Received: 17 July 2025 | Revised: 2 December 2025 | Accepted: 6 February 2026 | Published online: 11 September 2026

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2026, Vol. 00(00) 1–20
DOI: 10.47852/bonviewJCCE62026839

BON VIEW PUBLISHING

An Online Clustering Algorithm for Handling Evolving Data Streams with the Ability to Prevent Clusters' False Merging Using Adaptive Time Interval

Redhwan Al-amri[1,*], Gamal Alkawsi[2], Abdulnaser A. Hagar[3] and Luiz Fernando Capretz[4]

[1] *Institute of Inner City Learning London, University of Wales Trinity Saint David, UK*

[2] *Institute of Informatics and Computing in Energy, Universiti Tenaga Nasional (Putrajaya Campus), Malaysia*

[3] *Faculty of Data Science and Information Technology, INTI International University, Malaysia*

[4] *Department of Electrical and Computer Engineering, Western University, Canada*

Abstract: Online clustering of evolving data streams presents unique challenges, particularly the problem of false merging, where distinct clusters are incorrectly combined during temporal overlap. Existing buffer-based and density-based algorithms, such as BOCEDS and CEC-Merge, lack mechanisms to adaptively distinguish transient overlaps from genuine merging events. To address this, we propose BOCEDS-ATI—a Buffer-based Online Clustering for Evolving Data Streams with Adaptive Time Interval algorithm. The core innovation is the adaptive time interval mechanism, which dynamically adjusts merging decisions based on the temporal persistence and relative velocity of cluster interactions. This allows the model to recognize and prevent false merging in continuously evolving data streams. Experimental evaluation on multiple synthetic and real-world datasets demonstrates that BOCEDS-ATI achieves superior clustering accuracy and robustness to dynamic drift compared with state-of-the-art algorithms, while maintaining linear time complexity suitable for real-time applications. The results confirm that incorporating localized temporal adaptation significantly improves the reliability of online clustering in non-stationary environments.

Keywords: adaptive time interval, clustering, data stream, evolving, false merging

1. Introduction

The rapid growth of real-time data from Internet of Things (IoT) devices, sensors, and digital platforms has intensified the need for online clustering algorithms capable of processing data streams that evolve continuously over time. This exponential growth can be attributed to various factors, including the widespread adoption of the IoT and the proliferation of diverse applications across multiple domains [1]. The IoT, with its interconnected network of devices, sensors, and systems, contributes significantly to the massive influx of data [2]. These interconnected devices generate data at an unprecedented scale, capturing valuable information about the physical world and human activities [3]. Additionally, the expanding landscape of applications, spanning from embedded systems in smart devices to social media analytics, plays a crucial role in driving the data explosion [4]. These data hold immense potential for insights, innovation, and informed decision-making across numerous industries and sectors [5]. As the digital ecosystem continues to evolve, the challenges and opportunities associated with handling and extracting value from these data will continue to shape the future of technology and drive advancements in data management, analytics, and artificial intelligence [6]. This continuous and unbounded arrival of data points is commonly known as data streams [7]. Data streams are a continuous and limitless sequence of data points that arrive in a constant stream [8]. They are often associated with embedded or precise timestamps and are generated at a rapid pace from a wide range of heterogeneous sources.

In the last decade, the volume of data has increased rapidly. Data have different sources and sizes according to the industry. For example, in e-commerce, data are generated in big amounts [5]. The existence of big data makes the problem of analyzing and knowledge extraction more challenging [9]. Data stream clustering plays a significant role in identifying anomalies in many application domains. In nearly every scenario, data clustering is incredibly beneficial [10]. Data analysts are continuously

***Corresponding author:** Redhwan Al-amri, Institute of Inner City Learning London, University of Wales Trinity Saint David, UK. Email: r.al-amri@uwtsd.ac.uk

looking for techniques that might retrieve hidden information from data streams. Scholars utilize the obtained information to tackle social issues that lead to more pleasant living and a better planet for humans. Some instances of such initiatives include forecasts of societal unrest using social media data [11], early health issue diagnosis [12], people demand analysis for new product development [13], and finding weather regimes [14]. Analyzing patterns is defined based on their characteristic. One significant aspect in the field of online clustering is the ability to adapt to the dynamic nature of data streams while maintaining the integrity of clusters. This is particularly important to prevent clusters' false merging, where distinct clusters are mistakenly combined due to overlapping or proximity [15]. False merging can lead to inaccurate analysis and misinterpretation of the underlying data structure. Therefore, the development of online clustering algorithms that can effectively handle evolving data streams and prevent false merging is crucial [16].

There has been a recent proliferation of proposed clustering algorithms specifically designed for data streams. Yet, there is a lack of studies handling the issue of overlapping clusters and false merging [15, 17]. This issue poses a major challenge in terms of the accuracy of existing data stream clustering algorithms, especially when considering evolving data streams. Hence, this paper aims to solve the issue by developing an algorithm for preventing the false merging of overlapped moving clusters using an adaptive time interval called ATI. The primary objective of the adaptive time interval (ATI) technique is to ensure the production of high-quality clusters, thereby mitigating the occurrence of false merging that may arise when two or more clusters overlap. It is an online clustering technique, and it applies the Euclidean distance function for the velocity calculation of binary class data. This works by selecting a reference point and then calculating the relative speed with respect to the speed threshold. If the relative speed exceeds the speed threshold, split the clusters into two clusters; otherwise, merge the clusters.

This research introduces a novel online clustering algorithm, named BOCEDS-ATI (Buffer-based Online Clustering for Evolving Data Streams with Adaptive Time Interval), designed to address the persistent issue of false merging in evolving data streams. The key contributions of this paper are summarized as follows:

1) Adaptive time interval (ATI) mechanism: We propose a new time-adaptive decision mechanism that dynamically evaluates the temporal persistence of cluster proximity. Unlike existing fixed-interval approaches such as CEC-Merge or the nontemporal BOCEDS framework, ATI continuously adjusts the merging threshold based on the relative velocity and temporal consistency between clusters. This allows the model to distinguish between transient overlaps and genuine cluster convergence.
2) Localized temporal adaptation: While prior adaptive-window methods rely on global time windows, ATI introduces localized link timers at the cluster-pair level, offering fine-grained temporal awareness without requiring large memory buffers or historical data stores. This makes BOCEDS-ATI computationally efficient and scalable for high-velocity streams.
3) Improved handling of evolving clusters: The integration of velocity-sensitive decision rules and adaptive link intervals enables the algorithm to maintain cluster integrity during dynamic evolution, minimizing false merges and improving accuracy across multiple real-world data streams.
4) Comprehensive empirical evaluation: The algorithm is benchmarked against several state-of-the-art stream clustering methods, demonstrating superior performance in dynamic environments while maintaining computational efficiency.

Overall, this study contributes a spatiotemporal online clustering framework that bridges density-based clustering and adaptive temporal reasoning, thereby extending the capability of online clustering models to handle complex and continuously evolving data.

The subsequent sections of this paper are structured as follows to provide a comprehensive understanding of the research conducted. In Section 2, a motivation section highlights the significance of this research. In Section 3, a thorough review of the existing literature related to the topic is presented. This review encompasses key studies, theories, and methodologies that have been previously explored by researchers in the field, followed by a research background, which is presented in Section 4. Section 5 is dedicated to presenting the methodology employed in this research. It outlines the specific approach adopted and highlights the development of the algorithm. A detailed illustration of the algorithm is provided, elucidating its core components and functionalities. The results obtained from the research and their subsequent analysis are presented and discussed in Section 6. This section offers an in-depth examination and interpretation of the findings, drawing meaningful insights and correlations from the collected data. Section 7 focuses on the study's key findings and provides a concise summary of the research. The implications, significance, and potential applications of the findings are discussed, giving readers a clear understanding of the study's contributions. Finally, Section 8 presents the concluding remarks of the paper. It provides a succinct summary of the main points discussed throughout the paper, emphasizes the significance of the research, and offers suggestions for future work and areas of further investigation.

2. Motivation

Data stream clustering is essential to handle the analysis of the data generated from IoT systems. Considering that IoT devices are predicted to cross over 75 billion devices by 2025, which is over triple the global population [18]. This shows the importance of having effective, reliable, and efficient data stream clustering. In real-world applications:

1) Generally, naturally formed clusters are not spherical in shape. For example, in habitat monitoring applications utilizing wireless sensor networks, the sensors generate a continuous stream of geographic data, including the locations of monitored objects. Clusters can take on any shape in these applications due to the constraints imposed by geographic elements such as mountains and rivers.
2) In some applications, there is a significant quantity of noise or outliers; for example, due to the influence of various causes such as transitory sensor failure in a data stream situation, some random sounds arise on occasion. Detecting noise is a critical challenge, especially in an evolving data stream, where the importance of real data varies over time.
3) Overlapping clusters and false merging arise as a challenge when multiple clusters overlap, leading to inaccurate merging of distinct clusters.
4) Numerous real-world datasets lack a priori knowledge that may be used to measure the number of clusters.

To address the challenges, researchers have developed density-based clustering techniques. Yet, the existing density-based techniques have the following limitations:

Density-based clustering techniques require a long computation time due to the time-consuming process of determining the nearest neighbors to build clusters. Recently, several density-based clustering algorithms for data stream clustering have been proposed. Finding neighbors to construct clusters is conducted on summarized data in data stream contexts. While several algorithms attempt to lower the number of comparisons to decrease the computational time, the number of comparisons remains too high to be applied to data streams [19]. Moreover, density-based clustering algorithms are degraded when handling evolving data due to various causes such as parameter invalidity over time and static quantization of data [17].

Density-based clustering techniques require the full data to be processed. It is difficult to have all the data in advance in a data stream because the data arrives continually over time. Additionally, they require two methods to obtain the density distribution of the data before clustering based on related information. Clustering data streams, on the other hand, must be performed in a single pass. As a result, there is no viable algorithm for determining the precise density of a data stream with evolving characteristics. In addition, density-based clustering algorithms encounter the problem of false merging, which arises when multiple clusters overlap.

3. Literature Review

Within the literature on clustering algorithms, Clustering of Evolving Data-streams into Arbitrary Shapes (CEDAS) is the first online algorithm to be able to handle evolving data streams [20]. In CEDAS, there are two main stages. Creating micro-clusters or adding data to already-existing micro-clusters is the initial stage, which is then followed by the correcting of the information. Micro-clusters are grouped into kernel and shell regions in the second phase by connecting micro-clusters. This technique involves a graph for each macro-cluster that shows the intersection of each micro-cluster.

Improved Clustering Online Data Streams into Arbitrary Shapes (i-CODAS) is a further improved edition of CODAS. It works according to the idea that each micro-cluster's local radius should be preserved individually [21]. CODAS, like most of the existing density-based clustering algorithms, keeps every micro-cluster radius global and constant. In practice, nonetheless, determining the proper micro-cluster radius can be quite challenging, and for certain micro-clusters, a global radius may not even be the best option. A bad radius selection significantly lowers the clustering quality. To overcome this challenge, every time new data arrive at the cluster, i-CODAS changes the radius online to its local optimal value. Micro-clusters, often referred to as metadata, are used to summarize the data, and a clustering graph is used to show how the micro-clusters are related to one another. Finally, the clustering graph is used to create the shape of the cluster. Yet, memory consumption is considered the main limitation of the proposed algorithm.

CEDAS was then further enhanced by the proposal of a new algorithm known as buffer-based online clustering for evolving data stream (BOCEDS) [9]. One of the major limitations identified within CEDAS is solved by altering the micro-cluster radius repeatedly until it reaches its local optimal value; besides that, BOCEDS offers a buffer for preserving pointless micro-clusters and an online pruning technique for deleting those micro-clusters that are only temporarily irrelevant. The findings of the experiments demonstrate the superiority of BOCEDS across the other methods for clustering. The proposed algorithm, however, is unable to reduce memory usage without deteriorating additional clustering performance standards. Additionally, the suggested technique cannot recognize erroneous merging clusters that occur when clusters overlapped each other.

In the year after, a new algorithm known as Clustering of Evolving Data Streams via a Density Grid-Based Method (CEDGM) was introduced for using a density grid to cluster data streams [7]. This method comprises two separate stages and is conducted online. Core micro-clusters (CMCs) are created during the first stage, and CMCs are combined during the second phase to create macro-clusters. To handle data with different densities and levels of noise, the grid-based method was used as an outlier buffer. The created method was demonstrated to be a successful means of reducing distance function calls and improving cluster quality. However, the use of grids has proven to be ineffective in reducing memory usage, particularly when dealing with high-dimensional sparse data. This drawback hinders the performance of the CEDGM method in effectively handling high-dimensional datasets.

An algorithm called DCDGA (Density-based Clustering of Data Streams using Genetic Algorithm) has been introduced as an enhancement to the recently developed CEDGM algorithm [22]. DCDGA is specifically designed for clustering data streams by leveraging a genetic algorithm (GA) approach. By utilizing a GA, DCDGA dynamically adjusts the cluster radius and minimum density threshold to encompass density clusters more accurately. This adaptive mechanism allows for precise clustering based on the data distribution characteristics. Additionally, DCDGA incorporates a Chebyshev distance function, which facilitates the calculation of the distance between the center of CMCs and incoming data points. This distance measurement aids in determining the clustering assignment for new data points. Despite these advancements, DCDGA faces a challenge known as cluster false merging, which occurs when two or more clusters overlap. This phenomenon can lead to inaccurate clustering results and the merging of distinct clusters. Addressing the issue of cluster false merging remains an area of ongoing research to improve the effectiveness of DCDGA and similar algorithms for clustering data streams.

Another improvement of CEDGM was also proposed by Connelly et al. [15]. CEC-Merge is a dynamic clustering algorithm specifically designed for data streams, employing the Chebyshev distance metric to address the issue of false merging. The core objective of this algorithm is to enhance the efficiency of clustering operations. Operating entirely in an online environment, CEC-Merge encompasses two primary processes. In the initial stage, CMCs are generated and subsequently merged to create macro-clusters. To determine the distance between a new data point and the center of existing CMCs, the Chebyshev distance function is employed. This distance calculation plays a crucial role in the clustering process, aiding in the accurate identification and allocation of data points to appropriate clusters. To prevent false merging, CEC-Merge makes use of two critical parameters, namely, minimum links and time intervals. However, the suggested method has a significant disadvantage in that it requires many distance function calls to calculate the distance between the data point and the CMC or outlier centers, which is particularly inefficient when dealing with high-dimensional data.

DyClee, a new dynamic clustering method for live monitoring of a changing environment [23]. It is a holistic strategy for observing emerging settings that employ a distance- and density-based two-stage clustering method. Data samples are

progressively supplied in real time into the distance-based clustering phase, at which they are aggregated to form clusters in this method. In the case of multi-density distributions, the technique can identify clusters with significant levels of overlaps despite imposing any inferences about cluster convexity. The outcomes of the experiments show that the introduced method reacts fast to data streams and possesses a high rate of anomaly detection. Yet, the DyClee fails to take drift data stream clustering into account.

Recently, an algorithm called DeepStream was proposed by Maia et al. [24]. DeepStream is an advanced approach for stream temporal clustering and anomaly detection that is based on autoencoders. It is specifically designed to handle dynamic data streams and detect consecutive as well as overlapping clusters. One of the key strengths of DeepStream is its ability to efficiently process high-dimensional feature spaces in real time. The technique leverages stacked autoencoders to compress the size of unbounded data streams and represent clusters effectively. DeepStream operates in two distinct phases: offline and online. During the offline phase, a neural network is trained on a labeled dataset to learn a model that can generate compact representations of each record. In the online phase, the learned model is applied to continuously generate clusters from incoming data streams. This results in improved clustering performance and enhances the detection of anomalies within the data. However, it's worth noting that DeepStream does not explicitly address the challenge of concept drift, which refers to the inherent changes in the underlying data distribution over time. Addressing concept drift remains an important area of research in the context of stream clustering algorithms like DeepStream.

4. Research Background

The data stream was introduced in the late twentieth century with the advancements in technologies that empowered rapid and large-scale data acquisition [25]. The data stream model attracted attention because several assumptions made by traditional static data models were violated in many application domains. Static data models assumed that (i) data are finite, (ii) memory is unlimited, and (iii) data can be accessed randomly. The benefits of these assumptions are well-known: algorithms could read every data point several times, processing time was not strictly bounded, and data points could be accessed in any order, for example, finding the two closest data points in the whole dataset [26].

The data stream has three distinctive characteristics, namely, volume, velocity, and variety, which present several issues when it comes to developing data clustering methods for extracting hidden knowledge. Volume represents the volume of a data stream, velocity represents the speed at which data are generated, and variety represents the variation in structure within a data stream [9]. Next, the paper presents important definitions:

Definition 1 (**Data Stream).** *The data stream is represented by an ordered set* $D = \{X_t\}$ *where* $X_t = (x_{t1}, x_{t2}, \ldots x_{td}) \in \mathbb{R}^d$.

The amount of data generated in many applications inhibited using traditional machine learning techniques. Take an IoT smart city system where millions of data records are generated daily as an example [27]. In this example, the data are too big to be loaded into the main memory, and traditional data processing techniques that have high complexity cannot be applied to the data. Since memory is limited, certain tasks, such as calculating a dissimilarity matrix for the whole dataset, are not feasible. This raises challenges for researchers when applying traditional machine learning tasks such as clustering and classification in these scenarios.

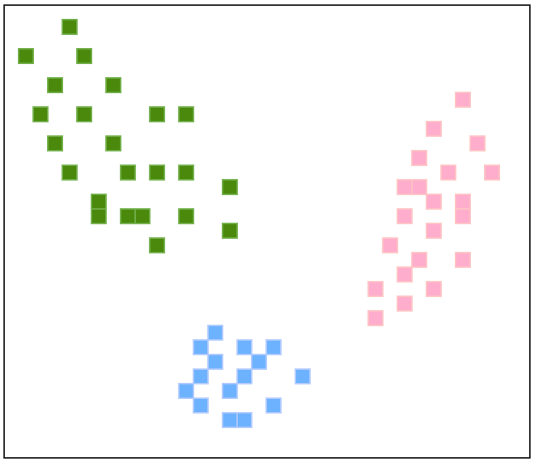
Data stream clustering provides the foundation for resolving a variety of problem classes in a variety of application areas, including data mining, pattern identification, data classification, and image processing [28]. It is the cornerstone of exploratory data analysis in machine learning. It involves classifying data points into clusters, with data points in the same clusters being closer to one another than data points within distinct clusters [6].

Definition 2 (Clustering Data Stream). *The clustering data stream is the process of partitioning the data stream into a set of clusters* $\{C_1, C_2, \ldots C_K\}$ *where similar data points are placed together as one cluster* C_i *using similarity function, whereas dissimilar data points are separated into different clusters or categorized as noise (outliers).*

Clustering is an unsupervised technique that is generally performed when there is no (or little) knowledge about the data. It aims at simplifying data analysis by finding and studying representatives of the clusters, rather than every data point. Unlike clustering static datasets, clustering data streams present unique obstacles [29]. The data stream is a continuous, infinite stream. Thus, it's difficult to retain the whole data stream in memory. Furthermore, data stream flows only once; hence, several scans are not possible [30]. Also, the data stream requires quick and real-time handling to keep up with the huge volume of data arriving each second [31]. An example of data stream clustering is demonstrated in Figure 1, where three clusters are identified. The data points in each cluster can be represented by a single point.

Data stream clustering refers to the process of grouping or partitioning data points in an online or real-time manner as they arrive in a continuous, high-speed stream [1]. Unlike traditional batch clustering, where all data are available upfront, data stream clustering algorithms must handle data points sequentially, often with limited memory and computational resources [17]. The goal of data stream clustering is to discover and adapt to evolving patterns, trends, and clusters in the data stream, providing timely and meaningful insights without the need for storing or revisiting past data points. It is a challenging task due to the dynamic nature of data streams, concept drift, varying data distributions, and the need for efficient and scalable algorithms that can handle the velocity and volume of incoming data [6]. It has become a progressively important, pervasive, and critical method for pattern discovery in data stream analysis. However, data streams introduce additional clustering issues, including

Figure 1
An example of data clustering where three clusters are identified



clustering within limited time, memory and high dimensionality, and processing data only once as it arrives, and the need to treat data dynamically to capture underlying cluster changes [32]. Given these challenges, an efficient clustering algorithm should be able to handle fast and unbounded data, handle data in a short memory, have the capability to process high-dimensional data, and also cluster data within an evolving data stream [1].

Definition 3 (Clusters' False Merging). *Cluster's false merging refers to a situation in which two or more distinct clusters are erroneously combined into a single cluster during the clustering process.*

Clusters' false merging occurs when there is overlap or proximity between clusters, leading to incorrect grouping of data points [16]. False merging can negatively impact the accuracy and reliability of clustering results, as it distorts the underlying structure of the data. False merging can arise due to various reasons, including [15, 17]:

1) Data overlapping: When clusters have overlapping regions or share common data points, traditional clustering algorithms may struggle to accurately separate them. This can result in the merging of clusters that should be distinct.
2) Noise or outliers: The presence of noise or outliers in the dataset can introduce challenges in cluster identification. If outliers are mistakenly included in a cluster, it may lead to the merging of multiple clusters into one.
3) Inadequate distance metrics: The choice of distance metrics used to measure the similarity between data points can impact the clustering process. If the distance metric does not adequately capture the underlying patterns and dissimilarities in the data, false merging can occur.
4) Insufficient cluster separation: When the clusters in the dataset are not well-separated, either due to their inherent characteristics or the complexity of the data, clustering algorithms may struggle to accurately distinguish between them. This can lead to the merging of clusters that should be separate.

The implications of cluster's false merging can be significant. It can distort the representation of distinct patterns or groups within the data, leading to incorrect interpretations and decisions [16]. Clustering algorithms that are prone to false merging may result in the loss of valuable insights and hinder the effectiveness of downstream analyses or applications that rely on accurate cluster tasks [15, 17].

5. Discussion

Table 1 summarizes and compares the previously mentioned density-based clustering algorithms. The main characteristics for evaluating the algorithms include the window model used, the algorithms' working principles, and data processing method, the clustering shape generated by the algorithms, the ability to handle the evolving nature, the applicability for a high-dimensional data stream, the scalability, the ability to perform the work within limited time and limited memory, the ability to detect noise and finally the ability to detect cluster's false merging.

As shown in Table 1, algorithms like CEDAS, BOCEDS, i-CODAS, and CEDGM are online density-based clustering algorithms. They construct clusters in real time, with the clustering results provided instantly. Evolving is one of the most important characteristics of the data stream and therefore must be processed with caution. Other algorithms such as DCDGA and CEC-Merge are also online, and they can generate clusters with arbitrary shapes. DCDGA, for instance, can adjust its parameters for the cluster radius and minimum density threshold. This enables the algorithm to have a better performance when clustering evolving data streams. On the other side, CEC-Merge is equipped with the ability to detect clusters' false merging, which is a critical issue that occurs when two or more clusters overlapped each other. Nevertheless, when CEC-Merge deals with high-dimensional data, it calculates the distance across the data point and the CMC or outlier centers, necessitating a large number of distance function calls. Finally, a clear observation found is that most of the algorithms discussed are not capable of detecting clusters' false merging. The only two algorithms discussed in Table 1, which have addressed the issue of detection of cluster false merging, are CEC-Merge and CyClee. Yet, CEC-Merge still has some limitations, such as the requirement to have a high number of distance function calls. DyClee, on the other hand, fails to handle the clustering of drift data.

Given that, the available literature provides a variety of techniques that try to solve the issue of clustering evolving data streams. Yet, the issue of clusters' false merging, which occurs when two or more clusters overlap on top of each other, is still a major concern. The avoidance of false merging of clusters improves the quality performance of the clustering algorithm. This is because of its capacity to identify false merging caused by evolution and avoid it by employing the time interval and the minimal linkages and does not indicate cluster unification. Yet, false merging was not considered in most of the existing algorithms. Few algorithms however did attempt to address the issue such as CEC-Merge [15]. However, the CEC-Merge technique, particularly for high-dimensional data, involves numerous distance function calls for determining the distance across the data point and the CMC or outlier centers. Hence, an ATI technique is developed for preventing the false merging of overlapped moving clusters, which records the historical behavior of the data stream.

Furthermore, to situate the proposed ATI mechanism within the broader literature, we compare it against adaptive-windowing and memory-based clustering strategies.

Adaptive-window methods. In works such as Learning from Time-Changing Data with Adaptive Windowing [33], the idea is to dynamically adjust the size of the sliding time window to emphasize more recent data and discount older observations. These methods maintain a window of recent data points whose size expands or contracts in response to detected drift or statistical changes. The window adaptation is typically global or semi-global in nature and is used for classification or density tracking.

In contrast, ATI does not adjust a global window of stored data. Instead, ATI operates at the cluster level: it maintains per-cluster timers that track how long pairs of clusters remain in proximity and dynamically adjusts the threshold for merging based on that localized temporal behavior. Thus, ATI provides link-level temporal adaptation, rather than a monolithic window over the entire data stream.

Memory-driven (memory-based) approaches. More recently, memory-based clustering methods proposed in Reference [34] adopt explicit memory modules or buffer structures to store past micro-cluster summaries and use them to guide merging, splitting, or reactivation of clusters. These techniques exploit the stored memory to revisit previous cluster configurations or recall clusters that might have disappeared under transient drift.

ATI differs from pure memory-based strategies in that it does not require a large, persistent memory store of historical clusters. Instead, ATI embeds a lightweight temporal memory in the form of link timers and velocity history for cluster pairs. While memory-based methods may enable reactivation or rollback of

Table 1
Summary of reviews of density-based algorithms

Algorithms	Window model	Working principle	Data processing	Cluster shape	Evolving	High dimensionality	Scalable	Limited time	Limited memory	Noise detection	False merging detection
CEDGM [7]	Fading	Micro-cluster	Online	Arbitrary	√	√	√	√	√	√	X
BOCEDS [9]	Fading	Micro-cluster	Online	Arbitrary	√	√	√	√	√	√	X
CEC-Merge [15]	N/A	Micro-cluster	Online	Arbitrary	√	X	√	√	√	√	√
i-CODAS [21]	Fading	Micro-cluster	Online	Arbitrary	√	√	√	X	X	√	X
DCDGA [22]	N/A	Micro-cluster	Online	Arbitrary	√	√	√	√	√	√	X
DyClee [23]	Sliding window	Micro-cluster	Online	Arbitrary	√	√	√	X	√	√	√
DeepStream [24]	N/A	N/A	Online–Offline	N/A	X	X	√	X	X	√	√
CEDAS [25]	Fading	Micro–macro-cluster	Online	Arbitrary	√	√	X	X	√	√	X
AutoEncoder-based [26]	N/A	N/A	Online	Arbitrary	√	X	√	√	√	√	X

clusters using stored summaries, ATI focuses on preventing false merges proactively by using the temporal persistence signal and relative motion between clusters.

In short, ATI is distinct in combining local temporal link persistence and velocity-sensitive merging decisions at the cluster level, without the overhead of large global windows or maintaining full historical cluster stores. This positioning further clarifies how ATI complements but does not replicate existing adaptive-window or memory-based techniques. Although this study benchmarked BOCEDS-ATI against widely recognized and well-established online clustering algorithms such as CEDAS, BOCEDS, i-CODAS, and MuDi-Stream, it is important to recognize the emergence of deep learning–based and hybrid clustering models that have recently advanced data stream analysis. Methods such as DeepStream [24] and AutoEncoder-based incremental clustering [26] integrate neural network architectures with incremental density estimation to enhance adaptability in high-dimensional data environments.

These approaches demonstrate strong performance in feature extraction and noise handling but often involve high computational and memory overhead, making them less suitable for real-time, resource-constrained applications. In contrast, BOCEDS-ATI is designed as a lightweight, online algorithm emphasizing temporal adaptability and computational efficiency, particularly for continuous IoT and sensor data streams.

Nonetheless, incorporating such deep learning–based and hybrid benchmarks in future evaluations could provide a broader comparative perspective and help identify opportunities to integrate representation learning with the ATI mechanism. Future work will therefore consider hybrid models that couple feature learning layers with the proposed spatiotemporal clustering framework.

6. Methodology

The proposed BOCEDS-ATI algorithm fundamentally advances previous extensions such as BOCEDS and CEC-Merge by introducing a novel ATI mechanism that incorporates temporal awareness into the clustering process. Unlike BOCEDS, which relies solely on buffer-based micro-cluster maintenance without temporal adaptation, BOCEDS-ATI dynamically adjusts the linkage between clusters using ATIs and relative velocity analysis. This mechanism records and analyzes the historical behavior of cluster interactions, thereby determining whether the proximity between clusters is transient or persistent.

In contrast to CEC-Merge, which employs a static time interval and minimum link threshold to avoid false merging, the ATI mechanism in BOCEDS-ATI adapts the interval based on the evolving nature of the data stream. This dynamic adjustment allows the algorithm to distinguish between temporary overlaps caused by cluster motion and genuine cluster convergence. Consequently, BOCEDS-ATI reduces false merging by leveraging relative velocity-based decision-making and temporal adaptation, features that are absent in previous algorithms.

In short, the key innovation of BOCEDS-ATI lies in combining time-adaptive linkage thresholds with velocity-sensitive analysis within a buffer-based online framework, enabling the model to prevent false merging of clusters in evolving data streams more effectively than existing approaches.

The development of this algorithm begins with the literature review, where the issue of clusters' false merging has been highlighted. The handling of false merging will be handled by "designing an algorithm for preventing the false merging of overlapped moving clusters using adaptive time interval." Following that, the BOCEDS baseline technique is applied. Combining the three established algorithms with BOCEDS results in a new algorithm known as BOCEDS-ATI. The performance of the proposed BOCEDS-ATI algorithm is evaluated by comparing it to BOCEDS using ordinary evaluation measures and benchmarking datasets. BOCEDS [9], BOCEDS Inc, CEDAS [20], and MuDi-Stream [35] are the benchmarks. Spiral, Landsat, DNA, Traffic Dataset, and KDD Cup'99 were utilized in the assessment.

The performance of the BOCEDS-ATI algorithm is evaluated through two distinct assessment methods: quality evaluation and efficiency evaluation. These methods provide comprehensive insights into the algorithm's effectiveness. Quality evaluation measures, including anomaly detection ratio (ADR), Rand index (RI), adjusted Rand index (ARI), and normalized mutual information (NMI), are employed to assess the quality of clustering results. ADR quantifies the algorithm's ability to accurately detect anomalies, while RI, ARI, and NMI measure the agreement between the true cluster labels and the predicted cluster assignments. Efficiency evaluation focuses on assessing the algorithm's computational efficiency and resource usage. Memory usage and computational cost are evaluated to understand the algorithm's efficiency in terms of memory requirements and computational complexity. By employing both quality and efficiency evaluation methods, the performance of the BOCEDS-ATI algorithm can be comprehensively analyzed, providing insights into its clustering accuracy and computational efficiency.

To assess the scalability of the proposed BOCEDS-ATI algorithm, we provide a formal analysis of its time complexity in relation to the number of incoming data points (n), the number of maintained micro-clusters (m), and the dimensionality of the data (d).

1) Micro-cluster maintenance: Each new data point is either assigned to an existing micro-cluster or used to initialize a new one. The nearest-cluster search typically requires $O(m \times d)$ operations using a linear scan, or $O(\log m)$ if spatial indexing (e.g., KD-tree) is employed. Hence, the incremental update cost per data point is $O(m \times d)$.
2) Buffer and pruning operations: BOCEDS-ATI maintains a temporary buffer of inactive clusters. Buffer updates and pruning are bounded by $O(m)$, as obsolete clusters are checked periodically against a threshold.
3) Adaptive time interval (ATI) mechanism: The ATI introduces additional link-timer updates between pairs of *CMCs* that come into proximity. Assuming each cluster interacts with at most k neighbors ($k \ll m$), the ATI update cost is $O(k)$ per iteration, resulting in a total ATI overhead of $O(m \times k)$ across all clusters. Because k is typically small and bounded, this term behaves linearly with m.
4) Merging and velocity computation: The relative velocity calculation between neighboring clusters requires $O(k)$ distance evaluations. Since this step is only triggered when potential merging conditions are met, the amortized cost remains $O(m)$.
5) Overall complexity: Combining the above, the total amortized time complexity per update is:

$$O(m \times d + m \times k) \approx O(m \times d)$$

assuming k and d are significantly smaller than m.

In comparison, the baseline **BOCEDS** algorithm exhibits the same asymptotic complexity, $O(m \times d)$, whereas

CEC-Merge incurs a higher $O(m^2)$ cost due to pairwise distance evaluations at each iteration. Thus, the inclusion of ATI preserves near-linear scalability while introducing only a marginal constant-factor overhead for time tracking and velocity updates.

This analysis demonstrates that BOCEDS-ATI remains computationally feasible for real-time streaming environments, capable of processing incoming data with linear scalability relative to the number of maintained micro-clusters.

6.1. Definitions of algorithm parameters

Certain interdependent parameters are defined prior to implementing the proposed BOCEDS-ATI algorithm according to professional expertise in the topic. The subsequent clustering parameters are defined by the proposed algorithm.

1) CMC Threshold: The density threshold employed for converting data received at a particular grid into a CMC.
2) Link Timers (*LinkTimer*): Refers to an array that counts the number of times a link appeared between every two CMCs.
3) Link Interval (*LinkInterval*): Refers to the threshold of the time interval of inserting a link among two CMCs to merge them into one while avoiding false merging.

6.2. Algorithm operation

As shown in Figure 2, a block diagram represents the developed algorithm. The algorithm is designed to process a data stream, where the incoming data are fed into the core component called BOCEDS-ATI. The core algorithm incorporates a memory-based approach to address the issue of false merging when overlapping occurs between two or more clusters. The algorithm's primary objective is to prevent the merging of moving clusters that overlap each other. By utilizing a memory-based approach, the algorithm is able to effectively handle this scenario and maintain the integrity of individual clusters. The specific mechanisms employed within the BOCEDS-ATI component are designed to detect and address false merging, ensuring accurate and reliable clustering results in the presence of overlapping clusters.

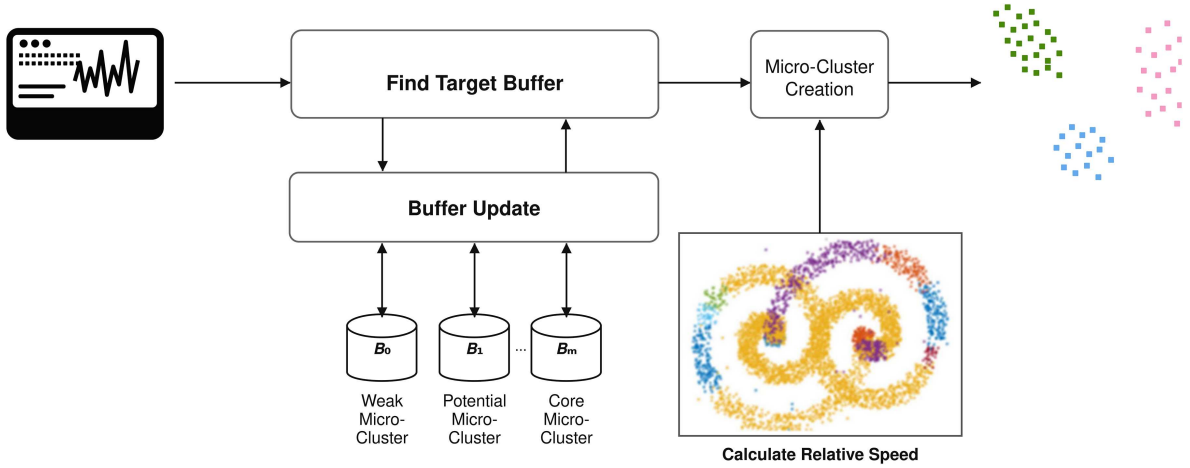
6.3. Developed algorithm

This part illustrates the process of developing the proposed BOCEDS-ATI. It incorporates two developed algorithms, namely, BOCEDS as the core and ATI as an extension to prevent clusters' false merging using ATIs. A single-phase clustering method is extended by the proposed ATI algorithm to assist in generating clusters across changing data streams in real time. It may work on its own or in combination with an already-existing clustering method like BOCEDS. BOCEDS-ATI is an entirely novel clustering algorithm comprised of the proposed ATI algorithm and the BOCEDS core. The subsequent section introduces and presents the details of the proposed ATI algorithm. This algorithm is specifically designed to address the challenges associated with handling evolving data streams and overcoming issues such as cluster false merging. The section will provide a comprehensive description of the ATI algorithm, including its key components, steps, and underlying principles.

1) Adaptive time interval (ATI)

The ATI technique is designed with the objective of producing high-quality clusters in order to mitigate the issue of false merging that can arise when two or more clusters overlap. This technique operates in an online manner, meaning it can handle evolving data streams in real time. It leverages the Euclidean distance function to calculate velocities specifically tailored for binary class data. By utilizing this approach, the ATI technique aims to provide accurate and reliable clustering results, particularly in scenarios where overlapping clusters pose a challenge. This works by selecting a reference point and then calculating the relative speed with respect to the speed threshold. If the relative speed exceeds the speed threshold, split the clusters into two clusters; otherwise, merge the clusters.

Figure 2
The developed evolving data streams with the ability to prevent clusters' false merging using adaptive time interval (BOCEDS-ATI) algorithm



Assuming that the stream data are received as a batch that contains a set of points, to prevent false merging when the points share the same feature values (the location in the feature space), we extend them by a Boolean feature that has the role of splitting the points that have two different velocities into different clusters. The approach is based on the equations as follows:

If point A has the velocityv_A and the point B has the velocityv_B

$$v_A = A(t) - A(t-1) \tag{1}$$

$$v_B = B(t) - B(t-1) \tag{2}$$

$$|v_{rel}| = |v_B| - |v_A| \tag{3}$$

Then we compare$|v_{rel}|$ with an adaptive threshold, which represents the mean of the relative velocities of the points in the buffer with respect to a reference point, to set the decision of 1, which means they are in two different clusters, or 0, which means they are in the same cluster. The addition of 1 or 0 is shown in lines 11 and 13, respectively, in the pseudocode given in algorithm (1). This decision is an extended Boolean value added to the original features.

As displayed in Figure 3, the flowchart of the ATI algorithm illustrates its step-by-step process. This flowchart provides a visual representation of the algorithm's workflow and the sequence of operations performed. Furthermore, Algorithm 1 presents the pseudocode implementation of the ATI algorithm, outlining the specific instructions and computations involved at each stage. The combination of the flowchart and pseudocode serves as a comprehensive reference for understanding and implementing the ATI algorithm effectively. The process starts by receiving input, which is, in this case, the data stream characterized by being evolving. For each batch (P) received from the data stream, the algorithm selects a reference point that will be used later for relative speed calculation. The algorithm then compares the relative speed with the speed threshold to decide when to merge clusters to avoid false merging of moving clusters when they overlapped with each other.

Algorithm 1: ATI speed calculation for binary classes data

Input:

1. *Data*: data stream.

Output:

1. newData: updated data stream

Procedure:

1. **start algorithm**
2. newData ←[]
3. **for** each Batch B of Data **do**
4. select reference point Ref
5. **for** each point P in B **do**
6. *relSpeeds (P) ← relativeSpeed (P, Ref)*
7. **end for**
8. *speedThresh ← mean (relSpeeds)*
9. **for** each point P in B **do**
10. **if** relSpeeds (P) > = speedThresh **then**
11. B(P) ← [*B(P)1*]
12. **else**
13. B(P) ← [*B(P)0*]
14. **end if**
15. **end for**
16. newData ← [*newData B*]
17. **end for**

end algorithm

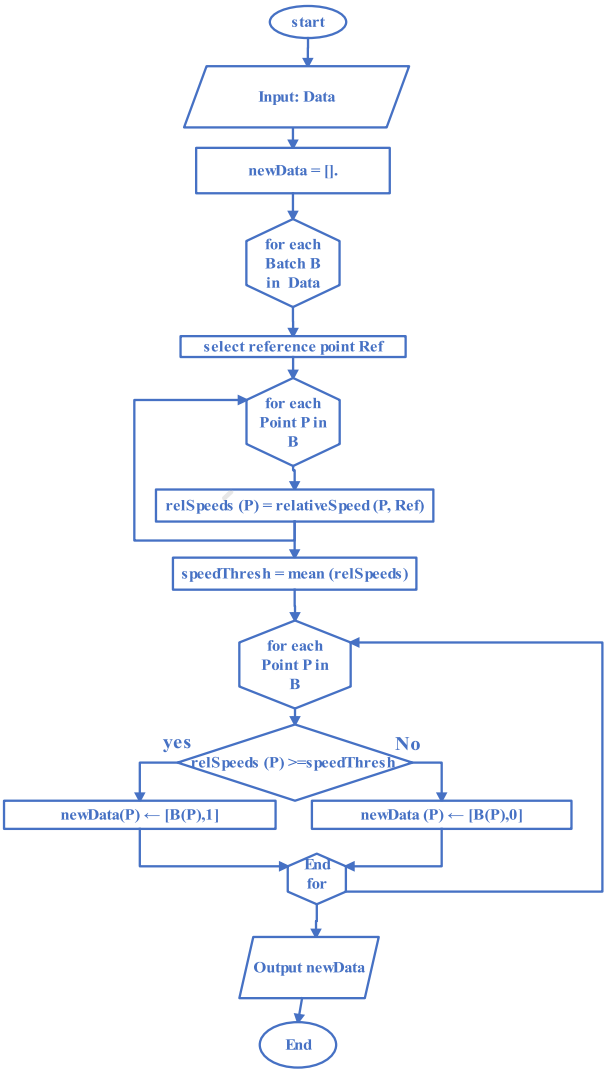
2) BOCEDS micro-cluster searching

As indicated by algorithm (2), the objective of micro-cluster searching is to identify the closest struct target to which an existing point *x_i* can be attached. There are three different categories of structs: WMCs, which stand for the weak micro-cluster set; PMCs, which stand for the prospective micro-cluster set; and CMCs, which stand for the core micro-cluster set. The findings are the relevant struct with the point added once it has been updated. For the search, Equation 6 is applied, which states that the distance between the point and the centroid of the structure must be less than the structure's radius.

$$d(Xi, C) < R \tag{4}$$

Preference is given to searching for WMCs first, PMCs second, and CMCs final. Whenever the criteria for d(*Xi*, *C*)*R* fail to

Figure 3
Adaptive time interval flowchart



be fulfilled, the outcome is going to be either one of these struct types or an empty set.

Algorithm 2: BOCEDS Micro-Cluster Searching

Input:

1. X_i: incoming data instance
2. $CMCs$: set of core micro-clusters
3. $PMCs$: set of potential micro-clusters
4. $WMCs$: set of weak micro-clusters
5. MC: micro-cluster repository

Output:

- T: the micro-cluster to which X_i is assigned

Procedure:

1. **Begin**
2. **Initialization:**
   - Set the selected micro-cluster to null:

$$T \leftarrow \varnothing$$

3. Identify a weak micro-cluster $Q \in WMCs$ that satisfies Equation (4).
4. **If** such a cluster exists **then**
   - Assign $T \leftarrow Q$.
   - Jump to the return step.

   End if
5. Search within the potential micro-cluster set $PMCs$ for a cluster Q meeting Equation (4).
6. **If** a valid cluster is found **then**
   - Update $T \leftarrow Q$.
   - Proceed directly to the return step.

   End if
7. Examine the core micro-cluster set $CMCs$ to locate a cluster Q that fulfills Equation (4).
8. **If** such a cluster is identified **then**
   - Set $T \leftarrow Q$.
   - Move to the return operation.

   End if
9. **Return:** output T.

end algorithm

3) BOCEDS micro-cluster update

The purpose of the micro-cluster update is to perform two functions: As stated in equation (5), the first stage is to increase the number of points associated with T by one.

$$\text{Local density, } N_{t+1} = N_t + 1 \tag{5}$$

The first stage is to ascertain if the recently arrived data item x_i will cause its target T to become a CMC. When the condition specified in algorithm (6) line (3) is met, this would be realized. As seen in lines 5 and 6 of equation (8), the radius will be changed in this case, and the energy will be adjusted to 1.

$$\text{Radius, } R_{t+1} = min\left(\left[R_t + \left\{\frac{2 \times d(X_{t+1}, C_t)}{R_t} - 1\right\} \times \frac{1}{\text{Decay}}\right], R_{max}\right) \tag{6}$$

Similar to how the radius and energy will be changed when a data point previously falls to the Micro Cluster (MC), as seen in algorithm (6) lines 8 and 9, equation (7) provides an example.

$$\text{Energy, } E_{t+1} = E_t + \left\{\frac{R_t - d(X_{t+1}, C_t)}{R_t}\right\} \times \frac{1}{\text{Decay}} \tag{7}$$

The final step is to confirm that the data point relates to the shell, as specified in algorithm (6) line 12. Equations (10) and (11) are used in this scenario to revise both the center and the total number of points in the shell.

$$N'_{t+1} = N'_t + 1 \tag{8}$$

$$\text{Center, } C^k_{t+1} = \frac{\left(N'_{t+1} - 1\right) \times C^k_t + X^k_{t+1}}{N'_{t+1}} \tag{9}$$

The algorithm outputs a revised CMC, PMC, or WMC. The following phase is to determine if it is necessary to move micro-clusters from one buffer to another. This is explained in the following subsection.

Algorithm 3: BOCEDS Micro-Cluster Update

Input:

1. X_i: incoming data instance
2. $CMCs$: set of core micro-clusters
3. $PMCs$: set of potential micro-clusters
4. $WMCs$: set of weak micro-clusters
5. T: micro-cluster associated with X_i

Output:

1. Updated $CMCs$
2. Updated $PMCs$
3. Updated $WMCs$

Procedure:

1. **Begin**
2. Update the local density N_t of micro-cluster T according to Equation (5).
3. **If** $\left[(T \in PMCs \wedge N_{t+1} = Th_{density}) \vee (T \in WMCs)\right]$ **then**
   - Insert T into the core micro-cluster set:

$$CMCs \leftarrow CMCs \cup T$$

   - Recalculate the cluster radius R_{t+1} of T using Equation (6).
   - Initialize the energy value:

$$E_{t+1} \leftarrow 1$$

4. **Else if** T already belongs to the core micro-cluster set **then**
   - Update the radius R_{t+1} of T using Equation (6).
   - Update the energy E_{t+1} of T following Equation (7).

End if

5. Compute the distance between the data point and the cluster centroid:

$$d \leftarrow \text{distance}(X_i, T.\text{Center})$$

6. **If** $\frac{R_{t+1}}{2} < d \leq R_{t+1}$ **then**

- Update the number of points in the shell region N_{t+1} of T using Equation (8).
- Recalculate the cluster centroid C_{t+1} based on Equation (9).
- Update the cluster connectivity graph using algorithm (6).

End if

End Algorithm

4) BOCEDS moving weak micro-cluster to a buffer

Algorithm (4) presents the pseudocode for transforming a weak micro-cluster. Alongside decay, it takes a pair of micro-clusters, called CMCs and WMCs. The result is a revision for both CMC and WMC. The pseudocode begins with the energy of all CMCs having a value of $\frac{1}{Decay}$. Following that, the energy is evaluated for every T in CMCs; if the energy is less than zero, T is eliminated from all CMCs and placed in WMCs. The energy of CMCs will be adjusted to 0.5 once T is eliminated. Following the conversion of CMCs to WMCs, the subsequent phase is to examine the WMCs and PMCs for updates on micro-cluster elimination, as detailed in the following subsection.

Algorithm 4: BOCEDS Moving Weak Micro-Clusters to a Buffer

Input:

1. $CMCs$: collection of core micro-clusters
2. $WMCs$: collection of weak micro-clusters
3. $Decay$: decay interval parameter

Output:

1. Updated $CMCs$
2. Updated $WMCs$

Procedure:

1. **Begin**
2. Apply an energy reduction of $\frac{1}{Decay}$ to every micro-cluster in the core set.
3. **For each** micro-cluster $T \in CMCs$ **do**

- **If** the energy value of T satisfies

$$T.\text{Energy} \leq 0$$

then

1. Remove all graph connections associated with T:

$$T.\text{Edges} \leftarrow \emptyset$$

2. Eliminate T from the adjacency lists of all remaining core micro-clusters.
3. Reset the membership indicator of T:

$$T.M \leftarrow 0$$

4. Remove T from the core micro-cluster set $CMCs$.
5. Reinitialize the energy of T:

$$T.\text{Energy} \leftarrow 0.5$$

6. Insert T into the weak micro-cluster buffer:

$$WMCs \leftarrow WMCs \cup T$$

4. **End if**
5. **End for**

end algorithm

5) BOCEDS micro-cluster removal

Algorithm (5) provides the pseudocode for micro-cluster elimination. PMCs, WMCs, and decay are used as inputs, and PMCs and WMCs are outputs. The algorithm begins by lowering energy to a value of $\frac{1}{Decay}$ across every micro-cluster within WMCs. The energy of every micro-cluster in WMCs is then examined, and if it is less than zero, the micro-cluster is deleted from WMCs. Furthermore, the energy across every micro-cluster within PMCs is examined, and if it is 0 or less, the micro-cluster is eliminated.

Algorithm 5: BOCEDS Micro-Clusters Removal

Input:

1. PMCs: potential micro-cluster set
2. WMCs: weak micro-cluster set
3. Decay: the decaying period

Output:

1. PMCs: potential micro-cluster set
2. WMCs: weak micro-cluster set

Procedure:

1. **start algorithm**
2. Lower the energy of each MC in the WMCs set by $\frac{1}{Decay}$
3. **for** every W in the WMCs set **do**
4. **if** the energy of W (T.Energy) is ≤ 0 **then**
5. Delete W from the WMCs set
6. **end if**
7. **end for**
8. Lower the energy of each MC in the PMCs set by $\frac{1}{Decay}$
9. **for** every P in the PMCs set **do**
10. **if** the energy of P (T.Energy) is ≤ 0 **then**
11. Delete P from the PMCs set
12. **end if**
13. **end for**

end algorithm

6) BOCEDS update cluster graph

By computing the overlapping distance between the micro-clusters, the micro-cluster graph is updated. A new edge is created, and the graph is revised, as shown by the algorithm (6), if the amount of the overlapping distance d', which is calculated using equation (10) in accordance with the radius R and R' of the two micro-clusters, is less than d.

$$d' = \begin{cases} R + \frac{R'}{2}, & \text{if } R \geq R' \\ R' + \frac{R}{2}, & \text{if } R' > R \end{cases} \quad (10)$$

Algorithm 6: BOCEDS Update Cluster Graph

Input:
1. CMCs: core micro-cluster set
2. T: A core micro-cluster that has been generated or modified
3. G: clustering graph

Output:
1. CMCs: core micro-cluster set
2. G: clustering graph

Procedure:
1. **start algorithm**
2. **for** every C in CMCs **do**
3. d ← *Euclideandistance(T.c, C.c)*
4. $d' \leftarrow$ *intersectingdistance(T, C)* from Eq. 3
5. **if** $d \leq d'$ **then**
6. T.EL = T.EL ∪ Edge (T, C)
7. C.EL = C.EL ∪ Edge (T, C)
8. **end if**
9. **end for**
10. **if** any micro-cluster edge list has changed **then**
11. Set a new number of macro-clusters throughout the graph.
12. **end if**

end algorithm

7) ATI speed calculation for binary classes data

The goal of ATI speed is to prevent false merging when each of the two clusters is moving in a different direction. To elaborate on this concept, we present two clusters overlapping as shown in Figure 4. The first one consists of a set of points marked with a blue star, and the second one consists of a set of points marked with a black circle. In the visual representation, the direction of the points belonging to the first cluster is indicated by black arrows, while the direction of the points belonging to the second cluster is represented by blue arrows. This color-coded arrow scheme provides a clear visual distinction between the two clusters and helps in understanding their respective orientations. As it is observed, by measuring the relative velocity between them, we can add a feature that differentiates between their points and assists in avoiding false merging between them.

The pseudocode of ATI speed calculation is presented in algorithm (2). The input of the algorithm is the data stream, and the output is the new data stream after adding the additional feature. The algorithm selects reference points from each batch of data and uses them to calculate the relative speed between the points and the reference points. The average speed is used as a threshold to partition the points into two sets of points: one is extended by feature 0, and the other one is extended by feature 1.

Algorithm 7: ATI speed calculation for binary classes data

Input:
1. *Data*: data stream

Output:
1. newData: update data stream

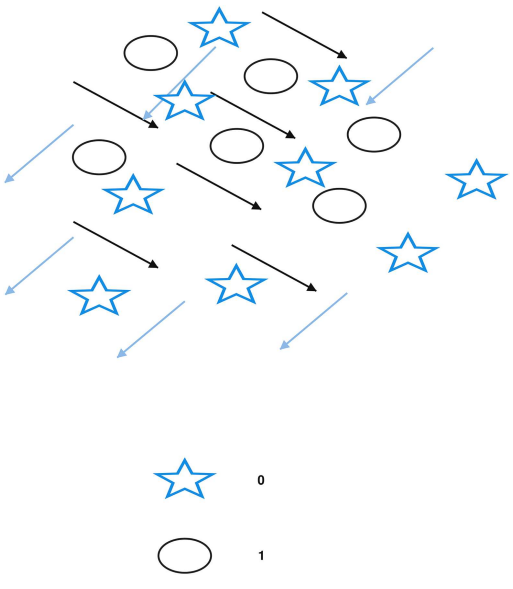
Procedure:
1. **start algorithm**
2. *newData*←[]
3. **for** each Batch B of Data **do**
4. select reference point Ref
5. **for** each point P in B do
6. *relSpeeds(P)* ← *relativeSpeed(P,Ref)*
7. **end for**
8. *speedThresh* ← *mean(relSpeeds)*
9. **for** each point P in B **do**
10. **if** relSpeeds(P)>=speedThresh **then**
11. B(P) ← [B(P) 1]
12. **else**
13. B(P) ← [B(P) 0]
14. **end if**
15. **end for**
16. nweData ← [newData B]
17. **end for**

end algorithm

8) ATI BOCEDS

An extension to BOCEDS to handle false merging is proposed in BOCEDS-ATI. The pseudocode of the algorithm is given algorithm (3). The algorithm is like BOCEDS with the difference in line 10, where the graph update algorithm is changed to the one presented in algorithm (6), Appendix 1. Hence, the algorithm receives two additional inputs, namely, linkTimer, which is an array that counts the number of times a link appeared between each two CMCs, linkInterval, which is the threshold of the time interval of inserting a link between two CMCs. These two parameters add to BOCEDS the time awareness in order to prevent false merging when two clusters are intersecting temporarily with each other. The details of the functions of these two parameters are presented in the following subsection.

Figure 4
Two overlapping clusters with different velocities of their points



Algorithm 8: ATI BOCEDS

Input:
1. *Data:* data stream.
2. linkTimer
3. linkInterval

Output:
Clusters: the clustering result for each data point in B.

DetectedAnomaly: the anomaly data detected in the stream

Procedure:

1. **start algorithm**
2. *initialization:*
endOfStream ← *false*
initiate: CMCs, PMCs, WMCs "types of micro-clusters"
Tc←*1*
G ← *graph( )*
Clusters ← *[ ]*
3.**while** ¬ *endOfStream* **do**
4. *B* ← *data(Tc)*
5. **for** each data point *Xi in B* **do**
6. *T* ← *searchForTargetMC(Xi, Rmin, Rmax, CMCs, PMCs, WMCs)* algorithm [5]
7. **if** T == [] **then**
8. create a new PMC
9. ***else***
10. update the micro-clusters via algorithm [9] instead of [8] for graph update via linkTimer and linkInterval
11. **end if**
12. update energy of CMC via algorithm [7] and for PMC and WMC via algorithm [8]
13. **end if**
14. Tc←Tc+1
15. **if** Tc ≥ size(data) **then**
16. endOfStream←true
17. **end if**
18. **end while**

end algorithm

9) ATI update cluster graph

The pseudocode of ATI update cluster graph is presented in algorithm (4). The role of the two input variables linkTimer and linkInterval is shown in lines 8–10. As it is observed, when the value of linkTimer exceeds linkInterval, then the edge is added between the two micro-clusters. This is only when the distance between the two micro-clusters' centers is lower than the intersection distance.

Algorithm 9: ATI Update Cluster Graph

Input:

1. CMCs: core micro-cluster set
2. T: A core micro-cluster that has been generated or modified
3. G: clustering graph
4. linkTimer: an array that counts the number of times a link appeared between each two CMCs
5. linkInterval: the threshold of the time interval of inserting a link between two CMCs

Output:

1. CMCs: core micro-cluster set
2. G: clustering graph

Procedure:

1. **start algorithm**
2. **for** each C in CMCs **do**
3. d ← *Euclideandistance(T.c, C.c)*
4. d′ ← *intersectingdistance(T, C)* from Eq. 3
5. **if** d ≤ d′ **then**
6. *linkTimer(T, C)* ← *linkTimer(T, C) + 1*
7. *linkTimer(C, T)* ← *linkTimer(C, T) + 1*
8. **if** *linkTimer(T, C)* ≥ *linkInterval* **then**
9. T.EL = T.EL ∪ Edge (T, C)
10. C.EL = C.EL ∪ Edge (T, C)
11. **end if**
12. **end for**
13. **if** any MC edge list has changed **then**
14. Set a new number of macro-clusters throughout the graph.
15. **end if**

end algorithm

7. Results and Discussion

The BOCEDS-ATI algorithm, which was specifically developed for this study, was implemented using MATLAB R2021a. The algorithm's performance was analyzed on a MacBook Pro 2020, which features a 1.6 GHz Quad-Core Intel Core i7 CPU and 16 GB RAM. This hardware setup provided the computational resources necessary to execute the algorithm and analyze its results effectively. A set of examinations was carried out on five datasets, two of which were actual data and three of which were synthetic data, to assess the effectiveness of the designed BOCEDS-ATI algorithm when comparing it to four benchmark clustering methods. The experimentation process was conducted five times, with each iteration utilizing unique data points. The evaluation of the designed BOCEDS-ATI algorithm involves two key aspects: quality and efficiency. Quality assessment involves measuring metrics such as ADR, RI, ARI, and NMI. On the other hand, efficiency is evaluated by considering factors such as memory usage and computational cost. These measures provide a comprehensive evaluation of the algorithm's performance in terms of both the accuracy of clustering results and the computational resources required.

7.1. Quality evaluation

Clustering quality evaluation plays a vital role in assessing the effectiveness and reliability of clustering algorithms and results. It provides an assessment of the performance and accuracy of clustering techniques in organizing data into meaningful groups or clusters. By evaluating clustering quality, researchers and practitioners can make informed decisions about the suitability of clustering algorithms for specific applications and datasets. The clustering quality evaluation employed to evaluate the proposed BOCEDS-ATI includes the following:

7.1.1. Anomaly detection ratio (ADR)

This metric is produced by dividing the total number of anomaly data points by the number of accurate anomalies discovered. ADR is computed by dividing the count of accurately detected anomalies by the total number of anomalies in the dataset. It provides a measure of the algorithm's effectiveness in distinguishing between normal and anomalous instances. In research by Pandey et al. [36], ADR was utilized to evaluate the ADR.

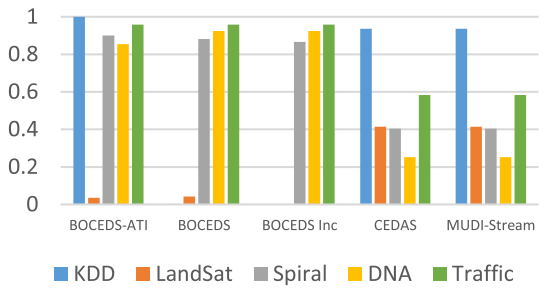
$$ADR = \frac{CAS}{Total\ Anomaly\ Samples} \tag{11}$$

where *CAS* refers to the samples with the correct anomaly identified. The ADR metric provides valuable insights into the algorithm's ability to correctly identify anomalies, minimizing

both false positives (instances incorrectly labeled as anomalies) and false negatives (anomalies missed by the algorithm) are taken into consideration. A higher ADR value indicates a better performance of the algorithm in accurately detecting anomalies [36].

Figure 5 illustrates the ADR findings for several datasets. As indicated in Figure 3, BOCEDS-ATI achieved exceptional results with a median of ADR approximately 1.00 across the KDD dataset, which was greater than the benchmark, alongside a median of approximately 0.95 for traffic data, making it like the BOCEDS and BOCEDS Inc benchmarks. Furthermore, we find that the median of ADR for BOCEDS-ATI in spiral data was 0.9, which is also higher than BOCEDS, which accomplished 0.87, and BOCEDS Inc. DNA and Landsat were found to cause BOCEDS-ATI to perform poorly. The experiment demonstrates BOCEDS-ATI's overall superiority in terms of ADR. The structure of the dataset itself is the cause for several cases resulting in superiority of the benchmarks compared to the proposed algorithm. For instance, in Landsat, the findings of ADR were low across the BOCEDS-ATI, BOCEDS, and BOCEDS Inc algorithms, whereas CEDAS and MuDi-Stream achieved approximately 0.4 of ADR, which is explained by the type of Landsat anomaly that remains undetectable by BOCEDS' buffer-based framework as well as the developed BOCEDS-ATI algorithm. On the contrary, the DNA abnormality was identifiable by both BOCEDS-ATI and BOCEDS at a score exceeding 0.8, while the nature of the datasets did not require the dynamic awareness exhibited by BOCEDS-ATI, the prediction accuracy of BOCEDS and BOCEDS Inc was slightly lower as a result.

Figure 5
ADR evaluation over KDDCup'99, Landsat, Spiral, DNA, and Traffic datasets



7.1.2. Rand index (RI)

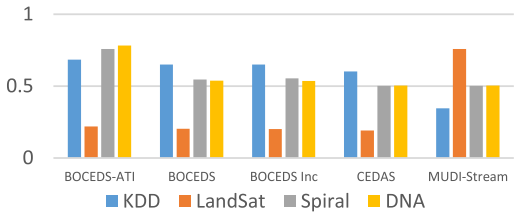
The RI is a popular evaluation metric used to calculate the similarity across two clustering scores or compare a clustering result to a ground truth partition. It assesses the agreement between clusterings by considering the number of agreements and disagreements between pairs of data points [35]. The RI calculates a similarity coefficient ranging from 0 to 1, where 1 indicates a perfect match between the clusterings and 0 indicates no agreement beyond what would be expected by chance. It does not consider the size or distribution of the clusters, focusing solely on pairwise agreements and disagreements. The RI is calculated as follows [35, 37, 38]:

$$\text{RandIndex} = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{12}$$

The RI index is well-known for assessing the resemblance between two divisions and has been frequently employed in many investigations.

Figure 6 depicts the RI results for multiple datasets. The median RI for BOCEDS-ATI in DNA, spiral, and KDD datasets was 0.78, 0.75, and 0.68, respectively, as shown in Figure 4, which is greater than the minimum thresholds. Landsat is the only dataset that led BOCEDS-ATI to perform poorly. The decreased performance in Landsat is due to the structure of the data, which is constant and does not require a buffer. MuDi has improved its efficiency by organizing and detecting various clusters from a density aspect.

Figure 6
RI evaluation over KDDCup'99, Landsat, Spiral, and DNA datasets



7.1.3. Adjusted Rand index (ARI)

The ARI is a widely used evaluation metric for measuring the similarity between two clustering results or comparing a clustering result to a ground truth partition. It is an extension of the RI and considers the effect of chance agreement when assessing the agreement between clusterings. To calculate the ARI, the algorithm constructs a contingency table that counts the number of pairs of data points falling into each combination of clusters in the two clusterings being compared. It then computes the ARI as the adjusted Rand index based on this contingency table, which can be defined as follows:

$$ARI = \frac{\binom{n}{2}(|TP| + |TN|) - [(|TP| + |FN|)(|TP| + |FP|) + (|Fp| + |TN|)(|FN| + |TN|)]}{\binom{n}{2}^2 - [(|TP| + |FN|)(|TP| + |FP|) + (|FP| + |TN|)(|FN| + |TN|)]} \tag{13}$$

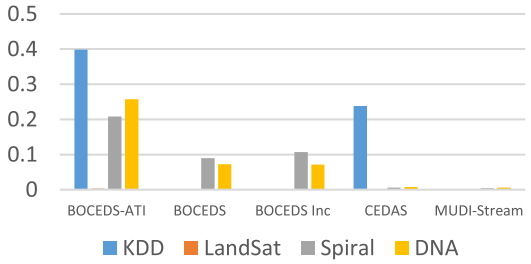
The total number of observations in a partitioned cluster is denoted by ARI. An ARI score usually varies between 0 and 1. If a partition corresponds to the core structure, the index value is 1; otherwise, it is near to 0. ARI has been employed in numerous types of studies [35, 37, 38] to assess data stream clusterings.

Figure 7 depicts the modified RI across the different datasets. The median ARI of BOCEDS-ATI in KDD, DNA, and Spiral was 0.39, 0.25, and 0.20, respectively, which were significantly greater than BOCEDS Inc, BOCEDS, MuDi-Stream, and CEDAS. Landsat's ARI values were nearly negative because of the structure of the data, which impacts the process of separating the data into groups using a non-supervised technique.

7.1.4. Normalized mutual information (NMI)

The NMI, widely employed as a measure for evaluating clustering quality, compares clustering results with a known ground

Figure 7
ARI evaluation over KDDCup'99, Landsat, Spiral, and DNA datasets



truth or reference clustering. It quantifies the mutual information (MI) between the true cluster labels and the predicted cluster tasks [7, 9, 20, 38]. NMI is described as follows:

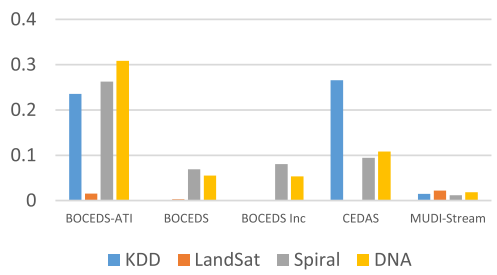
$$\mathrm{NMI}(C, G) = \frac{\sum_i \sum_j \frac{n_{ij}}{n} \log \frac{n \times n_{ij}}{a_i \times b_j}}{\sqrt{\sum_i a_i \times \log\left(\frac{a_i}{n}\right) \times \sum_j b_j \times \log\left(\frac{b_j}{n}\right)}} \quad (14)$$

NMI is a normalized version of the MI that accounts for the expected MI under a null model. It addresses the limitations of MI, such as its sensitivity to the number of clusters and the problem of excessive agreement due to chance. To calculate NMI, the algorithm first computes the MI between the true cluster labels and the predicted cluster assignments. MI measures the amount of shared information between the two sets of labels. It indicates how much knowledge of one set can reveal about the other set.

NMI is a metric used to assess the degree of accuracy of clustering. Just when both sets of identifiers exactly match does it achieve a maximum score of 1. NMI has been employed in several studies to analyze data stream clustering algorithms [7, 9, 20, 38].

Figure 8 shows the results of NMI across various datasets. As shown in Figure 6, BOCEDS-ATI achieved outstanding results over the NMI median, with approximately 0.32 on the DNA and 0.26 on the Spiral dataset, which outperformed the benchmark, and a median of approximately 0.23 on the KDD dataset, which outperformed the benchmark of MuDi-Stream, BOCEDS Inc, and BOCEDS. The only dataset that resulted in BOCEDS-ATI performing poorly is Landsat. In a comparable manner, every technique has been unable to provide satisfactory scores of NMI using Landsat dataset, as perceived due to the structure of the dataset.

Figure 8
NMI evaluation over KDDCup'99, Landsat, Spiral, and DNA datasets



7.2. Quality evaluation

While the proposed BOCEDS-ATI algorithm demonstrated superior performance across most datasets, a noticeable performance decline was observed with the Landsat dataset. This behavior can be attributed to several inherent characteristics of the dataset and to the design limitations of the algorithm.

First, the Landsat dataset exhibits high feature dimensionality and low temporal variation, where most data points are relatively static and not strongly influenced by evolving temporal behavior. Since BOCEDS-ATI's ATI mechanism is optimized for evolving and dynamic data streams where clusters move or drift over time, the limited temporal evolution in Landsat offers fewer opportunities for the ATI to adaptively differentiate overlapping clusters.

Second, the Landsat data distribution is highly imbalanced and noise-sensitive, with several overlapping classes that are weakly separable in the feature space. The velocity-based component of the ATI mechanism assumes a relatively smooth trajectory of cluster evolution; hence, in datasets with irregular or abrupt noise patterns, such as Landsat, the velocity estimation may lead to suboptimal link updates.

Third, the Landsat features contain non-spherical and heterogeneous cluster shapes, which are not easily separable using Euclidean distance, the metric used in BOCEDS-ATI. The use of Euclidean distance can limit the model's sensitivity to curved or manifold-shaped clusters, which partially explains the reduced NMI and ARI scores.

These findings collectively suggest that while BOCEDS-ATI effectively handles dynamic and velocity-driven clusters, its performance can degrade in datasets with static, irregular, or non-temporal patterns. Future work could address these challenges by incorporating adaptive distance metrics or multi-kernel similarity functions to improve robustness across diverse data types.

7.3. Efficiency evaluation

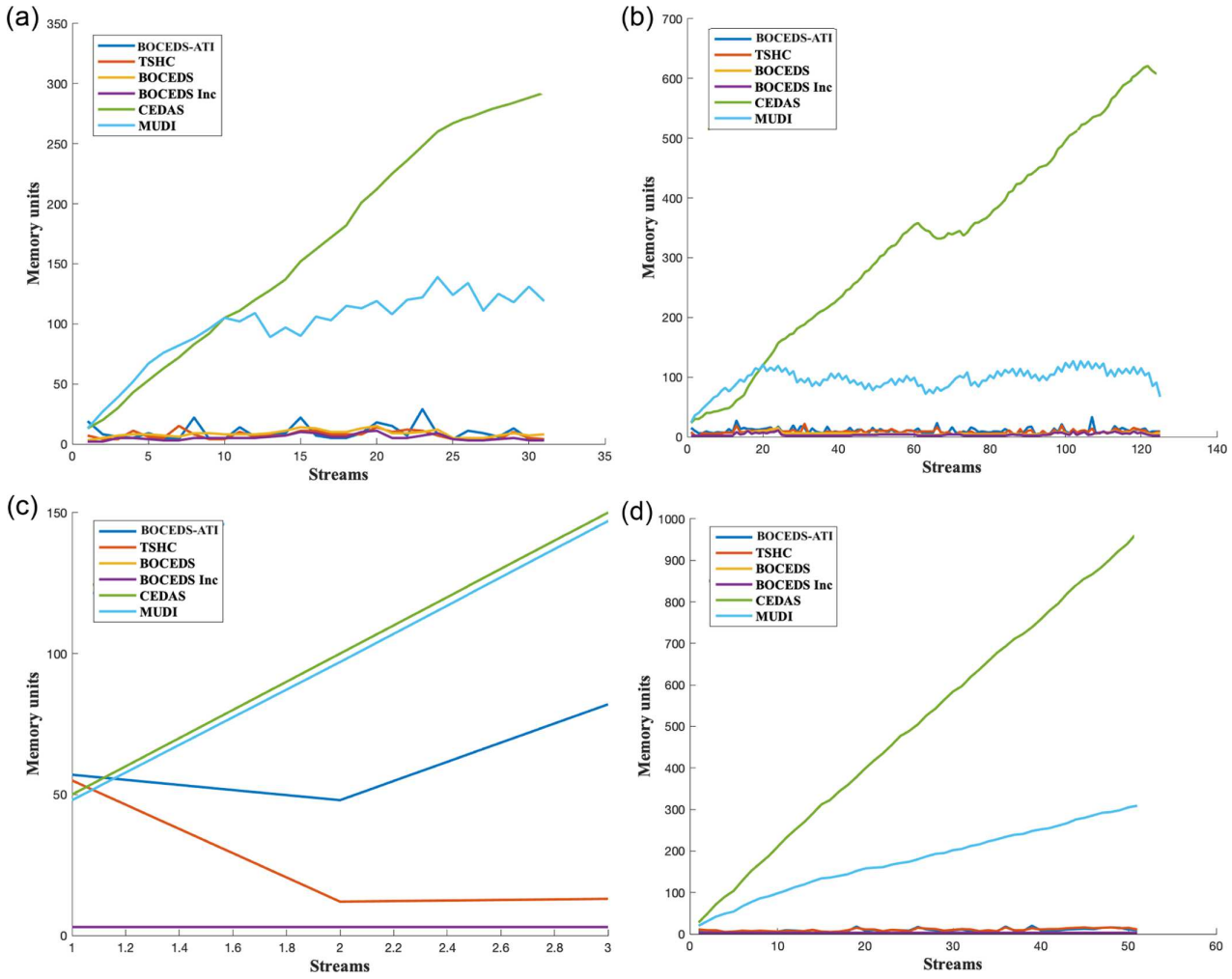
Clustering efficiency evaluation focuses on assessing the computational cost and memory usage of clustering algorithms. It aims to measure the efficiency and performance of clustering techniques in terms of execution time and memory usage. Evaluating clustering efficiency is crucial for selecting appropriate algorithms that can handle the computational demands of clustering tasks efficiently. The efficiency evaluation employed to evaluate the proposed BOCEDS-ATI algorithm includes the following:

7.3.1. Memory usage

The memory utilization is determined by the total number of core-mini clusters formed by every algorithm. Memory usage evaluation is an essential aspect of assessing the efficiency and effectiveness of algorithms, systems, or software applications [19]. It involves measuring and analyzing the amount of memory or storage space required by a program or system to perform its tasks. The evaluation of memory usage is particularly important when dealing with large datasets, complex computations, or resource-constrained environments. It helps identify potential memory-related issues such as excessive memory consumption, memory leaks, or inefficient memory management [17].

As shown in Figure 9, the experimental results demonstrate that the BOCEDS-ATI algorithm utilizes considerably less memory compared to the CEDAS and MuDi-stream algorithms across all datasets. Conversely, BOCEDS and BOCEDS Inc show slightly lower memory usage in both KDD and Spiral datasets.

Figure 9
Memory usage evaluation over: (a) DNA data, (b) Spiral data, (c) Landsat data, and (d) KDD Cup'99 data



The reason for that is that the developed BOCEDS-ATI algorithm has a buffer that serves as a temporary repository for the data, after which it is going to be erased to clear up the memory storage.

7.3.2. Computational cost

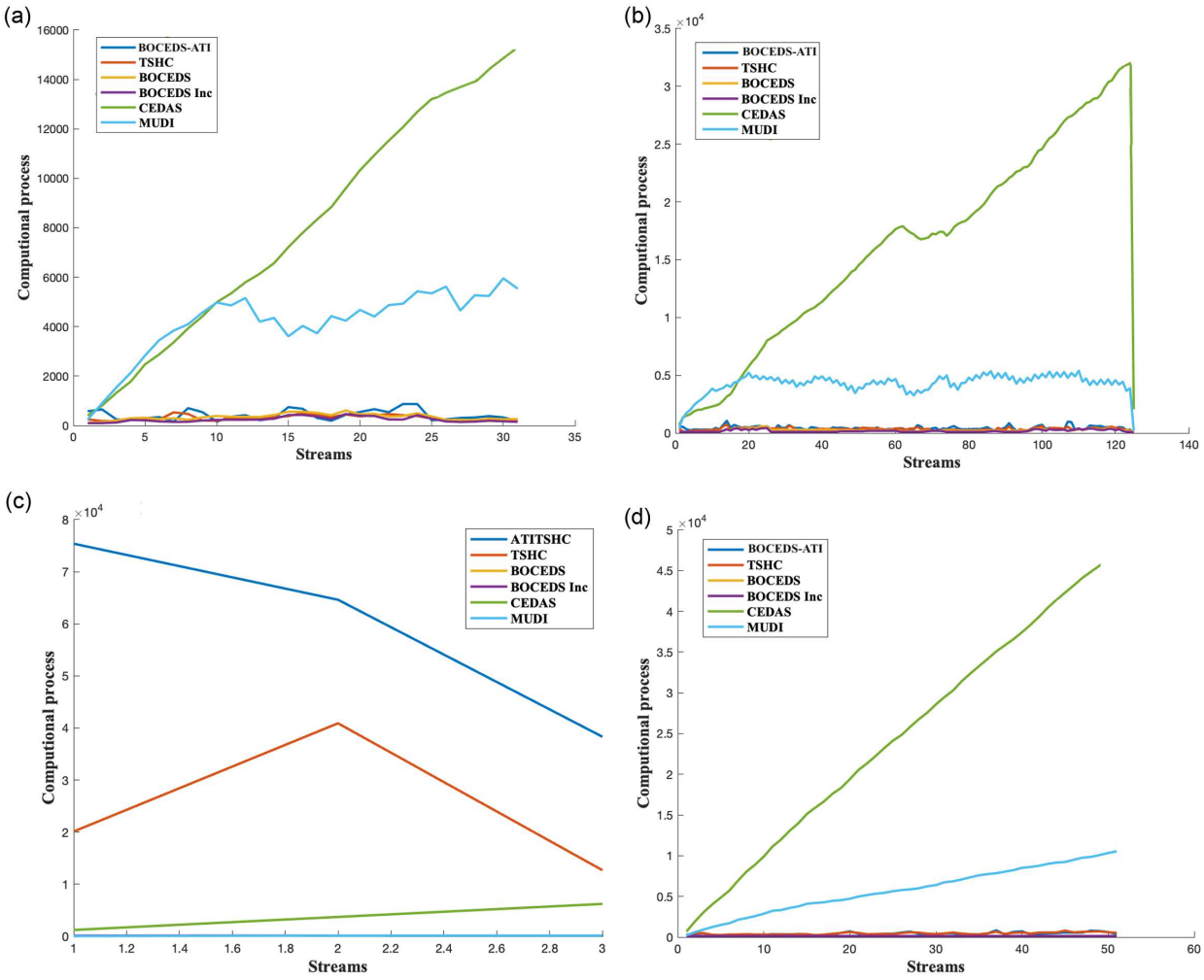
Computational cost evaluation is a fundamental aspect of assessing the efficiency and performance of algorithms, systems, or software applications. It involves measuring and analyzing the computational resources, such as processing time or CPU usage, required to execute a program or perform a specific task [17]. The evaluation of computational cost is essential for several reasons. First, it helps identify bottlenecks and areas of inefficiency in the code or algorithm, allowing for optimization and improvement. Second, it enables the comparison and selection of different algorithms or approaches based on their computational efficiency. Third, it assists in resource planning, capacity estimation, and scalability considerations [19].

The results demonstrated in Figure 10 compare the computational cost of the proposed BOCEDS-ATI algorithm in comparison with Mudi-Stream algorithms, CEDAS, BOCEDS, and BOCEDS Inc. As illustrated in Figure 10, the developed BOCEDS-ATI algorithm shows lower computational cost in all the datasets used in comparison to both CEDAS and Mudi-Stream, which reported high computational costs. BOCEDS reported the lowest computational cost when used with KDD and Spiral datasets, whereas the developed BOCEDS-ATI algorithm reported less computational cost when using the DNA dataset.

8. Findings and Summary

Cluster's false merging refers to a situation in clustering analysis where two or more distinct clusters are incorrectly merged into a single cluster. It occurs when there is an overlap or proximity between clusters, leading to misinterpretation and misrepresentation of the underlying data structure. False merging can significantly impact the accuracy and reliability of clustering results, as it distorts the true boundaries and relationships between data points. The consequences of cluster's false merging can be detrimental, as it can lead to incorrect interpretations and decisions based on clustering results. It can affect various applications, such as customer segmentation, anomaly detection, pattern recognition, and data exploration. To mitigate the risk of false merging, researchers and practitioners employ various strategies, including the development of advanced clustering algorithms, refinement of distance metrics, application of outlier detection techniques, and utilization of validation measures to evaluate clustering quality.

Figure 10
Computational cost evaluation over: (a) DNA data, (b) Spiral data, (c) Landsat data, and (d) KDD Cup'99 data



The proposed BOCEDS-ATI algorithm is an online algorithm for clustering evolving data streams with the ability to prevent clusters' false merging that occurs when data points from different clusters are in proximity or exhibit similar characteristics; it becomes challenging for clustering algorithms to accurately separate them. This overlap can result in the merging of clusters, as the algorithm may incorrectly interpret the shared characteristics as indicative of a single cluster using an adaptive time interval called ATI. The objective of the ATI technique is to generate high-quality clusters to avoid false merging when overlapping occurs between two or more clusters. This technique operates online and utilizes the Euclidean distance function to calculate velocities for binary class data, as demonstrated in the pseudocode provided in the previous sections of the paper. This works by selecting a reference point and then calculating the relative speed with respect to the speed threshold. If the relative speed exceeds the speed threshold, split the clusters into two clusters; otherwise, merge the clusters.

In general, the experiments indicate that the BOCEDS-ATI algorithm effectively addresses the problem of cluster false merging caused by overlapping clusters by enabling a dynamic calculation of the data points. Based on the evaluation performed, the developed BOCEDS-ATI algorithm demonstrated superior performance compared to the baseline clustering algorithms (BOCEDS, BOCEDS Inc, CEDAS, and Mudi-Stream) in most of the evaluation matrices used (ADR, ARI, NMI, and RI) using five datasets (DNA, Spiral, Landsat, KDD, and Traffic). Similarly, the efficiency assessment of the BOCEDS-ATI algorithm that was developed shows that it utilizes minimal memory and incurs a low computational cost.

9. Limitations

Although the proposed BOCEDS-ATI algorithm demonstrates strong performance across a range of evolving data streams, several limitations should be acknowledged to provide a balanced perspective on its applicability.

1) High-dimensional sparse data: In datasets with very high dimensionality or sparsity, the Euclidean distance metric used for cluster proximity and velocity estimation can become less informative due to the "curse of dimensionality." This may reduce ATI's ability to accurately track temporal proximity, leading to less stable cluster boundaries. Dimensionality reduction or feature selection could mitigate this effect.

2) Velocity noise sensitivity: The ATI mechanism assumes that cluster movement trajectories are relatively smooth over time. In cases where velocity vectors are highly erratic—such as in extremely noisy or chaotic streams—the adaptive interval may fluctuate excessively, causing premature or delayed merging decisions. Integrating noise-aware smoothing (e.g., exponential decay or Kalman filtering) could improve stability in such environments.
3) Misleading velocity-based separation: In scenarios where distinct clusters move in parallel at similar velocities, the relative velocity component may underestimate their divergence, potentially leading to false merging. This highlights a limitation of using motion similarity as a primary indicator of cluster relationships without incorporating spatial density context.
4) Non-Euclidean or curved cluster structures: As noted in earlier experiments, ATI's reliance on Euclidean distance limits its ability to handle complex, non-spherical, or manifold-shaped clusters. Future research could extend ATI to incorporate adaptive distance metrics (e.g., Mahalanobis or kernel-based measures) or multi-view learning to handle diverse cluster geometries more effectively.

Overall, while ATI enhances temporal adaptivity and prevents false merging in dynamic environments, it remains best suited to moderately dimensional, smoothly evolving data streams. Addressing these limitations provides clear directions for future extensions toward more general and noise-robust clustering frameworks.

10. Conclusion

Overall, this research contributes to the advancement of online clustering algorithms and provides a valuable tool for analyzing and understanding evolving data streams. The ability to prevent false merging and maintain accurate clusters opens new possibilities for real-time analysis and decision-making in the face of rapidly changing data. Hence, an online clustering algorithm was developed to handle evolving data streams with the ability to prevent clusters' false merging using ATIs called BOCEDS-ATI. Clusters' false merging arises when two or more clusters overlap or coincide with each other. The avoidance of false merging of clusters improves the quality performance of the clustering algorithm. The algorithm's effectiveness in detecting and preventing false merging in evolving data streams is attributed to its use of a time interval and minimum links. It accomplishes this without merging unrelated clusters, which can be a common issue when dealing with evolving data. Yet, false merging was not considered in most of the existing algorithms. Few algorithms however did attempt to address the issue such as CEC-Merge [22]. However, the CEC-Merge algorithm has a significant computational overhead due to the need to perform many distance function calls. This becomes especially evident when working with high-dimensional data, as the calculations of distances between data points and the centers of CMC or outliers become increasingly intricate. Hence, an ATI technique is developed for preventing the false merging of overlapped moving clusters, which records the historical behavior of the data stream. However, this research presents certain research challenges that need to be addressed in the future. One potential extension of the developed BOCEDS-ATI algorithm could involve handling false merging within clusters that have curved shapes.

Funding Support

Funded by NSERC (Natural Sciences and Engineering Research Council of Canada) Discovery Grant Number RGPIN-2020-04325.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data are available from the corresponding author upon reasonable request.

Author Contribution Statement

Redhwan Al-amri: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Gamal Alkawsi:** Validation, Project administration. **Abdulnaser A. Hagar:** Writing – review & editing. **Luiz Fernando Capretz:** Supervision, Funding acquisition.

References

[1] Al-amri, R., Murugesan, R. K., Almutairi, M., Munir, K., Alkawsi, G., & Baashar, Y. (2022). A clustering algorithm for evolving data streams using temporal spatial hyper cube. *Applied Sciences*, *12*(13), 6523. https://doi.org/10.3390/app12136523

[2] Al-Amri, R., Murugesan, R. K., Alshari, E. M., & Alhadawi, H. S. (2022). Toward a full exploitation of IoT in smart cities: A review of IoT anomaly detection techniques. In *Proceedings of International Conference on Emerging Technologies and Intelligent Systems (ICETIS 2021 Volume 2)*, 193–214. https://doi.org/10.1007/978-3-030-85990-9_17

[3] Babu, C. V. S., Moorthy, G. A. V., Lokesh, S., Niranjan, A. K., & Manivannan, Y. (2025). Unleashing IoT data insights: Data mining and machine learning techniques for scalable modeling and efficient management of IoT. In D. S. Rajput, G. S. Lalotra, V. Kumar, P. Kumar, & H. Patel (Eds.), *Scalable modeling and efficient management of IoT applications* (pp. 153–188). IGI Global Scientific Publishing. https://doi.org/10.4018/979-8-3693-1686-3.ch008

[4] Mohanty, S., Jain, C., Bharadwaj, R., & Pattnaik, P. K. (2025). Embedded and cloud computing for ingesting big multimedia data in IOT sensor networks applications. In *Multimedia Technologies in the Internet of Things Environment*, *Volume 4*, 95–111. https://doi.org/10.1007/978-981-96-4356-1_5

[5] Alkawsi, G., Al-amri, R., Baashar, Y., Ghorashi, S., Alabdulkreem, E., & Kiong Tiong, S. (2023). Towards lowering computational power in IoT systems: Clustering algorithm for high-dimensional data stream using entropy window reduction. *Alexandria Engineering Journal*, *70*, 503–513. https://doi.org/10.1016/j.aej.2023.03.008

[6] Amini, A., Saboohi, H., Herawan, T., & Wah, T. Y. (2016). MuDi-Stream: A multi density clustering algorithm for evolving data stream. *Journal of Network and Computer Applications*, *59*, 370–385. https://doi.org/10.1016/j.jnca.2014.11.007

[7] Arowoiya, V. A., Oke, A. E., Aigbavboa, C. O., & Aliu, J. (2020). An appraisal of the adoption internet of things (IoT) elements for sustainable construction. *Journal of Engineering, Design and Technology*, *18*(5), 1193–1208. https://doi.org/10.1108/JEDT-10-2019-0270

[8] Arthi, R., Priscilla, G. M., Maidin, S. S., & Yang, Q. (2025). Optimizing survival prediction in children undergoing hematopoietic stem cell transplantation through enhanced chaotic Harris hawk deep clustering. *Journal of Applied Data Sciences*, *6*(1), 405–415. https://doi.org/10.47738/jads.v6i1.468

[9] Barbosa Roa, N., Travé-Massuyès, L., & Grisales-Palacio, V. H. (2019). DyClee: Dynamic clustering for tracking evolving environments. *Pattern Recognition*, *94*, 162–186. https://doi.org/10.1016/j.patcog.2019.05.024

[10] Bezerra, C. G., Costa, B. S. J., Guedes, L. A., & Angelov, P. P. (2020). An evolving approach to data streams clustering based on typicality and eccentricity data analytics. *Information Sciences*, *518*, 13–28. https://doi.org/10.1016/j.ins.2019.12.022

[11] Bhattacharjee, P., & Osborn, W. (2026). A spatial data stream ensemble for prediction of emergency service priorities. In *Advances in Networked-Based Information Systems: The 28th International Conference on Network-Based Information Systems (NBiS-2025)*, *Volume 1*, 244–257. https://doi.org/10.1007/978-3-032-06161-4_24

[12] Islam, M. R., Kabir, M. M., Mridha, M. F., Alfarhood, S., Safran, M., & Che, D. (2023). Deep learning-based IoT system for remote monitoring and early detection of health issues in real-time. *Sensors*, *23*(11), 5204. https://doi.org/10.3390/s23115204

[13] Zizic, M. C., Mladineo, M., Gjeldum, N., & Celent, L. (2022). From Industry 4.0 towards Industry 5.0: A review and analysis of paradigm shift for the people, organization and technology. *Energies*, *15*(14), 5221. https://doi.org/10.3390/en15145221

[14] Masud, N., Chowdhury, P., Al Bassam, A., Sourav, M. T. I., & Sobhan, A. (2025). A machine learning and IoT-driven framework for real-time disaster management. In M. Azrour, I. Hossain, & A. K. M. M. Haque (Eds.), *Addressing environmental challenges with AI, robotics, and augmented reality* (pp. 247–288). IGI Global Scientific Publishing. https://doi.org/10.4018/979-8-3373-1892-9.ch010

[15] Connelly, A. C., Zaidi, S. A. R., & McLernon, D. (2023). Autoencoder and incremental clustering-enabled anomaly detection. *Electronics*, *12*(9), 1970. https://doi.org/10.3390/electronics12091970

[16] Ikotun, A. M., Ezugwu, A. E., Abualigah, L., Abuhaija, B., & Heming, J. (2023). K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, *622*, 178–210. https://doi.org/10.1016/j.ins.2022.11.139

[17] Han, W., Tian, Z., Shi, W., Huang, Z., & Li, S. (2019). Low-power distributed data flow anomaly-monitoring technology for industrial Internet of Things. *Sensors*, *19*(12), 2804. https://doi.org/10.3390/s19122804

[18] Harush, S., Meidan, Y., & Shabtai, A. (2021). DeepStream: Autoencoder-based stream temporal clustering and anomaly detection. *Computers & Security*, *106*, 102276. https://doi.org/10.1016/j.cose.2021.102276

[19] Hassan, A. A., & Hassan, T. M. (2022). Real-time big data analytics for data stream challenges: An overview. *European Journal of Information Technologies & Computer Science*, *2*(4), 1–6. https://doi.org/10.24018/compute.2022.2.4.62

[20] Hyde, R., Angelov, P., & MacKenzie, A. R. (2017). Fully online clustering of evolving data streams into arbitrarily shaped clusters. *Information Sciences*, *382–383*, 96–114. https://doi.org/10.1016/j.ins.2016.12.004

[21] Islam, M. K., Ahmed, M. M., & Zamli, K. Z. (2019). A buffer-based online clustering for evolving data stream. *Information Sciences*, *489*, 113–135. https://doi.org/10.1016/j.ins.2019.03.022

[22] Islam, M. K., Ahmed, M. M., & Zamli, K. Z. (2019). i-CODAS: An improved online data stream clustering in arbitrary shaped clusters. *Engineering Letters*, *27*(4), 752–762.

[23] Liu, Z., & He, X. (2024). Dynamic submodular-based learning strategy in imbalanced drifting streams for real-time safety assessment in nonstationary environments. *IEEE Transactions on Neural Networks and Learning Systems*, *35*(3), 3038–3051. https://doi.org/10.1109/TNNLS.2023.3294788

[24] Maia, J., Severiano, C. A., Guimarães, F. G., de Castro, C. L., Lemos, A. P., Fonseca Galindo, J. C., & Weiss Cohen, M. (2020). Evolving clustering algorithm based on mixture of typicalities for stream data mining. *Future Generation Computer Systems*, *106*, 672–684. https://doi.org/10.1016/j.future.2020.01.017

[25] Miloslavskaya, N., & Tolstoy, A. (2019). Internet of Things: Information security challenges and solutions. *Cluster Computing*, *22*(1), 103–119. https://doi.org/10.1007/s10586-018-2823-6

[26] Ratasich, D., Khalid, F., Geissler, F., Grosu, R., Shafique, M., & Bartocci, E. (2019). A roadmap toward the resilient Internet of Things for cyber-physical systems. *IEEE Access*, *7*, 13260–13283. https://doi.org/10.1109/ACCESS.2019.2891969

[27] Tareq, M., & Sundararajan, E. A. (2020). A new density-based method for clustering data stream using genetic algorithm. *Technology Reports of Kansai University*, *62*(11), 6557–6572.

[28] Tareq, M., & Sundararajan, E. A. (2021). An evolving approach to data streams clustering based on Chebyshev with false merging. *Journal of Theoretical and Applied Information Technology*, *99*(9), 1955–1965.

[29] Tareq, M., Sundararajan, E. A., Mohd, M., & Sani, N. S. (2020). Online clustering of evolving data streams using a density grid-based method. *IEEE Access*, *8*, 166472–166490. https://doi.org/10.1109/access.2020.3021684

[30] Unver, H. A. (2022). Using social media to monitor conflict-related migration: A review of implications for AI forecasting. *Social Sciences*, *11*(9), 395. https://doi.org/10.3390/socsci11090395

[31] Patidar, S., Kumar, N., & Jindal, R. (2024). IoT data stream handling, analysis, communication and security issues: A systematic survey. *Wireless Personal Communications*, *146*(suppl2), 133. https://doi.org/10.1007/s11277-024-11177-1

[32] Wang, K., Xiong, L., & Xue, R. (2024). Real-time data stream learning for emergency decision-making under uncertainty. *Physica A: Statistical Mechanics and Its Applications*, *633*, 129429. https://doi.org/10.1016/j.physa.2023.129429

[33] Xu, L., Ye, X., Kang, K., Guo, T., Dou, W., Wang, W., & Wei, J. (2020). DistStream: An order-aware distributed framework for online-offline stream clustering algorithms. In *2020 IEEE 40th International Conference on Distributed Computing*

Systems, 842–852. https://doi.org/10.1109/ICDCS47774.2020.00075

[34] Zhu, Y., Zhu, Z., Dai, G., Zhong, K., Yang, H., & Wang, Y. (2022). Exploiting parallelism with vertex-clustering in processing-in-memory-based GCN accelerators. In *2022 Design, Automation & Test in Europe Conference & Exhibition*, 652–657. https://doi.org/10.23919/DATE54114.2022.9774537

[35] Yu, K., Shi, W., & Santoro, N. (2020). Designing a streaming algorithm for outlier detection in data mining—An incremental approach. *Sensors*, *20*(5), 1261. https://doi.org/10.3390/s20051261

[36] Pandey, S. K., Ayyadurai, R., Mohammed, M. H., Prasad, K. R., Alisha, S. A., & Samal, A. (2024). Data mining techniques for customer behavior analysis in e-commerce. In *2024 International Conference on Augmented Reality, Intelligent Systems, and Industrial Automation*, 1–6. https://doi.org/10.1109/ARIIA63345.2024.11051563

[37] Zheng, J., Qu, H., Li, Z., Li, L., & Tang, X. (2022). An irrelevant attributes resistant approach to anomaly detection in high-dimensional space using a deep hypersphere structure. *Applied Soft Computing*, *116*, 108301. https://doi.org/10.1016/j.asoc.2021.108301

[38] Zubaroğlu, A., & Atalay, V. (2021). Data stream clustering: A review. *Artificial Intelligence Review*, *54*(2), 1201–1236. https://doi.org/10.1007/s10462-020-09874-x

How to Cite: Al-amri, R., Alkawsi, G., Hagar, A. A., & Capretz, L. F. (2026). An Online Clustering Algorithm for Handling Evolving Data Streams with the Ability to Prevent Clusters' False Merging Using Adaptive Time Interval. *Journal of Computational and Cognitive Engineering*. https://doi.org/10.47852/bonviewJCCE62026839