Vol.4, No.4, Page 1938- 1949, 2026 https://doi.org/10.59628/jast.v4i4.2271 ISSN 2958-9568 JAST

SUN PUBLISHER

مجلة جامعة صنعاء للعلوم التطبيقية والتكنولوجيا

Sana'a University Journal of Applied Sciences and Technology

https://journals.su.edu.ye/index.php/jast/

A Proposed Incremental Distance Based Clustering of Interesting Patterns for Big Data

Akram A. M. Mustafa *, Abdulmajed A. G. Al-Khulidi and Ahmed S. Al-Hegami

Department of Computer Science, Faculty of Computer Science and Information Technology, Sana'a University, Sana'a, Yemen

*Corresponding author: akram.mustafa@su.eduye

Abstract

Clustering is a fundamental task in unsupervised learning, aiming to group similar data points while maximizing inter-group dissimilarity. Traditional clustering algorithms often struggle with large-scale datasets containing mixed data types (numerical and categorical). This Research introduces an incremental clustering algorithm that incorporates a novelty criterion to identify only interesting patterns from big data. The novelty criterion functions as a distance metric, measuring deviation between instances and labeling an instance as novel if its deviation exceeds a user-defined threshold. Unlike conventional methods such as k-means, the proposed approach does not require predefining the number of clusters; instead, it dynamically determines them during execution. This flexibility enhances its applicability to complex datasets. The algorithm was implemented and evaluated using public datasets, Through experimentation it became clear that the proposed algorithm achieved the same high accuracy of 81.5% as the DBSCAN algorithm. and However it surpassed it in speed, taking only 22 seconds compared to 100 seconds. This means it is more efficient the same level of accuracy but in significantly less time, making it a better option for practical applications.

Article Info

Keywords:
Data mining, incremental clustering, parallel mining, machine learning, novelty measure.

Article History:
Received: 27-December-2025,
Revised: 14-February-2026,
Accepted: 1-March-2026,
Published: 28 April 2026.

1. INTRODUCTION

The rapid expansion of data in various domains has presented significant challenges for traditional clustering algorithms. Clustering algorithms are an important problem in data mining. It is an essential task in unsupervised learning that aims to group similar data points together while maximizing the dissimilarity between different groups [1–4]. However, conventional clustering algorithms often struggle to handle large-scale datasets with mixed data types. The rise of mixed data type datasets and their more influential role in future research plans illustrate the need to pay the utmost attention to these datasets. Mixed data type datasets have multiple attribute types, such as continuous, categorical, binary, ordinal, interval, and nominal. Most clustering algorithms can only consider one type of attribute in a dataset. Despite these efforts, there are no standardized methods for grouping mixed data types, and most studies conduct clustering with a pre-processin technique to deal with mixed data types [5–7].

Furthermore, clustering algorithms typically make the assumption that data is static, which is not true when data is evolving. also, Since the user domain knowledge is monotonically augmented with time, the conventional clustering algorithms not only waste computational, communication and I/O resources but also become ineffective in terms of pattern interestingness criteria [8].

To address these challenges, this research introduces a novel approach that integrates the concept of the novelty criterion into an incremental clustering algorithm specifically designed for mixed data types. By introducing the novelty criterion into an incremental clustering algorithm, this research aims to contribute to the advancement of clustering techniques for mixed data types and provide a robust and efficient solution for analyzing and extracting meaningful patterns from diverse datasets.

The design of efficient incremental clustering algorithms in dynamic environments is an important research topic. Although previous studies have made notable contributions to clustering algorithms for mixed data types, there is still a need for more effective and efficient approaches that can handle evolving datasets and incorporate novelty [9, 10]. In this study, we propose a novel incremental clustering algorithm that addresses these challenges by pushing the novelty criterion into the clustering process to handle mixed data types.

The proposed algorithm leverages a similarity measure to account for the unique characteristics of categorical and numerical features. In addition, it captures the inherent relationships and similarities within mixed data incrementally, thereby improving clustering performance. Moreover, the algorithm's incremental nature allows for the efficient updating of the clustering structure as new data points arrive, eliminating the need to reprocess the entire dataset. This feature is particularly advantageous in domains where data streams continuously evolve over time, such as social media analysis and sensor networks.

By incorporating the novelty criterion into the clustering process, the proposed algorithm can effectively identify and handle novel data points without disrupting the existing clustering structure. It provides the flexibility to adapt to evolving datasets, making it suitable for scenarios where new data continuously emerge, such as in streaming or dynamic environments [11, 12]. The contributions of this research can be summarized as follows:

1. Introduction of a novel incremental clustering algorithm tailored for mixed data types.
2. Using the novelty criterion as a similarity measure to handle the heterogeneous nature of mixed data.
3. Dynamic determining of cluster numbers,
4. Integrate of the novelty criterion during mining process to discover only novel clusters.
5. Integration of the novelty criterion to enhance the adaptability of the algorithm to new data points.
6. Evaluation of the proposed algorithm through extensive experiments on diverse real-world datasets.

2. MOTIVATIONS

The motivation for this work arises from the limitations of existing incremental clustering algorithms when dealing with mixed data types. Traditional incremental clustering algorithms typically assume a uniform data type or heavily rely on distance-based measures, which may not be suitable for datasets with heterogeneous features [12, 13]. By pushing the novelty criterion into the incremental clustering process, we aim to augment the algorithm's ability to identify and adapt to novel data patterns, regardless of the data type.

3. PROBLEM STATEMENTS

The research addresses the problem of clustering mixed data types, in which both categorical and numerical features are present. The goal is to develop an incremental clustering algorithm that can handle evolving datasets and incorporate the novelty criterion as a distance metric to adapt to new and unseen data points.

4. RELATED WORKS

Earlier research focused on novelty measures applied after the clusters were constructed, whereas subsequent work introduced a novelty measure as a distance metric that was used during the cluster-building process for an incremental clustering algorithm. Given these fundamental differences, we believe that these related works do not align closely with our research focus.

Several researches deal with the problem of incremental clustering and address challenges such as real-time processing, adaptability to evolving clusters, and scalability. In [14], an online K-means clustering algorithm that can generate approximately O(k) clusters with a K-means cost of approximately O(W^*) is proposed. Experimental results show that the algorithm performs similarly to k means++ in a more constrained computational model. In [15], an incremental clustering algorithm called Streaming K-means++ which extends the K-means++ algorithm to handle data streams is proposed. It maintains a representative set of centers called "coresets" that adaptively captures the evolving clusters. The algorithm incrementally updates the coreset as new data arrives, allowing for efficient and scalable stream clustering. In [16], DenStream is presented which is an incremental clustering algorithm designed specifically for data streams. It employs a density-based approach to identify micro-clusters that represent the evolving clusters. It dynamically updates the micro-clusters as new data arrives, allowing for efficient and scalable stream clustering. In [17, 18], CluStream and DenStream algorithms that are proposed. They are incremental clustering algorithm designed to handle data streams with concept drift. Density-based approaches using micro-clusters to capture the evolving clusters in the stream are employed. DenStream dynamically adjusts the micro-clusters based on the arrival of new data and the concept drift detection. The algorithms provide flexibility in handling different density distributions and is suitable for real-time stream clustering. In [19], MuDi-Stream is a density-based clustering algorithm for evolving data streams that improves clustering quality in multi-density environments. It does this by keeping summary information about the evolving data stream in the form of core mini-clusters in the online phase. In the offline phase, the algorithm uses a modified density-based clustering algorithm to generate the final clusters. In [20], a new incremental density-based clustering algorithm has been proposed that uses Non-dominated Sorting

Genetic Algorithm II (NSGA-II) to improve clustering precision. The algorithm adjusts the two input parameters (MinPts and Eps) iteratively using fitness functions, and the optimization process is parallelized to IJISAE, 2024, 12(4), 4668 - 4681 | 4669 International Journal of Intelligent Systems and Applications in Engineering improve efficiency. This results in a significant speed-up compared to the serial version of the algorithm. In [21], a clustering approach called SyncTree for evolving data is proposed. It works by maintaining the entire micro-clusters at several levels of granularity. It summarizes the constantly arriving data points sequentially in a batch way. This allows us to examine the structure of cluster between any two time stamps in the earlier period. In [22], an incremental K-Means for massive multidimensional datasets is presented. It is designed for evolving databases. The algorithm measures the new cluster cenroids by computing the new data from the means of the earlier discovered clusters. This approach is more efficient than rerunning the K-means algorithm, which can be computationally expensive. In [23], m-BIRCH is presented. It is an online clustering algorithm that is designed for incremental clustering large datasets of features used in computer vision. It is efficient because it only uses a fraction of the dataset memory, and it is effective because it can handle varying density regions in the feature space. m-BIRCH is a publicly available clustering tool that can be used to cluster data from a variety of sources. In [24], an incremental clustering which is based on OPTICS algorithm is proposed. It works by ordering the cluster structure that looks like the structure of OPTICS. However, ICA does not require the user to pre-set the parameters ε and MinPts, and it uses a simpler distance measure called Distance. This makes ICA more efficient than OPTICS. Incremental Other recent incremental partitioning clustering algorithms are PAM, Incremental CLARANS, Incremental CHAMELEON whish are the extended versions of PAM, CLARANS and CHAMELEON respectively. These algorithms are efficient for streaming data, can handle large datasets, robust to noise and can handle clusters of arbitrary. However, they are less efficient than Incremental k-means [3, 25]. In [26], an incremental clustering algorithm based on nearness is proposed. The algorithm does not comprises the quality of data. The main feature of this algorithm is its ability to update the cluster incrementally and saving computing complexity. In [27], a clustering approach is proposed. it makes use of an incremental clustering method and a pairwise clustering method to reduce the dimension of the dataset first and subsequently. clustering the dataset to a predetermined number of clusters. the approach is tested using large number of documents and the results shown promising. In [7], an incremental clustering algorithm, based on a new distance measure is proposed. It deals with both numerical and categorical attributes. The incremental clustering is performed in two stages, In the first stage, the traditional k-means is engaged for initial clustering of the static data set. In the second stage, the distance measure is used to generate the appropriate cluster for the incremental data points. The results of evaluation of the approach shows improving the accuracy and the computational time of clustering. In [28, 29], a parallel k-means clustering algorithm based on MapReduce is proposed. It efficiently processes large datasets on commodity hardware. In [30], A novel hybrid clustering algorithms is proposed. It is based on incremental clustering and initial selection to tune up Fuzzy C-Means (FCM) for the Big Data problem. In [31], a parallel and incremental clustering algorithm called pi-Lisco is proposed. It uses sliding windows to clusters point-cloud data in. It reuses the clusters from overlapping portions of the data to facilitate single-window processing. By combining these key ideas, pi-Lisco achieves significant performance improvements over the state-of-the-art algorithms. In [32], Stream SW is proposed. It is a density-based clustering algorithm for data stream over a sliding window. It consists of two-steps approach to discover clusters first clusters and then refines the clusters in the second step. This approach exhibits better speed and quality than current approaches. In [33], a parallel incremental k-means clustering technique is proposed. It clusters the data into a set of k clusters, and then using a forecasting method to predict the future values of each cluster. The clustering is performed incrementally, so that new data can be added to the model without having to re-cluster the entire dataset. In [34], a parallel incremental density based clustering algorithm called (IncAnyDBC) is introduced. It works by partitioning the data into a set of partitions. Then, the algorithm clusters each partitions in parallel and finally, the clusters from different partitions are merged to form the final clusters. In [35], a modified version of k-means for mining big data under Hadoop parallel approach is introduced. It works by partitioning the data into a set of partitions, and clusters are generated for each partition in parallel manner. The results from the individual partitions are then merged to form the final clustering solution. In [36], an algorithm is proposed to enhance the incremental DBSCAN clustering algorithm. It works by reducing the search space rather than the entire dataset. It builds and updates the shaped clusters in large datasets. In [37], an incremental k-means clustering algorithm is proposed for large multidimensional datasets in dynamic IJISAE, 2024, 12(4), 4668 - 4681 | 4670 International Journal of Intelligent Systems and Applications in Engineering environments where data may be frequently updated. This approach measures the new cluster centers by directly computing them from the means of the existing clusters, instead of rerunning the k-means algorithm from scratch. In [38], a parallel clustering approach of high dimensional data is proposed. It fits well for interactive online clustering and facilitates incremental clustering because chunks of

instances are clustered as separate sets. Subsequently, the results are merged with present clusters. In [13], an incremental anomaly detection approach is proposed that is based on Gaussian Mixture Model (GMM) to identify regular patterns and discover outliers in from flight data. The presented approach update its clusters incrementally based on new data, rather than running the clustering algorithm from scratch. This makes it more efficient and scalable for clustering large and streaming data. In [39], a Research presents a modified streaming k-means algorithm that is dynamic, incremental, and can take into account long-term patterns of data. Compared to streaming k-means clustering, the modified streaming k-means clustering has better convergence ability and more stable results. An incremental clustering algorithm called LIMBO (Learning Incremental Model- Based Online Clustering) is proposed. It uses a model-based approach to cluster the data points and updates the clustering model as new data points arrive and adjusts the clustering accordingly. However, its weakness is that it may not be suitable for datasets with varying densities. The proposed algorithm is based on the idea that patterns discovered in one dataset are likely to be similar to patterns discovered in other datasets, to varying degrees. This means that the algorithm can learn from previous datasets and use that knowledge to find patterns in new datasets. Subsequently, the model is continuously updated to reflects the changing data and user beliefs which makes the overall mining process more effective.

5. THE PROPOSED APPROACH

In this research, we address the issue of model maintenance in an environment comprising evolving data, changing user beliefs, and interestingness criteria. This issue is addressed in the context of a clustering algorithm. We propose an approach for incremental mining that efficiently discovers interesting clustering patterns from big data. The novelty measure proposed in [11, 12] is integrated with a clustering algorithm to form a constraint for the algorithm to discover novel clusters. As the clusters are built, the data points similar to known patterns are dynamically pruned. The advantage of pushing such a criterion is that the size of the clusters discovered can be reduced, maintaining up-to-date clusters, and the discovered knowledge reflects the user's requirement on novelty in an environment where the training set evolves over time. The following subsections explain the proposed approach.

5.1. Architecture of the proposed Approach

Figure 1 shows the environment in which the proposed approach operates. At time t_{i+1}, database D_{i+1} is preprocessed and subjected to a clustering algorithm. The algorithm considers the existing model MT_i representing the known patterns. The algorithm computes the similarity based on the novelty measure with respect to the existing model MT_i and prunes uninteresting instances that are not significant in the current dataset. Subsequently, it assigns only those instances that are likely to lead to novel patterns. For each cluster, rules are extracted and reported to the user, and the model MT_i is updated, resulting in model MT_{i+1} which is used by the prediction algorithm to predict unseen instances.

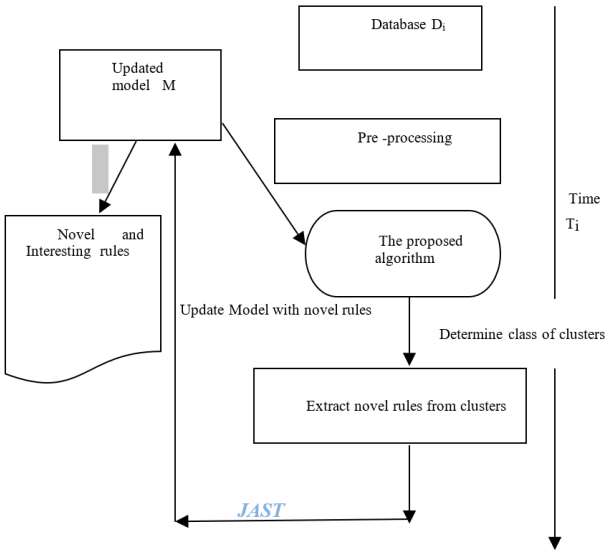

Figure 1. General Architecture of the proposed approach

5.2. Novelty criterion as a distance metric

One of the most important tasks in data clustering is to decide which metric should be used to calculate the distance between each data point. In various real life datasets where cluster analysis is commonly used, such as in biology, social sciences, or marketing surveys, datasets with both quantitative and categorical variables are often applied. This type of data is referred to as mixed data.

The proposed algorithm integrates the novelty criterion into the clustering process to enhance its adaptability. The novelty criterion measures the degree of dissimilarity between mixed data types and existing clusters. By considering the novelty criterion, the algorithm can identify and assign new data points to appropriate clusters or create new clusters if necessary.

Formal Definition of the Novelty Measure (Ψ) :Let each instance S be represented as a vector of attributes $(a_1, a_2, \ldots, a_m)$, where each attribute may be numerical or categorical.

The novelty measure between two instances S_1 and S_2 is defined as:

$$\Psi(S1, S2) = \frac{1}{m}\sum_{i=1}^{m} w_i \cdot \delta(a_{i1}, a_{i2}) \quad (1)$$

where:

- w_i = weight assigned to attribute i (reflecting its importance)
- $\delta(a_{i1}, a_{i2})$ = attribute-level deviation

For Numerical Attributes

$$\delta(a_{i1}, a_{i2}) = \frac{|a_{i1} - a_{i2}|}{\max(a_i) - \min(a_i)}$$

For Categorical Attributes

$$\delta(a_{i1}, a_{i2}) = \begin{cases} 0 & \text{if } a_{i1} = a_{i2} \\ 1 & \text{otherwise} \end{cases}$$

An instance S_2 is considered **novel** with respect to S_1 if:

$$\Psi(S_1, S_2) > \theta$$

where θ is a user-defined novelty threshold.

This formalization ensures that the novelty measure can handle mixed data types by combining normalized numerical deviations and categorical mismatches into a unified distance-like metric.

Because any instance is considered to be a set of conjuncts (where each conjunct represents an attribute, operator, and value), the deviation Ψ between S_1 (old data) and S_2 (new data) is computed as:

$$\Psi(S_1, S_2) = \frac{1}{|K|}\sum_{i=1}^{|K|} \delta(c_i^{(1)}, c_i^{(2)})$$

Where:

- Ψ = deviation scale between two points (novelty score)
- S_1 = existing or old data instance
- S_2 = new data instance
- K = the set of compatible conjunct pairs between S_1 and S_2
- $c_i^{(1)}$ and $c_i^{(2)}$ = the i-th pair of compatible conjuncts

An instance is considered **novel** if:

$$\Psi(S_1, S_2) > \theta$$

By applying this novelty criterion, the algorithm can identify and assign new data points to appropriate clusters. To use the novelty measure as a distance measure in clustering, it is necessary to ensure that it satisfies the properties of a metric, especially if clustering algorithms that require a proper metric space (such as k-means or hierarchical clustering) are considered. The key properties for the novelty measure to be considered a distance metric are

i) **Non-negativity:** $\Psi(S_1, S_2) \geq 0$
ii) **Identity of indiscernibles:** $\Psi(S_1, S_2) = 0$ if and only if $S_1 = S_2$
iii) **Symmetry:** $\Psi(S_1, S_2) = \Psi(S_2, S_1)$
iv) **Triangle inequality:** $\Psi(S_1, S_3) \leq \Psi(S_1, S_2) + \Psi(S_2, S_3)$

Analysis of the semi-metric nature of Ψ:

Although the novelty measure Ψ satisfies the first three properties of a metric— non-negativity, identity of indiscernibles, and symmetry, it may violate the triangle inequality when contextual or user-defined weighting is introduced. This violation does not undermine its usefulness; instead, it provides flexibility for modeling subjective or domain-specific deviations that cannot be captured by strict metric assumptions.

In practical terms, this semi-metric behavior allows Ψ to emphasize meaningful novelty even when transitive similarity does not hold. For example, two instances, A and B, may both be similar to a reference instance C but differ significantly from each other because of contextual weighting. Such behavior is desirable in novelty detection, where the goal is to highlight deviations rather than enforce geometric consistency.

To accommodate this property, the proposed algorithm employs density and deviation-based grouping rather than relying on metric-space clustering. This ensures that Ψ remains computationally efficient and theoretically consistent while capturing user-driven novelty patterns.

However, $\Psi(S_1, S_2)$ does not satisfy the triangular inequality in the case of CV-deviation, where user subjectivity is captured. For this reason, the novelty measure can be considered semi-metric. Many clustering algorithms can work with semi-metrics (which satisfy all metric properties except the triangle inequality), or even more general distance functions. For example, DBSCAN and OPTICS are density-based clustering algorithms that only require a notion of distance for determining the neighborhood of points and do not require the triangle inequality. By using the novelty measure as the distance metric, the clustering algorithm groups together instances that are similar based on their deviation from the known knowledge base. Instances that are considered novel (having a high novelty score) are assigned to different clusters, whereas instances that are more similar to the known knowledge base are pruned.

5.3. Dynamic Pruning Based on Novelty Criterion

In this research, we develop a dynamic pruning approach that reduces the number of accesses to the data. Consequently, our modified algorithms can process datasets with millions of objects in a fraction of the time required by standard clustering algorithms. The proposed dy-

namic pruning method is based on a novelty measure for reducing the number of computed distances in clustering algorithms to reduce the number of unnecessary computations of the proximity measure between data objects in distance-based clustering algorithms. Such unnecessary computations present one of the most problematic theoretical and practical issues associated with clustering algorithms, particularly when clustering large datasets. In fact, computed proximity measures are massively used to guide the clustering process in most clustering algorithms, and this generally far outweighs all other computational costs typical to both hierarchical and partitioning clustering algorithms.

Subsequently, the proposed pruning approach reduces the volumes of the discovered patterns. The quantification of novelty is based on the analysis and computation of the deviation of recently discovered patterns with respect to known knowledge, that is, domain knowledge (DK) and previously discovered knowledge (PDK), and the importance that the user gives to different types of deviations. The computed degree of novelty is then compared with the user-specified threshold to report novel patterns to the user.

5.4. Construction of Clusters

In contrast to k-means and other clustering algorithms that require the number of blocks to be predetermined, the proposed approach does not. The algorithm begins with no blocks, and as each new data point x arrives, it is evaluated based on its novelty criterion. If the data point is novel and does not belong to any existing block, it is treated as a new block.

The algorithm tracks existing blocks, centroids, associated data points, and their degree of novelty. When a new data point arrives, its degree of novelty is calculated and compared with the centroids of the existing blocks. The new data point is assigned to the most appropriate block if its degree of novelty falls within a certain range, or a completely new block is created for it. The novelty measure quantifies the difference between a new data point and existing ones. When a new data point is assigned to a cluster, the cluster's properties (e.g., centroid and variance) must be updated. The advantage of using the novelty criterion is that it ignores outliers and abnormalities in the data. Also, maintains only novel patterns within clusters, so no need to create new clusters every time a data point arrives. Figure 2 shows the proposed incremental clustering algorithm using a novelty measure as a distance metric.

The algorithm computes the novelty degree of the new point and compares it with the centroids of the existing clusters and assigns it to the most appropriate cluster. The novelty measure quantifies the difference between a new data point and existing ones. The advantage of using the novelty criterion is that it ignores outliers and abnormalities in the data. Also, maintains only novel patterns within clusters, so no need to create new clusters every time a data point arrives. The new data point is assigned to a cluster if its novelty degree is greater than the novelty threshold determined by the user. Subsequently, the properties of the cluster (e.g., centroid and variance) must be updated. This process continues until all data points have been assigned to clusters. Figure 2 shows the proposed incremental clustering algorithm using a novelty measure as a distance metric.

Input: Stream of data points, Novelty threshold θ
Output: Clusters of data points
1: Initialize an empty list of clusters
2: for each new data point x do
3: if list of clusters is empty then
4: Create a new cluster with x
5: else
6: Calculate Ψ (x, C) for each cluster C
7: Find the cluster C_min with the minimum Ψ (x, C)
8: if Ψ (x, C_min) < θ then
9: Add x to cluster C_min
10: Update C_min's centroid
11: else
12: Create a new cluster with x
13: end if
14: end if
15: Perform cluster maintenance
16: end for
17: Return the list of clusters

Figure 2. A Proposed algorithm of Incremental distance based clustering

5.4.1. Updating the centroids of clusters

The centroid of a cluster represents the center or representative point of the cluster and is crucial for calculating distances and determining cluster assignments. The proposed approach considers the novelty degree of the novel instances within clusters as the centroid of the clusters through a process of calculating the mean of all instances within clusters. After assigning a new instance to an existing cluster or creating a new cluster, the algorithm updates the centroid of the cluster to reflect the inclusion of the new instance. This update ensures that the cluster representation aligns with the characteristics of the instance it contains. Updating the centroids of clusters based on novel patterns within clusters while ignoring outliers and abnormalities can be achieved by

recalculating the mean of the novelty degree of each instance every time a novel instance is assigned to the cluster.

5.4.2. Merging The clusters

In this study, we present a novel approach to clustering that leverages a novelty measure as the primary metric. Unlike many traditional clustering algorithms, our method does not require the number of clusters to be specified in advance. This flexibility results in the generation of a large number of individual clusters from the input data. To consolidate this granular clustering output into a more concise set of meaningful groups, we propose a cluster merging strategy based on the statistical property of variance.

Specifically, we calculate the variance within each cluster for a selected set of features. Clusters with sufficiently low internal variance are then iteratively merged, continuing this process until an optimal tradeoff is reached between the number of final clusters and the variance exhibited within each one. This variance-driven merging technique allows the natural cluster structure of the data to emerge without imposing an arbitrary constraint on the total number of groups. By combining the advantages of novelty-based clustering and variance-based merging, our method provides an effective unsupervised approach for partitioning complex datasets into an interpretable set of meaningful clusters.

Variance is a statistical measure that indicates how spread out a set of numbers is from the mean or average value. It is a fundamental tool in data analysis, providing insights into patterns and trends. It is calculated by taking the average of the squared differences from the mean. The formula for variance is:

$$\text{Variance} = \sum_{i=1}^{n} \frac{(x-\mu)^2}{n} \tag{2}$$

Where:

x is novelty degree of a data point assigned to a cluster.
where μ is the mean of the novelty degree of the data points in a cluster.
n is the total number of data points in the cluster.

The larger the variance, the more spread out the numbers are.

The proposed algorithm iteratively merges pairs of clusters with the smallest distance. After merging the clusters, the structure of the merged cluster (centroid) is updated. Figure shows the proposed variance-based merging algorithm.

```
Input:
    novel data points of clusters and associated centroid, novelty degree
Output:
    Final clusters after merging
Steps:
function compute_variance(cluster):
    # Compute variance within a cluster
    mean_expression = sum(cluster) / len(cluster)
    variance = sum ((x - mean_expression) ^ 2 for x in cluster) / len(cluster)
    return variance
function merge_clusters(clusters):
    while len(clusters) > 1:
        // Initialize minimum variance increase
        min_variance_increase = infinity
        merge_candidates = None
        // Compare all pairs of clusters
        for i in range(len(clusters)):
            for j in range (i + 1, len(clusters)):
                combined_cluster = clusters[i] + clusters[j]
                combined_variance = compute_variance(combined_cluster)
                // Calculate variance increase
                variance_increase = combined_variance - (compute_variance(clusters[i]) + compute_variance(clusters[j]))
                // Update if variance increase is smaller
                if variance_increase < min_variance_increase
    min_variance increase = variance_increase
    merge_candidates = (i, j)
    // Merge the clusters with the smallest variance increase
    i, j = merge_candidates
    merged_cluster = clusters[i] + clusters[j]
    delete clusters[j]
    clusters[i] = merged_cluster
    return clusters
```

Figure 3. The algorithm for Merging the pair of clusters

5.5. Incremental Model

Incremental clustering is designed to handle evolving mixed datasets in an incremental manner by efficiently updating the clustering structure as new data points arrive. This incremental approach eliminates the need to reprocess the entire dataset and ensures scalability. The proposed approach considers the existing model M_i representing known knowledge (DK+PDK), consequently resulting in the discovery of Model M_{i+1}. For each cluster, only novel patterns are extracted and used to update Model M_{i+1}.

In this phase, for each cluster, the class with the majority of the data points in the cluster is determined. This is done by counting the number of data points for each class and then identifying the class with the highest number of data points.

Because the proposed approach deals with mixed data that may contain mixed data types, the following steps are performed:

If the mixed data contains only categorical features, the majority class label within that cluster is determined by counting the frequency of each class label in the categorical features of the data points in the cluster and

selecting the class label with the highest frequency.

If the data is mixed, with both categorical and numerical features, then for each cluster, the categorical features of the data points in the cluster are considered. The frequency of each class label in the categorical features is then counted. The class label with the highest frequency is selected as the most frequent class for that cluster.

After extracting a set of novel rules from each cluster, the rules are used to update model M_{i+1}.

The following are the steps required to build the incremental interesting model:

1. Determine the class label within each cluster.
2. Generation the rules R_i from each cluster.
3. Update Model M_i/M_{i+1}

The algorithm for building the incremental interesting model is presented in Fig. 4. The algorithm takes a set of clusters and existing knowledge (Model M_i) as inputs. It outputs the incremental interesting model (M_{i+1}).

Input:

Data: set of clusters

Model M_i

Output:

Updated incremental interesting model M_{i+1}

Steps:

Step 1: for each cluster, determine the class label

Step 2: extract the rules R_i from each novel cluster

Step 3 :For each rules R_i:

Update Model M_i/M_{i+1}

Figure 4. The proposed incremental interesting Model.

6. EXPERIMENTAL RESULTS AND DECISIONS

All experiments were conducted to ensure reproducibility, transparency, and fair comparison among the evaluated algorithms. The implementation was performed using Python 3.3 integrated with the Hadoop 3.3 parallel framework to efficiently process large-scale data. The computational environment consisted of an Intel ®Xeon ®E5-2620 CPU @ 2.10 GHz, 64GB RAM, and Ubuntu 22.04 LTS. All random seeds were fixed to guarantee consistent results across repeated runs. Four publicly available benchmark datasets—Census, German Credit, Sick, and Heart Disease—were used for the evaluation. Each dataset was treated as time-evolving and divided into three sequential batches (D_1, D_2, and D_3) arriving at times (T_1, T_2, and T_3) to simulate incremental data streams. Numerical attributes were normalized to [0,1], categorical attributes were one-hot encoded, and missing values were imputed using the mean or mode substitution.

To ensure reproducibility, all data preprocessing procedures , algorithm parameters, and experimental configurations

are explicitly described in this manuscript.

The novelty threshold (θ) controlling the detection of new patterns was empirically set to 0.5 after sensitivity analysis, which confirmed stable accuracy within ±2% for $\theta \in [0.3, 0.7]$. For comparative evaluation, the proposed algorithm was benchmarked against Incremental Kmeans++, Incremental DBSCAN, and Incremental Mean Shift under identical datasets and computational conditions. The parameter configurations were standardized as follows:

Incremental Kmeans++: k = 10, maxiterations = 100

Incremental DBSCAN: $\in$ = 0.5, MinPts = 5.

Incremental Mean Shift: bandwidth = 0.4.

Proposed Algorithm: θ = 0.5, max iterations = 100

Each experiment was repeated five times, and the average values of all metrics are reported. The evaluation metrics are defined as follows:

$$A = \frac{N_{\text{correct}}}{N_{\text{total}}} \times 100 \tag{3}$$

$$\text{NDR} = \frac{N_{\text{novel correct}}}{N_{\text{novel total}}} \times 100 \tag{4}$$

$$R = \frac{A - |A_{\text{noise}} - A_{\text{clean}}|}{A_{\text{clean}}} \tag{5}$$

where:

- A = accuracy on clean data
- A_{clean} = accuracy on clean data
- A_{noise} = accuracy on data with 5% random noise
- **NDR** is the novelty detection rate (%)
- **R** measures robustness under 5% random noise
- The execution time **(T)** is recorded in seconds for each dataset

These standardized settings and formal definitions ensure that the reported results are methodologically sound, reproducible, and directly comparable with subsequent analyses presented in the following subsections.

6.1. Evaluation Metrics

To validate the performance and reliability of the proposed incremental clustering algorithm, several quantitative metrics were employed. Each metric is formally defined in the previous section to ensure reproducibility and objective evaluation.

Accuracy: Measures the correctness of data grouping and represents the percentage of instances correctly assigned to their respective clusters. The proposed algorithm achieved an overall accuracy of 81.5%.

Execution time: Recorded in seconds to evaluate computational efficiency and scalability across different datasets.

Novelty detection rate (NDR): This metric quantifies the algorithm's ability to correctly identify new or previously unseen patterns during incremental updates.

Robustness to Noise: Assesses the stability of the clustering results under 5% random noise injection, reflecting the algorithm's resistance to outliers and abnormal data.

Sensitivity: Evaluates the independence of the results from the initial centroid selection and parameter variation, confirming stable performance for $\theta \in [0.3, 0.7]$.

Complexity: Represents the simplicity and computational feasibility of the proposed approach in comparison with other incremental clustering algorithms.

These metrics collectively provide a comprehensive assessment of accuracy, efficiency, stability, and adaptability, forming the basis for the comparative analysis presented in the following subsections.

6.2. A comparative study of the performance of incremental clustering algorithms in big data environments

This experiment presents a comparative evaluation of several incremental clustering algorithms, including the proposed approach, under identical computational and data conditions.

conditions. The comparison focuses on multiple performance criteria—accuracy, execution time, novelty detection rate, robustness to noise, and implementation complexity—to ensure a fair and comprehensive assessment.

All algorithms were executed using the experimental setup described earlier with standardized parameter configurations to maintain fairness. The Titanic dataset was selected as a representative benchmark because of its mixed attribute types and moderate size, which allowed us to observe the incremental behavior clearly.

The results summarized in Table 1 demonstrate that the proposed algorithm achieves competitive accuracy while significantly reducing execution time compared with Incremental K-means ++, Incremental DBSCAN, and Incremental Mean Shift. Furthermore, the proposed method exhibits stable performance across incremental updates and maintains robustness under noisy conditions, confirming its suitability for large-scale and evolving data environments.

6.3. The proposed algorithm is superior and more efficient than the rest.

The experimental results summarized in Table 2 present a detailed comparison between the proposed algorithm and other incremental clustering methods using the Titanic dataset. The proposed approach achieved an accuracy of 81.5%, which is equivalent to that of incremental DBSCAN, but with a significantly lower execution time only 22 s compared with 100 s for DBSCAN.

This substantial reduction in processing time demonstrates the computational efficiency of the proposed algorithm while maintaining the same level of clustering accuracy. Such performance makes it particularly suitable for real-time and large-scale applications, where rapid response and scalability are critical.

The graphical representation in Figure 5 illustrates the comparative results from Table 2, clearly showing that the proposed algorithm consistently outperforms the other incremental clustering techniques in terms of speed and overall efficiency, particularly in environments requiring fast data processing and continuous updates.

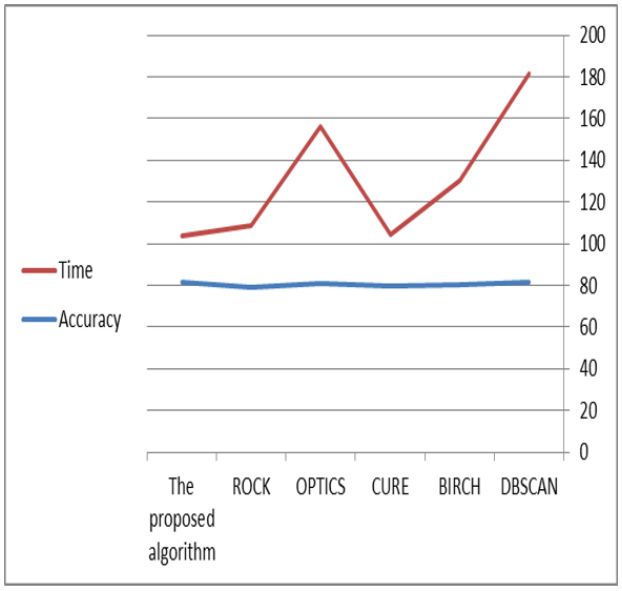

Figure 5. Comparison of our approach with different incremental clustering algorithms in terms of Time and Accuracy

6.4. Demonstrating the effectiveness of the proposed algorithm compared to other incremental clustering algorithms

This experiment evaluates the effectiveness of the proposed algorithm against several incremental clustering techniques when applied to mixed-type datasets. The comparative results are summarized in Table 3, which highlights both the accuracy and execution time across

Table 1. the performance of the proposed algorithm against incremental clustering algorithms

Algorithm	Speed	Robustness to noise	Sensitivity to initial choice of centroids	Complexity
The proposed Algorithm	Fast for large datasets	Not as robust as incremental clustering algorithms	Not Sensitive to initial choice of centroids	Simple to implement
Incremental k-means++	Faster for small datasets	More robust than algorithm you provided	Not as sensitive to initial choice of centroids	More complex to implement
Incremental DBSCAN	Faster for small datasets	More robust than algorithm you provided	Not as sensitive to initial choice of centroids	More complex to implement
Incremental Mean Shift	Fastest for small datasets	Most robust to noise and outliers	Not as sensitive to initial choice of centroids	Most complex to implement

Table 2. The proposed algorithm is compared with existing algorithms incremental clustering algorithms on the Titanic data set

Algorithm	Accuracy	Time (seconds)
DBSCAN	**81.5%**	**100**
BIRCH	**80.5%**	**50**
CURE	**79.5%**	**25**
OPTICS	**81.0%**	**75**
ROCK	**79.0%**	**30**
The proposed algorithm	**81.5%**	**22**

different data scales.

As shown in Table 3, the proposed algorithm consistently achieved the highest accuracy while maintaining the lowest execution time among all evaluated methods. In contrast, the incremental mean-shift algorithm exhibited the lowest accuracy, although it remained one of the fastest among traditional incremental approaches.

It was observed that the accuracy of all incremental clustering algorithms tends to decrease slightly as the dataset size increases. This behavior occurs because incremental methods often require maintaining partial or complete data representations in memory, which becomes challenging for large-scale datasets. Nevertheless, the proposed algorithm demonstrates superior scalability and efficiency, maintaining stable performance even as the data volume grows.

These findings confirm that the proposed approach provides an optimal balance between accuracy, speed, and resource utilization, making it particularly effective for big data environments, in which both precision and computational efficiency are essential.

6.5. The Evolution of Knowledge Discovery: A Quantitative Analysis of the Sustainability of New Patterns in Databases

In this research, we experimented with four public datasets. Table 4 summarizes the descriptions of these datasets. The objective of the experiment was to demonstrate that the number of discovered novel rules decreases over time. The novel patterns discovered at time t_1 are no longer novel at time t_2, and the number of patterns varied as expected. The nature of the datasets is summarized in Table 4. Table 5 shows the different numbers of discovered patterns and novel patterns from different datasets and

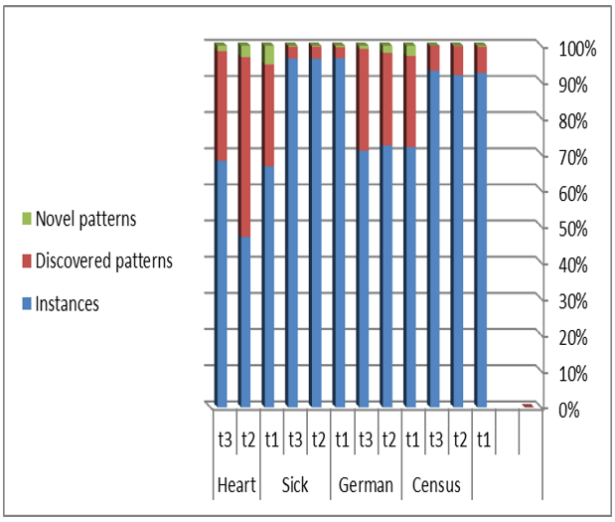

Figure 6. Number of discovered patterns and novel patterns and novel patterns from different datasets with novelty threshold (θ)

6.6. Discussion of Limitations and Computational Complexity

6.6.1. Computational Complexity

The computational complexity of the proposed incremental clustering algorithm is primarily determined by the distance calculations between new data points and existing cluster centroids. Unlike traditional K-means, which has a complexity of O(n.k.i), our approach leverages a dynamic pruning method based on the novelty criterion. This reduces the number of unnecessary distance computations, thereby lowering the practical time complexity. Because the algorithm is implemented on the Hadoop parallel framework, the workload is distributed across multiple nodes, making the execution time significantly faster (22 s) than that of sequential algorithms such as

Table 3. Demonstrating the effectiveness of the proposed algorithm

Online Retail	Customer Segmentation	Adult	Data set	
541909	200,000	48,842	Size of dataset	
65%	70%	80%	Accuracy	Incremental K-means++
2.8 hours	2 hours	56 Min	Time to cluster (hours)	
55%	60%	70%	Accuracy	Incremental DBSCAN
2.2 hours	1.7 hours	49 Min	Time to cluster (hours)	
50%	55%	65%	Accuracy	Incremental Mean Shift
1.7 hours	1.2 hours	38 Min	Time to cluster (hours)	
75%	82%	90%	Accuracy	The proposed approach
1.1 hours	52 Min	14 Min	Time to cluster (hours)	

Table 4. Summarized description of the datasets

Dataset name	Instances	Attributes	Data types
Census	32561	14	Mixed
German	1000	20	Mixed
Sick	2800	29	Mixed
Heart	270	13	Mixed

Table 5. Discovered patterns at time t_1, t_2 and t_3 or different datasets with $\Phi = 0.5$

Dataset	Time	Instances	Discovered patterns Rules	Novel patterns
Census	t_1	12000	943	29
	t_2	12000	1061	6
	t_3	8561	636	3
German	t_1	333	117	13
	t_2	333	118	9
	t_3	334	133	4
Sick	t_1	933	29	4
	t_2	933	33	2
	t_3	934	32	2
Heart	t_1	90	38	7
	t_2	90	95	6
	t_3	90	40	2

DBSCAN (100 s).

6.6.2. Limitations

Despite its efficiency, the proposed approach has some limitations:

6.6.2.1 **Threshold Sensitivity:** The performance and number of discovered clusters are highly dependent on the user-defined novelty threshold (θ). An inappropriate threshold may lead to over-segmentation or merging of distinct patterns.

6.6.2.2 **Memory Constraints:** As an incremental algorithm, it must maintain cluster summaries (centroids and variances) in memory to process evolving data streams.

6.6.2.3 **Semi-metric Nature:** The novelty measure used as a distance metric is considered a "semi-metric" because it does not always satisfy the triangle inequality. Although this allows for capturing user subjectivity, it may limit the use of certain indexing structures that require a formal metric space.

7. CONCLUSION

We proposed an incremental Clustering mining algorithm that integrates novelty criterion during the process of building the clusters in order to discover only interesting patterns from big data. The novelty criterion is adapted to be used as a distance metric into the clustering algorithm. It computes the deviation between two instances (mixed data types). In addition, the proposed approach dynamically determine the number of the clusters and no need to specify the number of clusters advance, It. The proposed is implemented and tested using public datasets and found the results very promising. And Through experimentation it became clear that the proposed algorithm achieved the same high accuracy of 81.5% as the DBSCAN algorithm. and However it surpassed it in speed, taking only 22 seconds compared to 100 seconds. This means it is more efficient the same level of accuracy but in significantly less time, making it a better option for practical applications.

REFERENCES

[1] A. E. Ezugwu et al., "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *Eng. Appl. Artif. Intell.*, vol. 110, p. 104 743, 2022. DOI: 10.1016/j.engappai.2022.104743.

[2] A. Abernathy and M. E. Celebi, "The incremental online k-means clustering algorithm and its application to color quantization," *Expert Syst. with Appl.*, vol. 207, p. 117 927, 2022. DOI: 10.1016/j.eswa.2022.117927.

[3] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Inf. Sci.*, vol. 622, pp. 178–210, 2023. DOI: 10.1016/j.ins.2022.11.139.

[4] J. Redha Mutar, "A review of clustering algorithms," *Int. J. Comput. Sci. Mob. Appl.*, vol. 10, no. 10, pp. 44–50, 2022. DOI: 10.5281/zenodo.7243829.

[5] V. Bhatnagar, A. S. Al-Hegami, and N. Kumar, "A hybrid approach for quantification of novelty in rule discovery," in *Proceedings of International Conference on Artificial Learning and Data Mining (ALDM'05)*, 2005.

[6] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011.

[7] A. M. Sowjanya and M. Shashi, "A cluster feature-based incremental clustering approach to mixed data," *J. Comput. Sci.*, vol. 7, no. 12, p. 1875, 2011.

[8] N. Kerdprasop and K. Kerdprasop, "Data partitioning for incremental data mining," in *Proceedings of 1st International Forum on Information and Computer Science*, 2003, pp. 114–118.

[9] R. K. Prasad, R. Sarmah, and S. Chakraborty, "Incremental k-means method," in *Pattern Recognition and Machine Intelligence*, ser. Lecture Notes in Computer Science, vol. 11941, Springer, 2019.

[10] A. P. Kushwah, S. Jaloree, and R. S. Thakur, "A comparative review of incremental clustering methods for large dataset," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 10, no. 2, 2021.

[11] H. Alsaeedi and A. S. Alhegami, "An incremental interesting maximal frequent itemset mining based on fp-growth algorithm," *J. Complex.*, vol. 2022, p. 1 942 517, 2022.

[12] A. S. Alhegami and H. Alsaeedi, "A framework for incremental parallel mining of interesting association patterns for big data," *Int. J. Comput.*, vol. 19, no. 1, pp. 106–117, 2020.

[13] W. Zhao, L. Li, S. Alam, and Y. Wang, "An incremental clustering method for anomaly detection in flight data," *Transp. Res. Part C: Emerg. Technol.*, vol. 132, p. 103 406, 2021.

[14] E. Liberty, R. Sriharsha, and M. Sviridenko, "An algorithm for online k-means clustering," 2014, pp. 81–89. DOI: 10.1137/1.9781611974317.7.

[15] Y. Li, W. G. Macready, and G. Chen, "Streaming k-means approximation," in *Proceedings of the 2015 SIAM International Conference on Data Mining*, 2015, pp. 189–197.

[16] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM International Conference on Data Mining*, 2006.

[17] C. C. Aggarwal, P. S. Yu, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings of VLDB 2003*, 2003.

[18] H. Al-Khamees, A. N. Al-A'araji, and E. S. Al-Shamery, "Survey: Clustering techniques of data stream," in *2021 1st Babylon International Conference on Information Technology and Science*, 2021, pp. 113–119. DOI: 10.1109/BICITS51482.2021.9509923.

[19] M. Amini, H. Saboohi, T. Herawan, and Y. Wah, "Mudistream: A multi density clustering algorithm for evolving data stream," *J. Netw. Comput. Appl.*, vol. 59, pp. 370–385, 2016.

[20] E. Azhir, N. N. Jafari, M. Hosseinzadeh, A. Sharifi, and A. Darwesh, "An efficient automated incremental density-based algorithm for clustering and classification," *Future Gener. Comput. Syst.*, vol. 114, pp. 665–678, 2021.

[21] J. Shao, Y. Tan, L. Gao, Q. Yang, C. Plant, and I. Assent, "Synchronization-based clustering on evolving data stream," *Inf. Sci.*, vol. 501, pp. 573–587, 2019. DOI: 10.1016/J.INS.2018.09.035.

[22] S. Chakraborty and N. Nagwani, "Analysis and study of incremental k-means clustering algorithm," in 2011, pp. 338–341. DOI: 10.1007/978-3-642-22577-2_46.

[23] S. Madan and K. Dana, "M-birch: An online clustering approach for computer vision applications," vol. 9408, 2015. DOI: 10.1117/12.2078264.

[24] J. S. Fu, Y. Liu, and H. C. Chao, "Ica: An incremental clustering algorithm based on optics," *Wirel. Pers. Commun.*, vol. 84, pp. 2151–2170, 2015. DOI: 10.1007/s11277-015-2517-9.

[25] A. F. Omer, H. A. Mohammed, M. A. Awadallah, Z. Khan, S. U. Abrar, and M. D. Shah, "Big data mining using k-means and dbscan clustering techniques," in *Big Data Analytics and Computational Intelligence for Cybersecurity*, ser. Studies in Big Data, vol. 111, Springer, 2022.

[26] P. Mulay and P. Kulkarni, "Knowledge augmentation via incremental clustering: New technology for effective knowledge management," *Int. J. Bus. Inf. Syst.*, vol. 12, p. 68, 2013.

[27] T. Tran, R. Nayak, and P. Bruza, "Document clustering using incremental and pairwise approaches," in *Focused Access to XML Documents*, ser. Lecture Notes in Computer Science, vol. 4862, Springer, 2008.

[28] M. K., A. M., and M. A., "Map reduce clustering in incremental big data processing," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 2, 2019.

[29] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Cloud Computing*, ser. Lecture Notes in Computer Science, vol. 5931, Springer, 2009.

[30] L. H. Son and N. D. Tien, "Tune up fuzzy c means for big data: Some novel hybrid clustering algorithms based on initial selection and incremental clustering," *Int. J. Fuzzy Syst.*, vol. 19, pp. 1585–1602, 2017.

[31] H. Najdataei, V. Gulisano, P. Tsigas, and M. Papatriantafilou, "Pi-lisco: Parallel and incremental stream-based point-cloud clustering," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022.

[32] K. Reddy and C. Bindu, "Streamsw: A density-based approach for clustering data streams over sliding windows," *Measurement*, 2019.

[33] S. Sahoo, "A parallel forecasting approach using incremental k-means clustering technique," in *Computational Intelligence in Data Mining*, ser. Advances in Intelligent Systems and Computing, vol. 556, Springer, 2017.

[34] S. T. Mai, J. Jacobsen, S. Amer-Yahia, I. Spence, and T. Nhat-Phuong, "Incremental density-based clustering on multicore processors," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1338–1356, 2022.

[35] W. Lu, "Improved k-means clustering algorithm for big data mining under hadoop parallel framework," *J. Grid Comput.*, vol. 18, pp. 239–250, 2020.

[36] A. Bakr, N. Ghanem, and M. Ismail, "Efficient incremental density-based algorithm for clustering large datasets," *Alex. Eng. J.*, vol. 54, pp. 1147–1154, 2015.

[37] S. Chakraborty and N. K. Nagwani, "Analysis and study of incremental k-means clustering algorithm," in *High Performance Architecture and Grid Computing*, ser. Communications in Computer and Information Science, vol. 169, Springer, 2011.

[38] T. Özyer and R. Alhajj, "Parallel clustering of high dimensional data by integrating multi-objective genetic algorithm with divide and conquer," *Appl. Intell.*, vol. 31, pp. 318–331, 2009.

[39] W. Zhu, W. Yu, B. Kan, and G. Liu, "Smart meter data analytics based on modified streaming k-means," in *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, 2017, pp. 328–333. DOI: 10.1109/BIGCOM.2017.49.