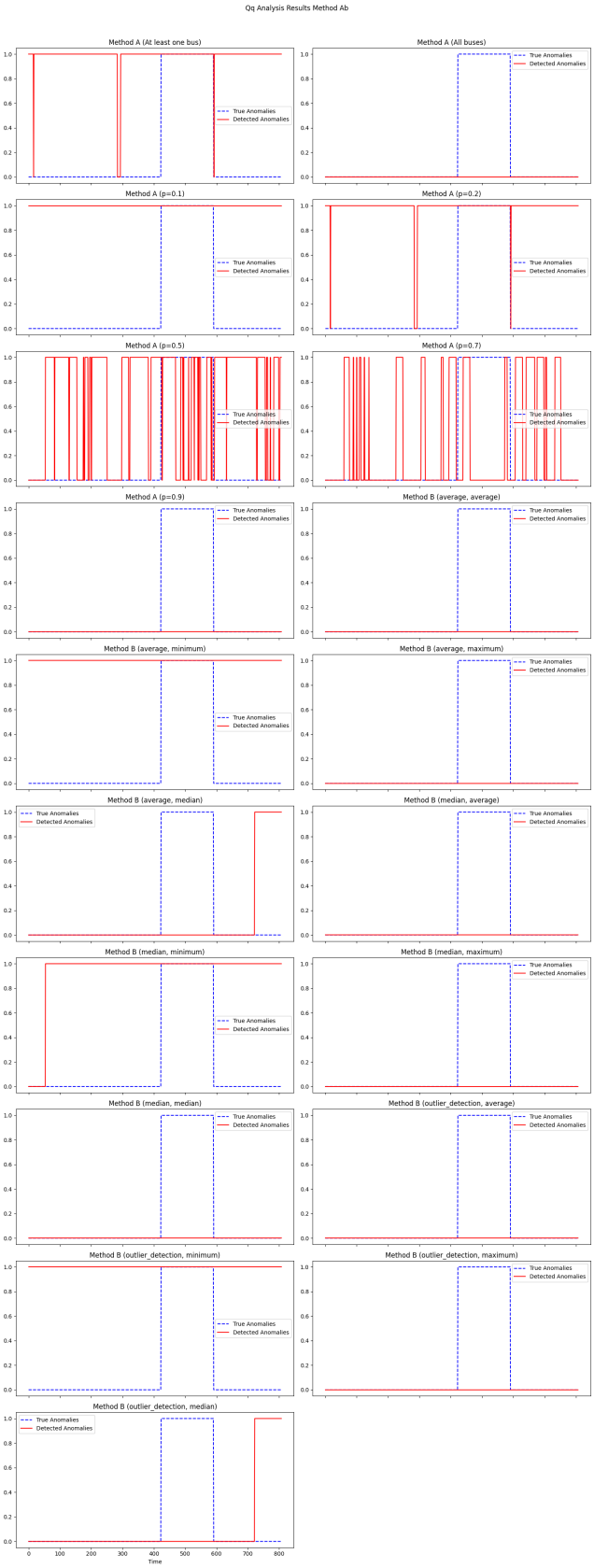

Figure A.16: QQ Distance Fusion Strategies Performance Comparison

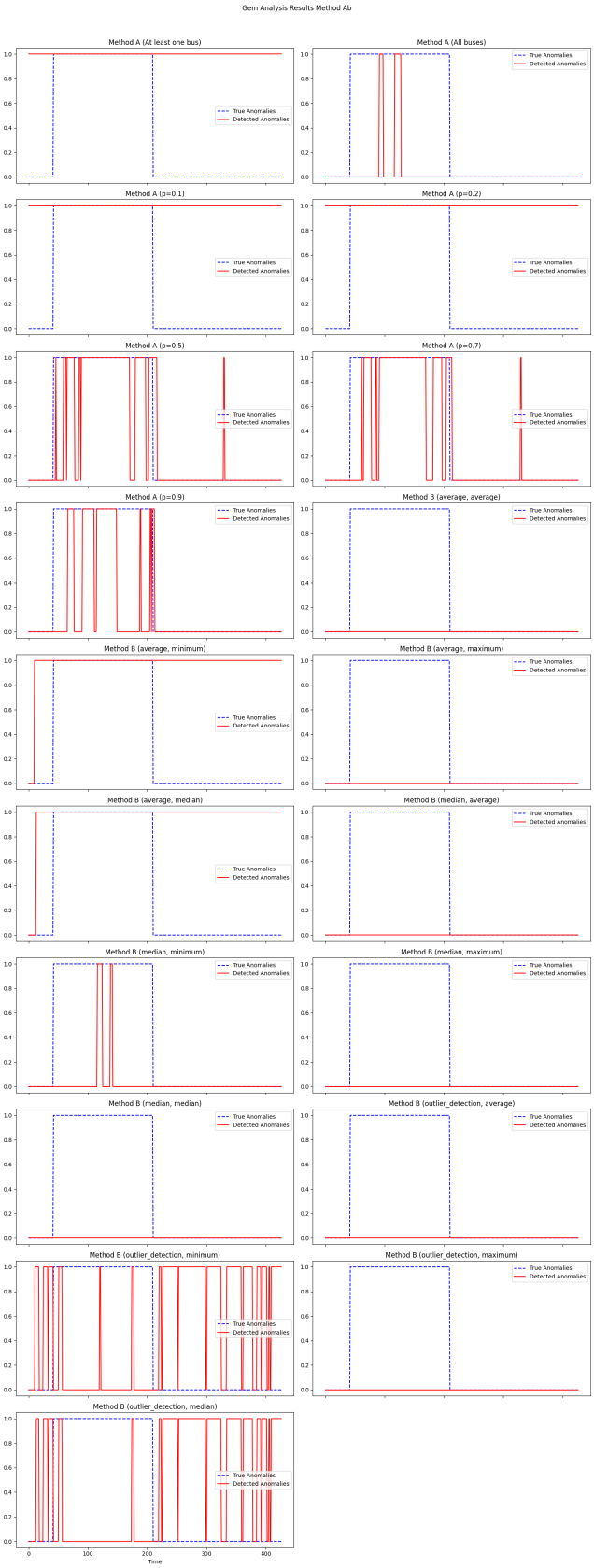

Figure A.17: GEM Fusion Strategies Performance Comparison

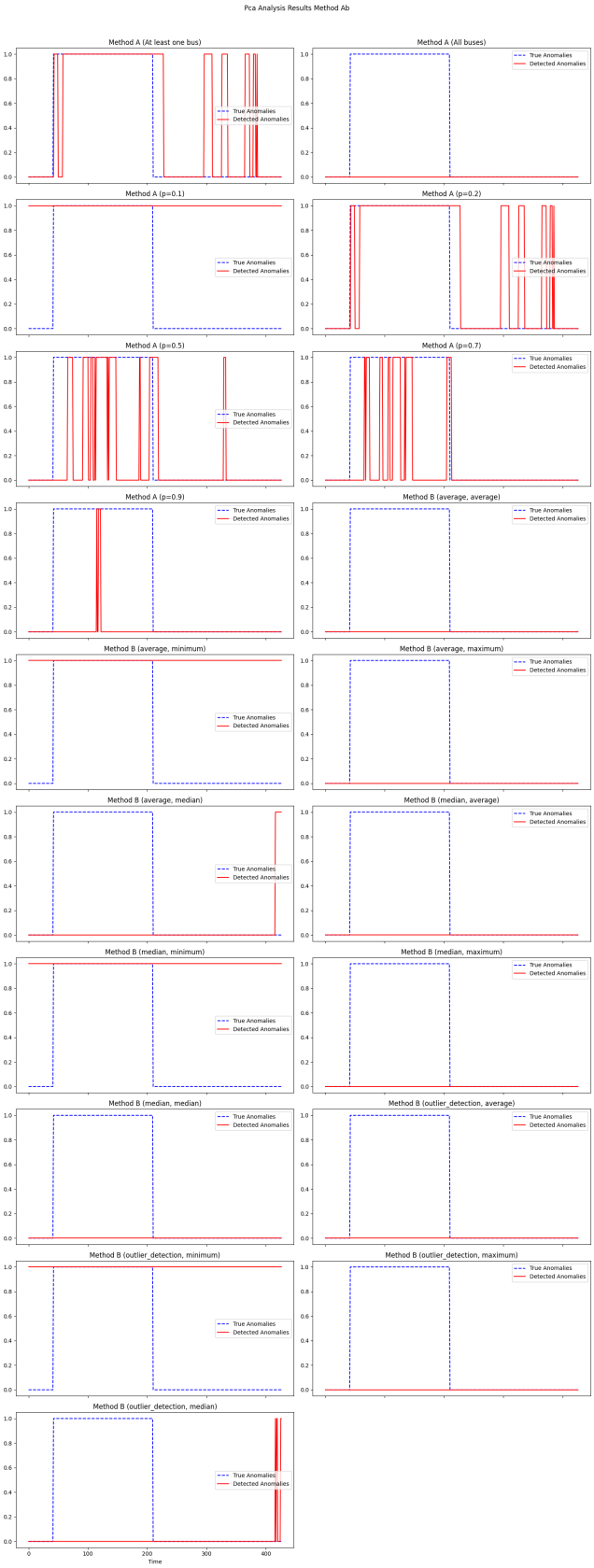

Figure A.18: PCA Fusion Strategies Performance Comparison

Bibliography

[1] Michèle Basseville and Igor Nikiforov. *Detection of Abrupt Change Theory and Application*, volume 15. 04 1993.

[2] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[3] G. Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908, 1971.

[4] Dayu Yang and Hairong Qi. An effective decentralized nonparametric quickest detection approach. pages 2278–2281, 08 2010.

[5] Alfred Hero. Geometric entropy minimization (gem) for anomaly detection and localization. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[6] Mehmet Necip Kurt, Oyetunji Ogundijo, Chong Li, and Xiaodong Wang. Online cyber-attack detection in smart grid: A reinforcement learning approach. *IEEE Transactions on Smart Grid*, 10(5):5174–5185, 2019.

[7] Mehmet Necip Kurt, Yasin Yilmaz, and Xiaodong Wang. Real-time detection of hybrid and stealthy cyber-attacks in smart grid. *IEEE Transactions on Information Forensics and Security*, 14(2):498–513, February 2019.

[8] Kumar Sricharan and Alfred Hero. Efficient anomaly detection using bipartite k-nn graphs. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[9] Yao Xie, Jiaji Huang, and Rebecca Willett. Change-point detection for high-dimensional time series with missing data. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):12–27, February 2013.