EasyChair Preprint

№ 15690

Multi-Criteria Analysis of Concept Drift Detection Algorithms: a Decision-Making Approach

Osama A. Mahdi, Savitri Bevinakoppa and Sarabjot Singh

January 8, 2025

Multi-Criteria Analysis of Concept Drift Detection Algorithms: A Decision-Making Approach

Osama Mahdi
School of IT and Engineering
Melbourne Institute of Technology
Melbourne, Australia
omahdi@mit.edu.au

Savitri Bevinakoppa
School of IT and Engineering
Melbourne Institute of Technology
Melbourne, Australia
sbevinakoppa@mit.edu.au

Sarabjot Singh
School of IT and Engineering
Melbourne Institute of Technology
Melbourne, Australia
sarabjotsingh@academic.mit.edu.au

***Abstract*— Concept Drift is a challenging problem in data streaming, where the underlying data distribution changes over time. Numerous algorithms have been proposed to address this issue, each evaluated using various metrics such as accuracy, runtime, and false alarms. However, a comprehensive evaluation that simultaneously considers all these metrics is lacking. Motivated by this gap, our paper systematically benchmarks eleven leading concept drift detection algorithms using a Multi-Criteria Decision-Making (MCDM) approach to identify the best-performing methods. We employ four datasets and seven performance measures: Average Delay Detection (ADD), Average True Detection (ATD), Average False Alarm (AFA), Average False Negative (AFN), Average Detection Runtime in milliseconds (ARMS), Average Memory Usage in bytes (MUB), and Average Accuracy. Our experimental evaluation and comparison are conducted against eleven existing detectors. The results show that our approach provides a balanced and comprehensive assessment, offering a significant advancement in the evaluation of concept drift detection methods. This paper provides a holistic strategy that integrates multiple performance metrics to enhance timely and efficient detection in various applications.**

Keywords— Concept drift, data stream, non-stationary environments, multi-criteria Decision-Making (MCDM), big data applications

I. INTRODUCTION

Classifying data streams is a complex task due to three primary characteristics: speed, size, and variability [1]. Speed and size are particularly challenging because they impose constraints on memory and processing time, requiring learning algorithms to temporarily store incoming data and process it only once. The most critical challenge, however, is variability, which refers to the dynamic nature of data streams. This variability often leads to what is known as concept drift [2], [3]. Concept drift occurs when the class labels of a dataset change over time, causing the underlying distribution at one time point (D_i) to differ from another (D_j). As a result, the concepts associated with these points become unstable, and the model struggles to accurately predict the most recent data distribution. Consequently, a key task in streaming data analytics is detecting significant changes in the incoming data [2], [4].

A real-world example of concept drift can be observed in customer behaviour within an online shop, where customer preferences evolve over time. For instance, a predictive model designed to forecast weekly sales might initially perform well. However, factors such as promotional activities and advertising expenditure, which influence sales, could lead to a gradual decrease in the model's accuracy—signalling the occurrence of concept drift. Additionally, seasonal variations in sales can contribute to concept drift, as shopping patterns shift, for example, with higher sales during the winter holidays compared to the summer. Moreover, any predictive model created before the COVID-19 pandemic that assumes a fixed relationship between inputs and outputs would likely perform poorly due to changes in underlying data patterns.

In the context of mining data streams affected by concept drift, approaches are generally classified as either active (trigger-based) or passive (evolving) [1], [5]. Active approaches focus on detecting concept drift using various detectors and updating the model when drift is detected. In contrast, passive approaches continuously update the model as new data arrives, regardless of whether drift is occurring. Despite their different methods, both approaches aim to keep the model current and accurate.

Current drift detection methods [6], [7], [8], [9], [10], [11] employ various performance metrics to evaluate and compare the effectiveness of different detectors, including accuracy, runtime, and false alarm rate [2], [5]. For example, certain detectors may achieve high accuracy but require substantial memory, while others may detect drift effectively but have long execution times. This scenario raises a critical question: which performance metric should be prioritized to determine the superiority of one detector over another? Should the evaluation be based on accuracy, runtime, false alarm rate, or another metric? Thus, this study aims to address the following research question:

- **Research Question**: How can the performance of concept drift detection algorithms be comprehensively evaluated using multiple performance metrics to identify the most effective methods for timely and efficient detection in online data streams?

To address the research question, this study proposes an approach that comprehensively considers all performance metrics simultaneously to evaluate the performance of drift detectors. A rigorous empirical evaluation will be undertaken, comparing eleven of the existing drift detectors using four synthetic data streams and seven performance metrics, namely: Average Delay Detection (ADD), Average True Detection (ATD), Average False Alarm (AFA), Average False Negative (AFN), Average Detection Runtime in milliseconds (ARMS), Average Memory Usage in bytes (MUB), and Average Accuracy. Each metric will be transformed and weighted, then summed to create a single score for each method. This score will represent the balance among all metrics. We can then visualize this score for each approach using a simple bar chart.

The structure of this paper is as follows: Section 2 provides a review of relevant literature. Section 3 details the proposed framework for multi-label classification. The results of our experimental evaluation, along with comparisons to existing studies, are discussed in Section 4. Lastly, Section 5

presents our conclusions and suggests avenues for future research.

II. Literature Review

Concept drift detection has been a pivotal area of research in the field of data stream mining. Numerous algorithms have been developed to tackle the challenges posed by evolving data streams. This section reviews prominent drift detectors from past studies, highlighting their methodologies and effectiveness.

A. Drift Detection Method (DDM)

The Drift Detection Method (DDM) [6] is a widely recognized technique that relies on monitoring the error rate of a classifier. DDM sets a cautionary threshold based on classifier error; if this threshold is exceeded, the method isolates incoming samples in a specific window. Should the error rate reach a predetermined drift threshold, the classifier is reconfigured using the samples from this window. This method effectively detects sudden drifts by focusing on abrupt changes in error rates.

B. Diversity Measure as a New Drift Detection Method (DMDDM)

The Diversity Measure as a New Drift Detection Method (DMDDM) [12] is a novel approach designed to detect concept drift in data streams efficiently. Unlike traditional methods that monitor error rates, DMDDM uses the disagreement measure between pairs of classifiers combined with the Page-Hinkley test to detect drifts quickly, with minimal memory and processing time. The method is particularly effective in handling sudden or abrupt drifts in binary classification problems. Experimental results demonstrate that DMDDM outperforms several existing methods in terms of detection speed, runtime efficiency, and memory usage.

C. Adaptive Sliding Window (ADWIN)

The ADWIN, or Adaptive Sliding Window approach [13], uses two sub-windows within a shifting window to detect concept drift. If a significant difference in the averages of these sub-windows is observed, ADWIN flags a concept drift and removes elements from the window's end until the significant difference disappears. This method is notable for its ability to adjust the window size dynamically, making it sensitive to changes in the data distribution.

D. Fast Hoeffding Drift Detection Method (FHDDM)

FHDDM leverages Hoeffding's inequality to detect drifts within a specified window size[8]. It identifies drift by monitoring significant changes between current probabilities and the peak of accurate forecasts. FHDDM is effective in identifying drifts by statistically bounding the probability of changes, making it reliable for quick detection.

E. McDiarmid Drift Detection Methods (MDDM)

MDDM-A, MDDM-G, and MDDM-E [10] utilize McDiarmid's inequality to detect drifts. These methods apply a fixed-size window over predictive outcomes, assigning a value of 1 for accurate predictions and 0 otherwise. The weighted average and peak weighted average within this window are calculated, with significant discrepancies indicating drift. These variations of MDDM are adept at capturing different types of drift, from abrupt to gradual.

F. PH Test

The PH Test, often used in signal processing, calculates the cumulative discrepancy between observed values and their average up to the present [14]. It identifies drift by noting significant discrepancies between cumulative discrepancies and their minimum values over time. This method is versatile and effective in various applications, particularly in detecting both abrupt and gradual drifts.

G. Hoeffding's Bounds Drift Detection (HDDM)

HDDM_A and HDDM_W test [15] use Hoeffding's bounds for drift detection, with HDDM_A comparing moving averages and HDDM_W examining weighted averages. The weighting process uses an Exponentially Weighted Moving Average (EWMA) forgetting scheme [16] , which adapts to immediate and gradual shifts respectively. These methods provide flexibility in handling different drift rates and types.

H. Segmented Drift Detection (SegDrift2)

SegDrift2[11]utilizes two storage mechanisms for incoming data, one combining new and old data, and the other exclusively housing new entries. By comparing the mean values of these repositories, SegDrift2 detects drift when a predefined threshold is exceeded. This method is effective in distinguishing between short-term fluctuations and long-term drifts.

III. Methodology

The primary goal of this study is to benchmark eleven leading concept drift detection algorithms using a Multi-Criteria Decision-Making (MCDM) approach [17], [18], [19]. This section outlines the methodology used for the evaluation, including the datasets, performance metrics, and the MCDM framework.

In general, consider an MCDA problem involving m alternatives and n decision criteria. Assume that all criteria are benefit criteria, meaning that higher values are preferable. Let w_j represent the relative importance weight of criterion C_j and let a_{ij} indicate the performance value of alternative A_i with respect to criterion C_j. The overall importance of alternative A_i, referred to as the $A_i^{\text{WSM-score}}$, when all criteria are considered simultaneously, is defined as follows:

$$A_i^{\text{WSM}-\text{score}} = \sum_{j=1}^{n} \omega_j a_{i,j}, \quad for\ i = 1,2,3,\dots,m.$$

Consider a straightforward numerical example where a decision problem involves three alternative options, A, B, and C, each evaluated based on four criteria: C1, C2, C3, C4. The corresponding numerical data for this problem is presented in the following decision matrix:

Table1: Decision Matrix

	Criteria				WSM SCORE
	C1	C2	C3	C4	
Weighting	0.40	0.30	0.20	0.10	-
Supplier A:	8	7	9	6	7.7
Supplier B:	7	8	8	7	7.5
Supplier C:	6	9	7	8	7.3

An organization needs to select a supplier based on four criteria: cost, quality, delivery time, and service. The weights assigned to these criteria, based on their relative importance, are 0.40, 0.30, 0.20, and 0.10, respectively. The organization

evaluates three suppliers (A, B, and C) and assigns scores to each based on the criteria (on a scale of 1 to 10):

The weighted scores are calculated as follows:

- **Supplier A:** (80.40) + (70.30) + (90.20) + (60.10) = 3.2 + 2.1 + 1.8 + 0.6 = 7.7
- **Supplier B:** (70.40) + (80.30) + (80.20) + (70.10) = 2.8 + 2.4 + 1.6 + 0.7 = 7.5
- **Supplier C:** (60.40) + (90.30) + (70.20) + (80.10) = 2.4 + 2.7 + 1.4 + 0.8 = 7.3

Based on the WSM, Supplier A, with the highest total score of 7.7, is deemed the most suitable choice for the organization. This example elucidates the practical application of the Weighted Sum Model, demonstrating its simplicity and effectiveness in facilitating well-informed and objective decision-making.

A. Multi-Criteria Decision-Making (MCDM) Framework

To identify the best-performing algorithm, we utilized a Multi-Criteria Decision-Making (MCDM) framework. MCDM is a powerful tool for evaluating multiple competing criteria, providing a balanced assessment across different performance aspects.

The overall process consists of the following steps:

1. **Criteria Identification**: Identify all relevant criteria that influence the decision-making process.
2. **Weight Assignment**: Assign a weight to each criterion to reflect its significance in the overall decision.
3. **Scoring Alternatives**: Evaluate each alternative by scoring them against all identified criteria based on their performance.
4. **Weighted Sum Calculation**: Calculate the total score for each alternative by multiplying each criterion's score by its assigned weight and summing the results.
5. **Decision Making**: The alternative with the highest total score is selected as the optimal choice.

The weighting process is also a crucial element of the WSM, as it directly impacts the decision outcome. The steps involved in this process are as follows:

- **Determining Importance**: Establish the relative importance of each criterion. This can be achieved using methods such as expert judgment, surveys, or statistical techniques.
- **Normalization**: Normalize the weights so that they sum to one, ensuring consistency and comparability across criteria.
- **Assigning Weights**: Based on their determined importance, assign a numerical weight to each criterion. For example, if the relative importance of four criteria is assessed as 30%, 25%, 20%, and 25%, the corresponding weights would be 0.30, 0.25, 0.20, and 0.25, respectively.
- **Consistency Check**: Conduct a consistency check on the assigned weights to avoid biases or errors that could distort the final decision.

A. Datasets

Four synthetic datasets were selected and used to evaluate the eleven drift detection methods. These datasets SEA, Sine1, Mixed, and AGRAWAL were created using the MOA (Massive Online Analysis) tool [20]. The primary benefit of using synthetic datasets is their ability to accurately identify the true location of drifts within a data stream. The specifications of these datasets are:

1. **Mixed (with abrupt concept drift):** this dataset includes two numerical variables, *x* and *y*, ranging from 0 to 1, and two boolean attributes, *v* and *w*. A data point is labeled positive if at least two of these conditions are met: *v*, *w*, or $y < 0.5 + 0.3 * \sin(3\pi x)$. When drift occurs, the classification rules are reversed.
2. **Sine1 (with abrupt concept drift)**: This dataset includes two features, *x* and *y*, evenly distributed between 0 and 1. Classification is based on y = sin(x), with points below the curve labeled positive and those above negative. When drift happens, class labels are reversed.
3. **AGRAWAL Generator (AGR)**: we use the AGR generator to generate 100,000 instances with multiple sudden drifts. We use the AGR generator to generate three drifts every 25,000 instances.
4. **SEA:** was used to create 100,000 data instances, simulating an abrupt concept drift. The *ConceptDriftStream* class manages the drift, with *SEAGenerator* -f 3 representing the current concept and *SEAGenerator* -f 2 representing the new concept. The drift occurs at position 10k within the data, with a specified width.

B. Performance Metrics

We employed Seven performance metrics to provide a comprehensive evaluation of the concept drift detection algorithms:

Average Delay Detection (ADD): Measures the time taken by the algorithm to detect a concept drift after it occurs.
Average True Detection (ATD): Quantifies the rate at which true drifts are correctly identified by the algorithm.
Average False Alarm (AFA): Counts the number of false alarms raised by the algorithm when no drift has occurred.
Average False Negative (AFN): Measures the rate at which the algorithm fails to detect actual drifts.
Average Detection Runtime in milliseconds (ARMS): Evaluates the computational efficiency of the algorithm in terms of detection time.
Average Memory Usage in bytes (MUB): Assesses the memory consumption of the algorithm during the detection process.
Average Accuracy: Measures the percentage of correct predictions overall.

C. Experimental Setup

All algorithms were implemented in Java using the MOA framework [20]. The experiments were run on a machine equipped with an Intel Core i7 processor @ 3.4 GHz, 16 GB of RAM, and Windows 10. To ensure a fair and meaningful comparison, identical parameter values were used for all algorithms.

IV. Results and Analysis

This section presents the results of the evaluation of the eleven concept drift detection algorithms across the four datasets (Mixed, Sine1, AGR, and SEA). The results are analysed based on the evaluation metrics discussed in the Methodology section, and the implications of these findings are discussed in detail. The evaluation metrics, including Average Delay Detection (ADD), Average True Detection (ATD), Average False Alarm (AFA), Average False Negative (AFN), Average Detection Runtime in milliseconds (ARMS), Average Memory Usage in bytes (MUB), and average accuracy, provide a comprehensive view of the performance of each algorithm. The scores for each algorithm across all datasets are summarized in Figures 1-4.

Figures 1-4 illustrate the balanced score of each drift detection method based on seven key metrics. The performance variations among the detection methods across the datasets are clearly visible. The length of each bar represents the score, with longer bars indicating a more balanced performance across all metrics. Methods at the top of the chart exhibit the best-balanced scores. This visualization allows us to easily determine which method achieves the most optimal balance across the four metrics, with the method having the longest bar demonstrating the best overall balance.

A. *Performance Overview*

The Weighted Sum Model results for the Mixed and Sine1 datasets provide insightful evaluations of various drift detection methods across critical performance metrics. In the Mixed Dataset, DMDMM and SeqDrift emerge as the leading methods, demonstrating their ability to effectively balance rapid drift detection, efficient processing time, and accuracy. These methods' consistent top scores make them strong candidates for scenarios requiring a well-rounded approach to concept drift management. In contrast, the Sine1 Dataset presents a slightly more competitive environment. While DMDMM and SeqDrift maintain their high performance, ADWIN also displays notable effectiveness, resulting in a more diverse distribution of scores. Nevertheless, DMDMM's and SeqDrift's consistent superiority across both datasets highlights their versatility and reliability. These methods' ability to consistently perform well, regardless of the dataset's characteristics, underscores their robustness and suitability for a wide range of drift detection applications.

Similarly, the AGR and SEA datasets showcase the adaptability of different drift detection methods under varying data conditions. In the AGR Dataset, ADWIN takes the lead, demonstrating its proficiency in balancing detection speed, memory usage, and accuracy, making it particularly suited for data scenarios with similar characteristics. SeqDrift and DMDMM follow closely, reaffirming their strong performance in different contexts. Meanwhile, in the SEA Dataset, DMDMM and SeqDrift once again dominate, reinforcing their status as top-tier methods. ADWIN also remains a strong contender, underscoring its consistency across datasets. This analysis reveals that while certain methods excel in specific datasets, DMDMM and SeqDrift consistently prove to be reliable choices across various scenarios, offering a robust solution for diverse drift detection challenges. Their balanced performance across the AGR and SEA datasets exemplifies their capability to adapt to different data dynamics effectively.

Fig. 1. Balanced scores for each drift detection method based on seven metrics using the Mixed dataset.

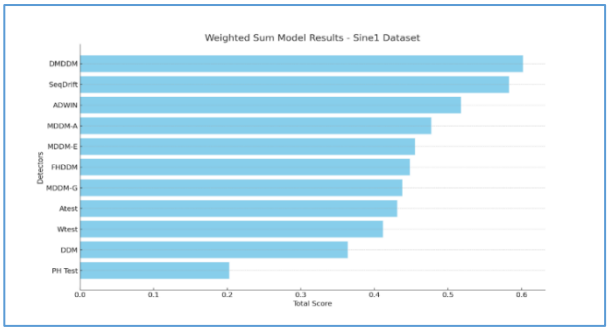


Fig. 2. Balanced scores for each drift detection method based on seven metrics using the Sine1 dataset.

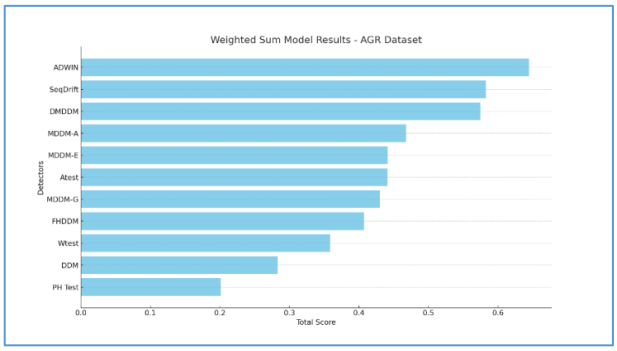


Fig. 3. Balanced scores for each drift detection method based on five metrics using the AGR dataset.

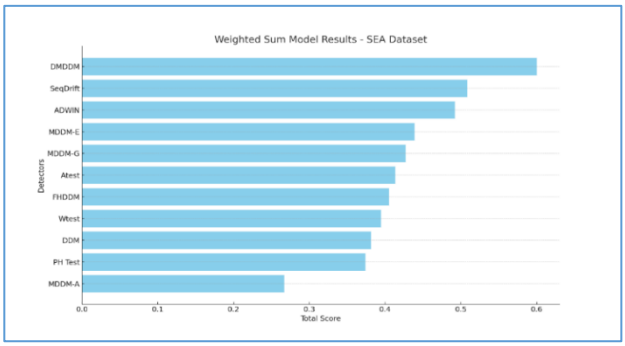


Fig. 4. Balanced scores for each drift detection method based on seven metrics using the SEA dataset.

B. *Detailed Analysis*

Figures 5-8 illustrate the performance of eleven concept drift detection algorithms across three different datasets. Each figure compares the algorithms based on key metrics, including ADD, ATD, AFA, AFN, DRMS, MUB, and Average Accuracy.

1) Average Delay Detection (ADD)

DMDDM demonstrates consistently low ADD across all datasets, making it one of the fastest detectors. In contrast, PH Test and SeqDrift have significantly higher ADD, particularly in the SEA and Mixed datasets, which implies slower response times. DDM also shows high ADD in some datasets, indicating delayed detection, especially in the SEA and AGR datasets.

2) Average True Detection (ATD)

ATD is generally consistent across all detectors, with most showing stable performance. DDM tends to have lower ATD in the Sine1 and SEA datasets, suggesting it may struggle with certain data types. DMDDM and FHDDM maintain high ATD across all datasets, indicating reliable detection accuracy, while SeqDrift and PH Test show variability.

3) Average False Alarm (AFA)

AFA is low across most detectors, indicating a minimal occurrence of false alarms. SeqDrift consistently shows zero false alarms, which is ideal. However, DDM and PH Test exhibit slightly higher AFA in some datasets, particularly in Sine1 and SEA, indicating more frequent false positives, which could affect overall accuracy and reliability.

4) and Average False Negative (AFN)

AFN is generally low across all detectors, suggesting effective detection capabilities. However, DDM and PH Test show higher AFN in the Sine1 and SEA datasets, indicating a higher rate of missed detections. DMDDM and Wtest maintain low AFN across all datasets, highlighting their reliability in detecting true positives with minimal false negatives.

5) Average Detection Runtime (DRMS)

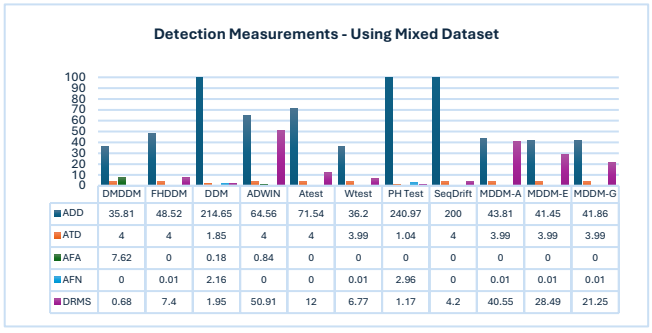
SeqDrift exhibits the highest DRMS across all datasets, indicating it is less efficient and has a longer detection runtime. DMDDM and Wtest consistently show lower DRMS, reflecting their efficiency in processing data quickly. ADWIN struggles with higher DRMS in the Mixed and AGR datasets, while PH Test also shows elevated DRMS in most datasets.

6) Average Memory Usage (MUB)

SeqDrift has the highest average memory usage across all datasets, particularly in the AGR and SEA datasets, indicating heavy resource consumption. ADWIN also shows significant memory usage, especially in the Mixed and SEA datasets. DMDDM, FHDDM, and DDM consistently have the lowest memory usage, indicating they are more resource-efficient across various datasets.

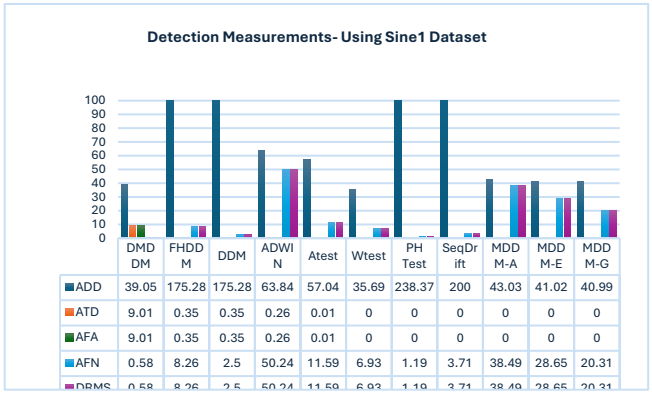
7) Average Accuracy

DMDDM, FHDDM, and MDDM-G achieve consistently high accuracy across all datasets, particularly in the SEA dataset, where they peak at around 89%. PH Test and DDM exhibit lower accuracy, especially in the AGR dataset. SeqDrift shows varying accuracy, with lower performance in the AGR dataset, indicating inconsistent detection reliability across different datasets.



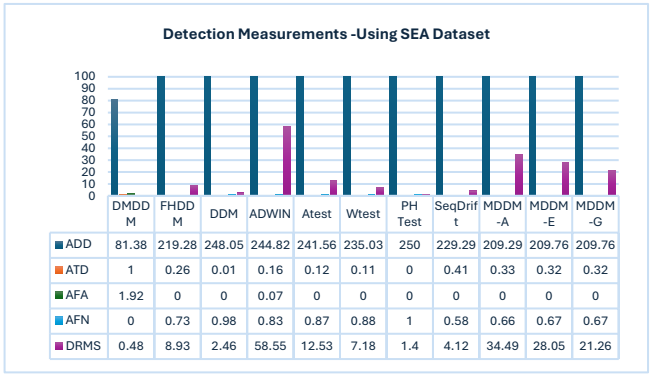
	DMDDM	FHDDM	DDM	ADWIN	Atest	Wtest	PH Test	SeqDrift	MDDM-A	MDDM-E	MDDM-G
ADD	35.81	48.52	214.65	64.56	71.54	36.2	240.97	200	43.81	41.45	41.86
ATD	4	4	1.85	4	4	3.99	1.04	4	3.99	3.99	3.99
AFA	7.62	0	0.18	0.84	0	0	0	0	0	0	0
AFN	0	0.01	2.16	0	0	0.01	2.96	0	0.01	0.01	0.01
DRMS	0.68	7.4	1.95	50.91	12	6.77	1.17	4.2	40.55	28.49	21.25

Fig. 5. Comparison of Detection Measurements across Multiple Methods using Mixed Dataset



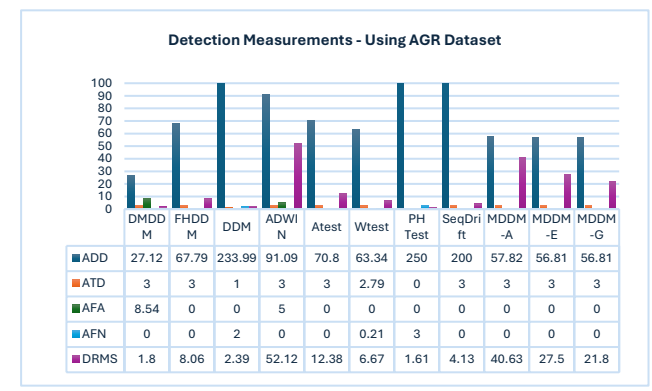
	DMDDM	FHDDM	DDM	ADWIN	Atest	Wtest	PH Test	SeqDrift	MDDM-A	MDDM-E	MDDM-G
ADD	39.05	175.28	175.28	63.84	57.04	35.69	238.37	200	43.03	41.02	40.99
ATD	9.01	0.35	0.35	0.26	0.01	0	0	0	0	0	0
AFA	9.01	0.35	0.35	0.26	0.01	0	0	0	0	0	0
AFN	0.58	8.26	2.5	50.24	11.59	6.93	1.19	3.71	38.49	28.65	20.31
DRMS	0.58	8.26	2.5	50.24	11.59	6.93	1.19	3.71	38.49	28.65	20.31

Fig. 6. Comparison of Detection Measurements across Multiple Methods using Sine1 Dataset



	DMDDM	FHDDM	DDM	ADWIN	Atest	Wtest	PH Test	SeqDrift	MDDM-A	MDDM-E	MDDM-G
ADD	81.38	219.28	248.05	244.82	241.56	235.03	250	229.29	209.29	209.76	209.76
ATD	1	0.26	0.01	0.16	0.12	0.11	0	0.41	0.33	0.32	0.32
AFA	1.92	0	0	0.07	0	0	0	0	0	0	0
AFN	0	0.73	0.98	0.83	0.87	0.88	1	0.58	0.66	0.67	0.67
DRMS	0.48	8.93	2.46	58.55	12.53	7.18	1.4	4.12	34.49	28.05	21.26

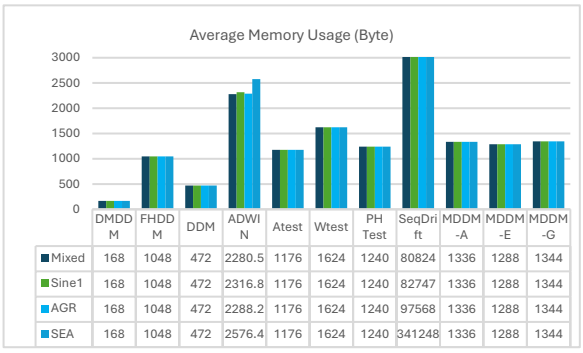
Fig. 7. Comparison of Detection Measurements across Multiple Methods using SEA Dataset



	DMDDM	FHDDM	DDM	ADWIN	Atest	Wtest	PH Test	SeqDrift	MDDM-A	MDDM-E	MDDM-G
ADD	27.12	67.79	233.99	91.09	70.8	63.34	250	200	57.82	56.81	56.81
ATD	3	3	1	3	3	2.79	0	3	3	3	3
AFA	8.54	0	0	5	0	0	0	0	0	0	0
AFN	0	0	2	0	0	0.21	3	0	0	0	0
DRMS	1.8	8.06	2.39	52.12	12.38	6.67	1.61	4.13	40.63	27.5	21.8

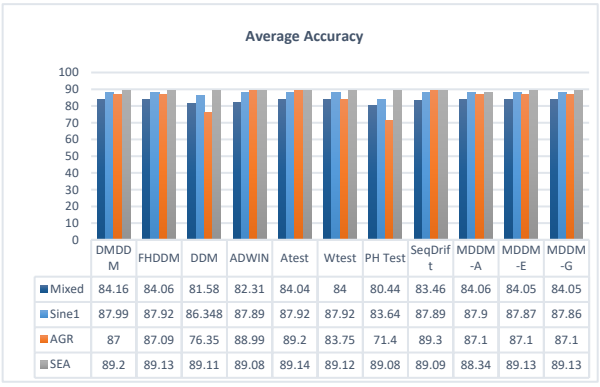
Fig. 8. Comparison of Detection Measurements across Multiple Methods using AGR Dataset

C. Discussion



	DMDDM	FHDDM	DDM	ADWIN	Atest	Wtest	PH Test	SeqDrift	MDDM-A	MDDM-E	MDDM-G
Mixed	168	1048	472	2280.5	1176	1624	1240	80824	1336	1288	1344
Sine1	168	1048	472	2316.8	1176	1624	1240	82747	1336	1288	1344
AGR	168	1048	472	2288.2	1176	1624	1240	97568	1336	1288	1344
SEA	168	1048	472	2576.4	1176	1624	1240	341248	1336	1288	1344

Fig. 9. Comparison of Memory Usage across Multiple Methods using all datasets.



	DMDDM	FHDDM	DDM	ADWIN	Atest	Wtest	PH Test	SeqDrift	MDDM-A	MDDM-E	MDDM-G
Mixed	84.16	84.06	81.58	82.31	84.04	84	80.44	83.46	84.06	84.05	84.05
Sine1	87.99	87.92	86.348	87.89	87.92	87.92	83.64	87.89	87.9	87.87	87.86
AGR	87	87.09	76.35	88.99	89.2	83.75	71.4	89.3	87.1	87.1	87.1
SEA	89.2	89.13	89.11	89.08	89.14	89.12	89.08	89.09	88.34	89.13	89.13

Fig. 10. Comparison of Average Accuracy across Multiple Methods using all datasets.

The results indicate that DMDDM is the best-performing algorithm overall, excelling in both detection accuracy and computational efficiency. Its strong performance across most datasets and metrics highlights its robustness and suitability for a wide range of applications.

However, the study also reveals that no single algorithm dominates all criteria. For example, while DMDDM is highly efficient, SeqDrift's high memory usage might be a concern in resource-limited environments. Similarly, PH Test's reliability in minimizing false alarms makes it an excellent choice for applications where accuracy is paramount, despite its slower detection speed.

These findings underscore the importance of selecting an algorithm based on the specific requirements of the application context. The Weighted Sum Model (WSM) proves to be an effective tool for balancing multiple criteria and guiding the selection of the most appropriate algorithm.

V. Conclusion

In this study, we performed a comprehensive evaluation of eleven leading concept drift detection algorithms using a Multi-Criteria Decision-Making approach. By leveraging synthetic datasets, we provided an in-depth analysis of each algorithm's performance under varying conditions of concept drift. The results of our study demonstrate that no single algorithm excels in all criteria, highlighting the importance of selecting drift detection methods based on the specific requirements of a given application. For instance, some algorithms showed superior performance in terms of detection accuracy but at the cost of higher computational overhead, while others were more efficient but less accurate in detecting true drifts.

Our proposed MCDM framework allows for a balanced assessment of drift detection algorithms, offering a nuanced understanding of their strengths and weaknesses. This approach is particularly beneficial for applications in which multiple performance criteria are critical, ensuring that the chosen algorithm aligns well with the overall system requirements. Future work could expand upon this research by incorporating additional datasets, exploring the impact of different weighting schemes in the MCDM process, and developing more sophisticated techniques for handling diverse types of concept drift. Moreover, integrating adaptive mechanisms into the MCDM framework could further enhance its applicability in real-time data stream environments.

References

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM computing surveys (CSUR), vol. 46, no. 4, pp. 1–37, 2014.

[2] O. A. Mahdi, N. Ali, E. Pardede, A. Alazab, T. Al-Quraishi, and B. Das, "Roadmap of Concept Drift Adaptation in Data Stream Mining, Years Later," *IEEE Access*, vol. 12, pp. 21129–21146, 2024, doi: 10.1109/ACCESS.2024.3358817.

[3] O. A. Mahdi, "Diversity Measures as New Concept Drift Detection Methods in Data Stream Mining," La Trobe University Melbourne, Australia 9, 2020.

[4] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Mach Learn*, vol. 98, pp. 455–482, 2015.

[5] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput Intell Mag*, vol. 10, no. 4, pp. 12–25, 2015.

[6] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*, 2004, pp. 286–295.

[7] O. A. Mahdi, E. Pardede, and N. Ali, "A hybrid block-based ensemble framework for the multi-class problem to react to different types of drifts," *Cluster Comput*, vol. 24, pp. 2327–2340, 2021.

[8] A. Pesaranghader and H. L. Viktor, "Fast hoeffding drift detection method for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*, 2016, pp. 96–111.

[9] O. A. Mahdi, N. Ali, E. Pardede, and T. Al-Quraishi, "Online Concept Drift Detector: Optimally Balancing Delay Detection, Runtime, Memory, and Accuracy.," *Procedia Comput Sci*, vol. 237, pp. 559–567, 2024.

[10] A. Pesaranghader, H. L. Viktor, and E. Paquet, "McDiarmid drift detection methods for evolving data streams," in *2018 International joint conference on neural networks (IJCNN)*, 2018, pp. 1–9.

[11] R. Pears, S. Sakthithasan, and Y. S. Koh, "Detecting concept change in dynamic data streams: A sequential approach based on reservoir sampling," *Mach Learn*, vol. 97, pp. 259–293, 2014.

[12] O. A. Mahdi, E. Pardede, N. Ali, and J. Cao, "Diversity measure as a new drift detection method in data streaming," *Knowl Based Syst*, vol. 191, p. 105227, 2020.

[13] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*, 2007, pp. 443–448.

[14] J. Gama, R. Sebastiao, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach Learn*, vol. 90, pp. 317–346, 2013.

[15] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-D\'\iaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on Hoeffding's bounds," *IEEE Trans Knowl Data Eng*, vol. 27, no. 3, pp. 810–823, 2014.

[16] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, “Exponentially weighted moving average charts for detecting concept drift,” *Pattern Recognit Lett*, vol. 33, no. 2, pp. 191–198, 2012.

[17] A. M. Alshamsi, H. El-Kassabi, M. A. Serhani, and C. Bouhaddioui, “A multi-criteria decision-making (MCDM) approach for data-driven distance learning recommendations,” *Educ Inf Technol (Dordr)*, pp. 1–38, 2023.

[18] W. Ma, Y. Du, X. Liu, and Y. Shen, “Literature review: Multi-criteria decision-making method application for sustainable deep-sea mining transport plans,” *Ecol Indic*, vol. 140, p. 109049, 2022.

[19] S. Chakraborty, R. D. Raut, T. M. Rofin, and S. Chakraborty, “A comprehensive and systematic review of multi-criteria decision-making methods and applications in healthcare,” *Healthcare Analytics*, p. 100232, 2023.

[20] A. Bifet *et al.*, “Moa: Massive online analysis, a framework for stream classification and clustering,” in *Proceedings of the first workshop on applications of pattern analysis*, 2010, pp. 44–50.