AFGBStream: Adaptive Fuzzy Granular-Ball Framework for Stream Clustering With Sliding Windows

Qijia Wang, Mingjing Du, *Member, IEEE*, and Weiping Ding, *Senior Member, IEEE*

***Abstract*—Existing sliding window-based data stream clustering methods often struggle with fixed-granularity representations and delayed responses to abrupt pattern changes, leading to reduced adaptability and clustering accuracy. To address these limitations, we propose an Adaptive Fuzzy Granular-Ball framework for Stream clustering with sliding windows (AFGBStream), a novel sliding window-based data stream clustering framework that integrates fuzzy set theory with granular-ball computing to support adaptive and incremental updates. Specifically, a window-guided data granulation strategy is developed using a fuzzy C-means (FCM)-based granular-ball generation approach, which enhances representation robustness and reduces sensitivity to microcluster parameters. In addition, a window-aware temporal decay mechanism is employed to improve the model's responsiveness to evolving data by dynamically filtering outdated information. Our comprehensive experimental evaluation on 17 benchmark datasets demonstrates that AFGBStream achieves superior performance over five competing methods.**

***Index Terms*—Clustering, data stream, granular ball (GB), granular computing, sliding window.**

I. Introduction

DATA streams have emerged as a dominant data format across multiple domains, including finance, industrial IoT, and social media. Unlike static datasets, which remain relatively constant after collection, data streams are characterized by high-frequency continuous generation and dynamic evolution [1], posing significant challenges for conventional analysis techniques [2].

The challenges of analyzing dynamic data streams have led to significant research into data stream clustering [3], a key technique for discovering inherent structures and patterns within data. Clustering enables the identification of groups of similar data points, offering insights into trends, relationships, and anomalies [4]. Nevertheless, most conventional clustering models rely on the assumption of static data distributions, making them ineffective in handling the evolving and dynamic nature of data streams [5].

In this context, window models have emerged as a common strategy for enabling clustering models to operate effectively in streaming scenarios. Window models typically employ three fundamental paradigms: sliding window, landmark window, and damped window methods [6]. Among them, the sliding window model has gained particular prominence due to its flexibility and simplicity [7]. Its ability to maintain a fixed-sized, continuously updated view of the stream makes it especially suitable for applications where recent data are most indicative of the current state.

This sliding window mechanism employs a dynamic maintenance strategy that continuously integrates new observations while expiring outdated data, ensuring the clustering model adapts to evolving data stream characteristics [8], [9]. Current sliding window-based data stream clustering methods still face several challenges. On one hand, many of these methods use single-granularity processing mechanisms (e.g., microcluster models using fixed-radius thresholds), which cannot adequately capture the multiple density structures inherent in data streams [10], [11]. On the other hand, sliding window models typically update the data window at a fixed rate, resulting in delayed responses to sudden pattern changes, which can negatively impact the timeliness and stability of clustering outcomes [12].

To address the aforementioned issues, this article proposes an **A**daptive **F**uzzy **G**ranular-**B**all framework for **Stream** clustering with sliding windows (AFGBStream) method. The proposed method leverages fuzzy set theory and granular-ball (GB) computing within a three-stage processing framework to achieve dynamic optimization for data stream clustering. First, during the data ingestion phase, AFGBStream employs a window-guided data granulation strategy, which uses a fuzzy partitioning mechanism to convert incoming data streams into adaptive GB

Received 15 April 2026; accepted 17 May 2026. Date of publication 21 May 2026; date of current version 6 August 2026. This work was supported in part by the Qinglan Project of Jiangsu Province of China, the National Natural Science Foundation of China under Grant 62006104, Grant 62576178, and Grant U2433216, in part by the National Key Research and Development Program of China under Grant 2024YFE0202700, in part by the Modern Agricultural Machinery Equipment and Technology Extension Project of Jiangsu Province under Grant NJ2024-06, and in part by the Jiangsu Normal University Faculty Research Excellence Development Program under Grant JSNUGZL2026032. Recommended by Associate Editor S. Auephanwiriyakul. *(Corresponding author: Mingjing Du.)*

Qijia Wang and Mingjing Du are with the Jiangsu Key Laboratory of Educational Intelligent Technology, School of Artificial Intelligence and Computer Science, Jiangsu Normal University, Xuzhou 221116, China (e-mail: wwqqjj98@163.com; dumj@jsnu.edu.cn).

Weiping Ding is with the School of Artificial Intelligence and Computer Science, Nantong University, Nantong 226019, China (e-mail: dwp9988@163.com).

Data is available on-line at https://github.com/Du-Team/M3W/AFGBStream.

This article has supplementary downloadable material available at https://doi.org/10.1109/TFUZZ.2026.3695688, provided by the authors.

Digital Object Identifier 10.1109/TFUZZ.2026.3695688

online. Next, in the model update phase, the microball (MB) evolution mechanism dynamically maintains the basic clustering units through generation, expansion, splitting, removal, shrinkage, and elimination operations. Finally, in the cluster formation phase, clustering operations are performed on the active MBs in the current window to achieve dynamic grouping of the data stream. The main contributions of the algorithm include the following.

1) We propose a novel sliding window-based data stream clustering (named AFGBStream), which introduces fuzzy set theory and GB computing into the sliding window-based framework, ensuring the model's adaptability while reducing its sensitivity to parameters.
2) We develop the window-guided data granulation strategy, which employs a fuzzy C-means (FCM)-based GB generation algorithm to granulate newly arrived data within each window, improving complex pattern recognition.
3) We propose a window-aware temporal decay mechanism, designed for MBs, that applies exponential decay to outdated data, preventing interference from obsolete information in the current model, thereby enhancing its ability to capture sudden changes in patterns.

The rest of this article is organized as follows. Section II provides a review of related research in the field; Section III presents the proposed AFGBStream framework; Section IV evaluates the method using benchmark datasets, comparing it against key baseline approaches. Finally, Section V concludes this article and outlines potential directions for future research.

II. Related Work

This section reviews two research areas that are closely related to our study: data stream clustering and GB clustering.

A. Data Stream Clustering

In data stream clustering, window models are commonly employed to handle dynamically arriving data and manage obsolete information through incremental updates. DenStream [6] employs a damped window model to dynamically maintain the microcluster density, and incorporates the concept of density connectivity from DBSCAN to guide cluster structure updates. However, the use of a fixed microcluster radius limits the model's ability to adapt to changes in data distribution. Unlike microcluster-based approaches, D-Stream [13], [14] employs a grid-based partitioning strategy that determines cluster patterns by density estimation within grid cells, thereby improving efficiency, especially in high-density regions. However, its fixed partitioning is sensitive to data scale and encounters the curse of dimensionality in high-dimensional spaces [15]. To tackle the challenges of high-dimensional data streams, EmCStream [16] integrates UMAP-based streaming dimensionality reduction and k-means clustering, projecting data into a 2-D latent space before clustering. Unlike the damped window model, B. Kim et al.[12] introduced an incremental density-based clustering algorithm for streaming data under the sliding window framework. It represents clusters through the DenTree structure (a specialized spanning tree representation) that reduces computational complexity for dynamic updates.

Beyond density and grid-based models, fuzzy clustering has evolved to handle complex uncertainties. For static data, AGK-FCM [17] improves robustness against high-dimensional noise by employing adaptive sample weighting. To extend fuzzy theory to data streams, AFSDP [3] proposes a diffusion-enhanced contextual affinity method that utilizes axiomatic fuzzy sets and spectral clustering to capture evolving patterns. However, relying on spectral clustering and complex affinity matrix updates can be computationally intensive, highlighting the need for more efficient granular frameworks in high-speed stream processing.

B. GB Clustering

To facilitate efficient and robust data representation, the GBkNN algorithm [18] introduces a GB coverage model for the sample space, significantly enhancing efficiency, robustness, and interpretability in data processing. Building on this representation framework, a variety of GB clustering methods have been subsequently proposed. For instance, Ball k-means [19] accelerates the clustering process by leveraging distances between data points and GB centers. It represents each cluster using GB, whose precise geometric structures and coarse-grained characterizations help identify coarse-grained neighboring relationships among clusters, thereby reducing unnecessary distance calculations and enhancing the efficiency of traditional k-means.

To effectively handle complex, nonspherical data, GBC [20] designs an adaptive GB generation mechanism, which dynamically adjusts the shape and size of each ball through heuristic quality evaluation and splitting strategies. Based on the constructed GB, it further employs a hierarchical clustering framework to efficiently group data with a complex distribution. To rigorously handle the uncertainty in boundary regions, recent research has integrated the three-way decision (3WD) theory into GB computing. Initially, the 3WD-FGBRS model [21] is constructed based on uncertainty invariance to avoid subjective parameter definitions. Building on this, the 3WC-GBNRS++ classifier [22] combines GBs with neighborhood rough sets, utilizing minimum fuzziness loss to objectively determine optimal thresholds. More recently, the CS3W-GBG method [23] introduces a cost-sensitive GB generation approach that incorporates sequential decisions to dynamically optimize granularity structure under uncertain conditions.

Similarly, Cheng et al. [24] proposed a GB-based density peak clustering algorithm, which computes densities at the GB level rather than for individual points and performs clustering using distances between ball centers. To address the high computational cost of spectral clustering, Xie et al. [25] developed a novel GB-based approach. By modeling data as GBs and deriving an adjacency matrix from their relationships, the algorithm achieves efficient spectral clustering without the need for expensive pairwise distance calculations. To improve clustering performance, GBCT [26] introduces an efficient and flexible approach for representing and covering the sample space by developing a center consistency measure based on GBs. Requiring only the cluster number K as input, it adapts to various complex datasets, demonstrating notable flexibility. Some work integrates GB computing into data stream clustering frameworks, such as GB-FuzzyStream [27].

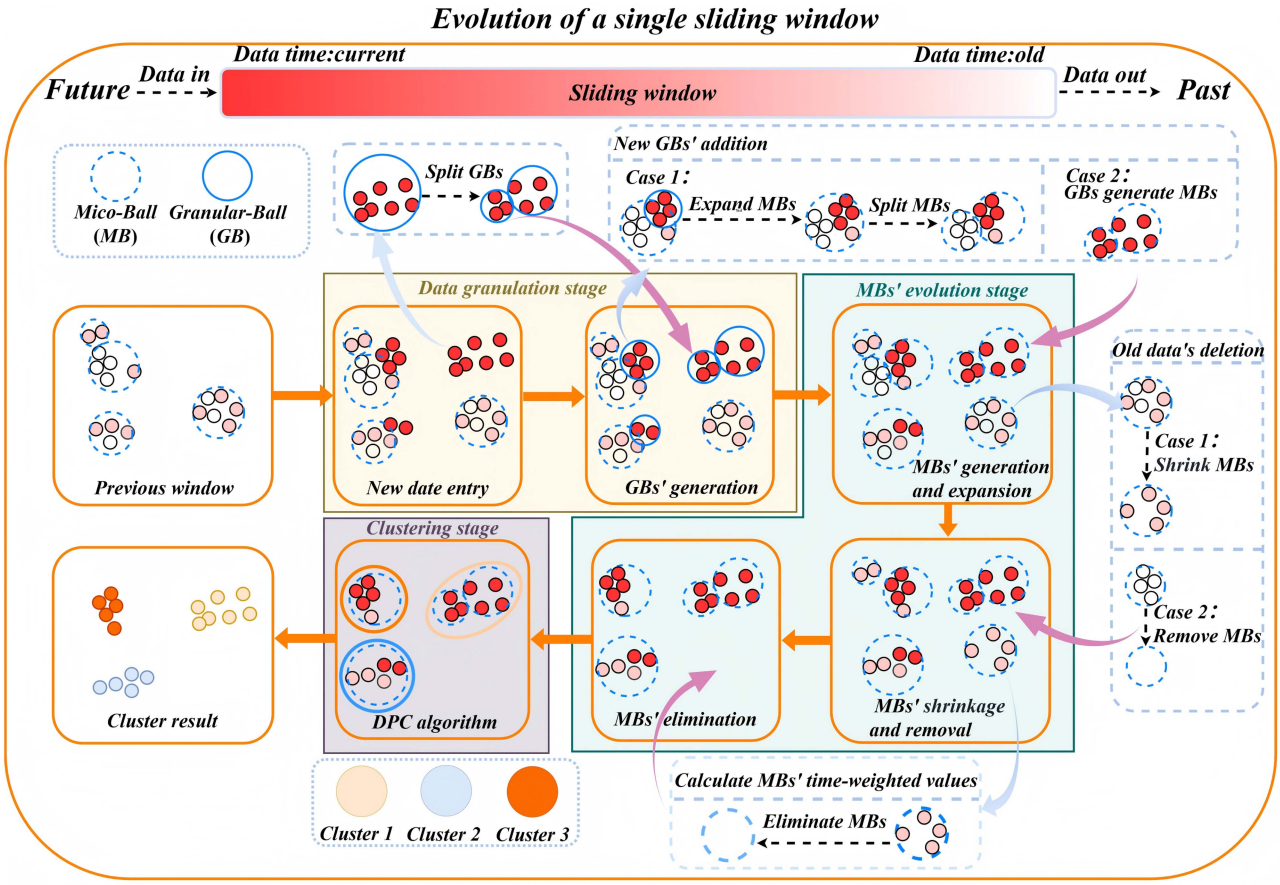

Fig. 1. Overall structure of AFGBStream framework.

As discussed, although much effort has been dedicated for data stream and GB clustering algorithms [28], the noted algorithms suffer from the following limitations and challenges.

1) Most existing data stream clustering methods rely on fixed-granularity representations, such as constant micro-cluster radii or rigid grid partitions, which lack the flexibility to adapt to evolving data distributions with multiple density structures.
2) While GB computing provides a flexible granularity representation, its direct application to streaming scenarios often results in excessive data accumulation within individual balls, necessitating frequent splitting and merging operations that increase computational overhead.
3) Conventional sliding window models typically employ uniform update rates that cannot promptly respond to sudden pattern changes, leading to delayed adaptation and unstable clustering performance.

In contrast, our method adopts a sliding window model, effectively mitigating the issue of data accumulation within GB.

III. Proposed Method

The framework of AFGBStream is illustrated in Fig. 1. AFGBStream comprises three main stages: data granularity, MB evolution, and clustering. Specifically, as new data points arrive within the window, they are processed in batches using a window-guided data granulation strategy. Each MB, serving as the basic clustering unit, is constructed with a MB feature vector and dynamically maintained through a weighted decay mechanism. Finally, the density peaks clustering (DPC) algorithm is applied to the set of MB within the window to uncover the underlying cluster structure.

A. Data Preprocessing

In data stream processing, the complete storage and analysis of data streams are inherently impossible, as they are unbounded and grow continuously, while processing systems are fundamentally limited by finite memory. Therefore, we adopt the sliding window model, which is widely used to capture the recent state of data streams. This model operates by maintaining a fixed-size window over the stream to incorporate new data and discard obsolete observations. Based on this, we provide the relevant definitions of the sliding window.

Definition 1 (Window and slide): Let the data stream be $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots\}$. Given a sliding window of size W (the number of data points analyzed per window) and stride length S (the interval between consecutive window updates), the subset of data in the ωth window, denoted as $\boldsymbol{X}^{(\omega)}$, is defined as

$$\boldsymbol{X}^{(\omega)} = \left\{\boldsymbol{x}_{(\omega-1)S+1}, \boldsymbol{x}_{(\omega-1)S+2}, \ldots, \boldsymbol{x}_{(\omega-1)S+W}\right\}. \quad (1)$$

The set of new data points that enter the ωth window, denoted as $\hat{\boldsymbol{X}}^{(\omega)}$, is defined as

$$\hat{\boldsymbol{X}}^{(\omega)} = \{\boldsymbol{x}_{(\omega-1)S+W-S+1}, \ldots, \boldsymbol{x}_{(\omega-1)S+W}\}. \tag{2}$$

The S data points removed in the ωth window are

$$\check{\boldsymbol{X}}^{(\omega)} = \left\{\boldsymbol{x}_{(\omega-2)S+1}, \boldsymbol{x}_{(\omega-2)S+2}, \ldots, \boldsymbol{x}_{(\omega-2)S+S}\right\}. \tag{3}$$

It is worth noting that all data points in the set $\hat{\boldsymbol{X}}^{(\omega)}$ are assigned a unified timestamp $t^{(\omega)}$, corresponding to the starting time of the ωth window.

To manage dynamic data streams efficiently, we design a sliding window management module based on the above definitions. This module is responsible for window initialization and ongoing updates. Specifically, a buffer is initialized with a predefined window size W, and the initial timestamp $t^{(1)}$ is recorded. The first window is formed once the buffer is saturated. In each subsequent update, the sliding window advances by a step size of S, where the newest S data points are incorporated, and the oldest S data points are removed, resulting in the formation of an updated window. The timestamps of all new data points in the updated window are aligned to the window's sliding timestamp $t^{(\omega)}$.

B. Data Granulation

1) GB Feature Vector: To efficiently store and summarize information in continuous data streams, microclusters are commonly employed to provide compact representations of local data distributions. However, traditional microcluster approaches based on a fixed radius struggle to adapt to the evolving density and shape of data distributions in streaming environments. To address these challenges, we design a granularization strategy (i.e., window-guided data granulation strategy) under the sliding window model by introducing the concept of the GB. During each window update, the window-guided data granulation strategy partitions the newly arrived data points into GB. In other words, these GB serve as temporary structures for maintaining new data points entering each window. To facilitate the introduction of these concepts, we first provide the relevant notations and definitions.

To record essential information of each GB, we define the concept of the GB feature vector.

Definition 2 (GB feature vector): For each granular ball $\boldsymbol{g}_k$, the corresponding GB feature vector is defined as a tuple

$$\mathbf{GBF}_{\boldsymbol{g}_k} = (\boldsymbol{\Theta_k}, \boldsymbol{\theta}_k, r_{\boldsymbol{g}_k}, \zeta_{\boldsymbol{g}_k}, \varepsilon_{\boldsymbol{g}_k}, \gamma_{\boldsymbol{g}_k}) \tag{4}$$

where $\boldsymbol{\Theta_k}$ denotes the set of data points within the GB $\boldsymbol{g}_k$, $\boldsymbol{\theta}_k$ is the center of the granular ball (see Definition 3), $r_{\boldsymbol{g}_k}$ is its radius (see Definition 4), $\zeta_{\boldsymbol{g}_k}$ quantifies the distribution measure of the GB (see Definition 5), $\varepsilon_{\boldsymbol{g}_k}$ is the GB splitting threshold [see (8)] and $\gamma_{\boldsymbol{g}_k}$ denotes its influence radius (see Definition 6).

Definition 3 (Center of GB): For a single GB $\boldsymbol{g}_k$, its center is determined by the mean of the data points within the GB

$$\boldsymbol{\theta}_k = \frac{1}{N_{\boldsymbol{\Theta_k}}} \sum_{\boldsymbol{x}_i \in \boldsymbol{\Theta_k}} \boldsymbol{x}_i \tag{5}$$

and $/varTheta_k$ denotes the set of data points within the GB.

Definition 4 (Radius of GB): For a single GB $\boldsymbol{g}_k$, its radius is defined as the maximum Euclidean distance from the data points within the GB to its center

$$r_{\boldsymbol{g}_k} = \max_{\boldsymbol{x}_i \in \boldsymbol{\Theta}_k} \|\boldsymbol{\theta}_k - \boldsymbol{x}_i\|_2. \tag{6}$$

To evaluate the stability of a GB (i.e., whether it needs further partitioning), the distribution measure can be employed as an indicator of internal compactness [27].

Definition 5 (Distribution measure of GB): For a GB $\boldsymbol{g}_k$, which has center $\boldsymbol{\theta}_k$ and data point set $\boldsymbol{\Theta}_k$ containing $N_{\boldsymbol{\Theta}_k}$ data points, its distribution measure is defined as the reciprocal of the average proximity between all data points in the GB and its center

$$\zeta_{\boldsymbol{g}_k} = \frac{N_{\boldsymbol{\Theta_k}}}{\sum_{\boldsymbol{x}_i \in \boldsymbol{\Theta_k}} \|\boldsymbol{\theta}_k - \boldsymbol{x}_i\|_2^2}. \tag{7}$$

Notably, an increase in the GB distribution measure corresponds to greater compactness and higher stability of the GB, and conversely.

Based on the concept of the distribution measure, the splitting threshold is defined as the weighted sum of the distribution measures of the two sub-GB, as follows:

$$\varepsilon_{\boldsymbol{g}_k} = \frac{N_{\boldsymbol{\Theta_\alpha}}}{N_{\boldsymbol{\Theta}_k}} \zeta_{\boldsymbol{g}_\alpha} + \frac{N_{\boldsymbol{\Theta_\beta}}}{N_{\boldsymbol{\Theta}_k}} \zeta_{\boldsymbol{g}_\beta} \tag{8}$$

where $N_{\boldsymbol{\Theta}_k}$ denotes the number of data points contained in the parent GB, while $N_{\boldsymbol{\Theta}_\alpha}$ and $N_{\boldsymbol{\Theta}_\beta}$ represent the number of data points in the two resulting sub-GB, respectively. The terms $\zeta_{\boldsymbol{g}_\alpha}$ and $\zeta_{\boldsymbol{g}_\beta}$ denote the distribution measures of the corresponding sub-GB.

After GB partitioning (see Algorithm 1) is completed, it is necessary to determine whether these new GB can be merged with existing MBs (see Definition 7). The merge criterion relies on the influence radii of both MBs and GB. The definition of the influence radius, in particular for GB, is provided below.

Definition 6 (Influence radius of the GB): For a given GB $\boldsymbol{g}_k$, its influence radius is defined as the Euclidean distance from its center to the center of the nearest neighboring MB

$$\gamma_{\boldsymbol{\theta}_k} = \min_{\boldsymbol{b}_l \in \boldsymbol{B}} \|\boldsymbol{\theta}_k - \boldsymbol{\phi}_l\|_2 \tag{9}$$

where $\boldsymbol{\theta}_k$ denotes the center of a GB, $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_L\}$ is the set of all MBs, and $\boldsymbol{\phi}_l$ denotes the center of MB $\boldsymbol{b}_l \in \boldsymbol{B}$ (see Definition 7).

2) Window-Guided Data Granulation Strategy: Inspired by [25] and [27], we incorporate the GB generation mechanism into a sliding window model and propose a window-guided data granulation strategy that transforms newly arriving data points within the window into adaptive GB. The window-guided data granulation strategy is detailed in Algorithm 1.

First, the newly arriving data points within the window are treated as a candidate set for processing. This set is then granulated into a single GB (see Lines 1 and 2). Next, the distribution measure of the GB is computed according to Definition 5 to assess its compactness and stability (Line 3).

Algorithm 1: Window-Guided Data Granulation Strategy.

Input: new data points in ω-th window $\hat{\boldsymbol{X}}^{(\omega)}$
Output: granular balls in ω-th window $\boldsymbol{G}^{(\omega)}$
1 Initialize: queue $\boldsymbol{Q} \leftarrow \emptyset$; granular-ball set $\boldsymbol{G}^{(\omega)} \leftarrow \emptyset$; index $k \leftarrow 1$;
2 Construct root granular ball $\boldsymbol{g}_0$ covering $\hat{\boldsymbol{X}}^{(\omega)}$;
3 Compute the distribution measure $\zeta_{\boldsymbol{g}_0}$ of a granular ball $\boldsymbol{g}_0$ accroding to Eq (7);
4 Enqueue $\boldsymbol{g}_0$ into $\boldsymbol{Q}$;
5 **while** $\boldsymbol{Q}$ *is not empty* **do**
6 Dequeue a granular ball $\boldsymbol{g}$ from $\boldsymbol{Q}$;
7 Apply FCM to split $\boldsymbol{g}$ two sub-granular balls: $\boldsymbol{g}_\alpha$ and $\boldsymbol{g}_\beta$;
8 Compute $\zeta_{\boldsymbol{g}_\alpha}$ and $\zeta_{\boldsymbol{g}_\beta}$ accroding to Eq (7);
9 Compute $\varepsilon_{\boldsymbol{g}}$ according to Eq (8);
10 **if** $\varepsilon_{\boldsymbol{g}} > \zeta_{\boldsymbol{g}}$ **then**
11 Enqueue $\boldsymbol{g}_\alpha$ and $\boldsymbol{g}_\beta$ into $\boldsymbol{Q}$;
12 **else**
13 $\boldsymbol{g}_k \leftarrow \boldsymbol{g}$;
14 Compute GBF attributes of $\boldsymbol{g}_k$;
15 $\boldsymbol{G}^{(\omega)} \leftarrow \boldsymbol{G}^{(\omega)} \cup \{\boldsymbol{g}_k\}$;
16 $k \leftarrow k + 1$;
17 **end**
18 **end**
19 **return** $\boldsymbol{G}^{(\omega)}$

To determine whether further refinement is necessary, a two-class FCM algorithm is applied, generating two sub-GB, where the resulting clusters represent the new sub-GB and their centroids serve as the centers of the sub-GB (see Lines 6 and 7).

Next, the distribution measures of the two sub-GB are calculated, followed by the computation of the splitting threshold for the parent GB (see Lines 8 and 9).

Subsequently, we adopt a commonly used splitting criterion: if the distribution measure of the parent GB is lower than the computed splitting threshold, the GB is divided into two sub-GB; otherwise, the parent GB is retained (see Lines 10–18).

This splitting procedure is recursively applied to each resulting GB until no GB satisfies the condition for further splitting (see Lines 5–19).

C. MB Evolution

1) MB Feature Vector: To address abrupt pattern changes in data streams within the sliding window model, a common strategy is the design of an adaptive window size. However, this strategy faces two critical limitations. First, the need for continuous monitoring and frequent computation of statistical measures introduces significant computational overhead, which can paradoxically degrade real-time processing capabilities. Second, the strategy typically requires the setting of mutation detection thresholds for the desired statistics. The choice of these thresholds is highly sensitive and often requires fine-tuning, as a single static threshold may not be suitable for different data distributions. Unlike the adaptive window size strategy, we introduce the concept of MBs into the sliding window model and propose an incremental MB management module, which involves three aspects: the MB feature vector, a window-aware temporal decay mechanism, and MB evolution operations (generation, expansion, splitting, removal, shrinkage, and elimination).

In our framework, MBs serve as fundamental clustering units. To capture their essential characteristics, we define the MB feature vector—a temporal extension of the GB feature vector that incorporates time-dependent attributes (e.g., timestamp, activity score) for precise modeling of evolving patterns in data streams.

Definition 7 (MB feature vector): For each MB $\boldsymbol{b}_l$, the corresponding MB feature vector is defined as a tuple

$$\mathbf{MBF}_{\boldsymbol{b}_l} = (\Phi_l, \boldsymbol{\phi}_l, r_{\boldsymbol{b}_l}, \zeta_{\boldsymbol{b}_l}, \gamma_{\boldsymbol{b}_l}, \hat{\varepsilon}_{\boldsymbol{b}_l}, t_{\boldsymbol{b}_l}^{(\omega)}, \eta_{\boldsymbol{b}_l}^{(\omega)}) \tag{10}$$

where Φ_l denotes the data points within the MB, $\boldsymbol{\phi}_l$ is the center of the MB, $r_{\boldsymbol{b}_l}$ is the radius, $\zeta_{\boldsymbol{b}_l}$ represents the distribution measure, $\gamma_{\boldsymbol{b}_l}$ is the influence radius (see Definition 8), $\hat{\varepsilon}_{\boldsymbol{b}_l}$ is the MB splitting threshold [see (8)], $t_{\boldsymbol{b}_l}^{(\omega)}$ denotes the timestamp of the window in which the MB was originally created, $\eta_{b_l}^{(\omega)}$ denotes the window-aware activity score (see Definition 9).

As previously discussed, the MB feature vector is an extension of the GB feature vector, where key components—including the set of data points within the MB, the MB's center, its radius, its distribution measure, and the splitting threshold—retain definitions analogous to those in the GB feature vector. This structural consistency ensures that the MB inherits the fundamental properties of GB.

However, the definition of the influence radius for a MB exhibits slight differences from that of a GB.

Definition 8 (Influence radius of the MB): For a given MB $\boldsymbol{b}_l$, its influence radius is defined as the Euclidean distance from its center to the center of the nearest neighboring MB

$$\gamma_{\boldsymbol{b}_l} = \min_{\boldsymbol{b}_p \in \boldsymbol{B} \setminus \{\boldsymbol{b}_l\}} \|\boldsymbol{\phi}_l - \boldsymbol{\phi}_p\|_2 \tag{11}$$

where $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_L\}$ denotes the set of all MBs, $\boldsymbol{\phi}_l$ denotes the center of MB $\boldsymbol{b}_l$, and $\boldsymbol{\phi}_p$ denotes the center of another MB $\boldsymbol{b}_p \in \boldsymbol{B} \setminus \{\boldsymbol{b}_l\}$.

Since explicit timestamps are often unavailable in real-world data streams, temporal information must be inferred indirectly from the sequence of sliding operations. Within the sliding window framework, and in accordance with Definition 1, the arrival time of newly added data points is uniformly assigned as the timestamp of the window in which they arrive, thereby establishing consistent temporal relationships. To incorporate temporal dynamics into the clustering process, each data point inherits its temporal attribute from the window it resides in, and this temporal attribute is further associated with the MB it is assigned to. Formally, let t^{ω} denote the timestamp of the ωth window. Then, for a given MB $\boldsymbol{b}_l$, we define $t_{\boldsymbol{b}_l}^{(\omega)}$ as the timestamp of the window in which $\boldsymbol{b}_l$ was originally created. This temporal initialization serves as the basis for tracking the evolution of MBs across successive windows, and is essential for the formulation of the window-aware activity score defined below.

To adaptively handle evolving data streams, we propose a window-aware temporal decay mechanism that progressively downweights historical information and emphasizes recent behavior within the sliding window framework. Unlike conventional decay strategies, this mechanism is explicitly aligned with

the temporal structure of sliding windows, allowing it to dynamically track the evolution of each MB across successive windows. By applying an exponential decay function over window-aligned timestamps, the mechanism provides a principled way to quantify the temporal relevance of each MB. This enables the model to detect and eventually eliminate outdated or inactive MBs, thereby improving its responsiveness to sudden pattern shifts while mitigating the influence of obsolete data.

To operationalize the proposed window-aware temporal decay mechanism, we define a window-aware activity score that incorporates both the data accumulation history and time decay effects to quantify the recent activity level of each MB. The formal definition is given as follows.

Definition 9 (Window-aware activity score): For a MB $\boldsymbol{b}_l$, its window-aware activity score at the ωth window is defined as

$$\eta_{\boldsymbol{b}_l}^{(\omega)} = \sum_{i=\underline{\omega}_{\boldsymbol{b}_l}}^{\omega-1} \frac{\hat{n}_{\boldsymbol{b}_l}^{(i+1)}}{n_{\boldsymbol{b}_l}^{(\omega)}} \cdot 2^{-\lambda\left(t^{(\omega)}-t^{(i)}\right)} \tag{12}$$

where $n_{\boldsymbol{b}_l}^{(\omega)}$ denotes the total number of data points contained in $\boldsymbol{b}_l$ at the ωth window, $\hat{n}_{\boldsymbol{b}_l}^{(i+1)}$ indicates new points added in $\boldsymbol{b}_l$ at the $(i+1)$th window, $\underline{\omega}_{\boldsymbol{b}_l}$ represents the window in which $\boldsymbol{b}_l$ was created, $t^{(\omega)}$ and $t^{(i)}$ are the timestamps of the corresponding windows, and λ is a positive decay coefficient that controls the rate at which past contributions diminish.

The activity score reflects the effective contribution of recent data to the current state of a MB. A high score indicates sustained recent activity, while a low score suggests temporal inactivity. This score is used to identify and eliminate inactive MBs.

2) Incremental Update Operations of MBs: Our model incorporates six distinct types of MB evolution: generation, expansion, splitting, removal, shrinkage, and elimination, each governed by different algorithmic rules. These evolutionary operations are designed to accommodate the dynamic nature of data streams under a sliding window mechanism. After newly arrived data points are granulated into GB, the model performs generation or expansion operations to transform them into MBs or merge them into existing ones; however, during expansion, some MBs may become structurally unstable, potentially triggering a splitting operation to preserve their compactness. Conversely, as outdated data points slide out of the current window, removal and shrinkage operations are employed to adjust the affected MBs accordingly. In addition, a decay mechanism is introduced to update the activation scores of MBs over time, leading to the elimination of inactive MBs.

To further clarify these evolutionary operations, we divide them into two categories based on the source of change: operations triggered by newly arrived data and those induced by data removal.

To handle newly arrived data points, the model triggers adaptive operations based on their structural relationship with existing MBs. Specifically, a GB derived from new data is either merged into an existing MB or initialized as a new one, determined by a a comparison of influence radii and proximity: if its influence radius is smaller than that of at least one existing MB, it merges into the nearest eligible one; otherwise, it forms a new MB.

When a MB expands (i.e., when a GB is merged into it), the resulting MB may become structurally unstable, potentially triggering a splitting operation to preserve its compactness. To determine whether a split is necessary, the model evaluates the distribution measure of the expanded GB following each expansion operation. Specifically, the updated MB is provisionally divided into two candidate subMBs using the FCM clustering, and a distribution measure is computed for both the original and the partitioned structures. Based on these, a splitting threshold is calculated. The splitting criterion is then applied: if the distribution measure of the parent MB is lower than the threshold, it is deemed structurally loose and is therefore split into the two sub-MBs; otherwise, the parent MB is retained. This evaluation process repeats recursively for each newly formed MB until no further division is warranted under the splitting criterion.

The operations triggered by newly arrived data (generation, expansion, and splitting) are termed the MB growth algorithm, as detailed in Algorithm 2.

Algorithm 2: MB Growth Algorithm.

Input: new granular balls $\boldsymbol{G}^{(\omega)}$, existing micro balls $\boldsymbol{B}$
Output: updated micro balls $\boldsymbol{B}$

Initialize: queue $\boldsymbol{Q} \leftarrow \emptyset$; index $l \leftarrow |\boldsymbol{B}| + 1$;
foreach $\boldsymbol{g}_i \in \boldsymbol{G}^{(\omega)}$ **do**
  if $\exists \boldsymbol{b}_j \in \boldsymbol{B}$ *where* $\gamma_{\boldsymbol{g}_i} < \gamma_{\boldsymbol{b}_j}$ **then**
    Find nearest eligible $\boldsymbol{b}_q$ with $\gamma_{\boldsymbol{g}_i} < \gamma_{\boldsymbol{b}_q}$;
    Apply FCM to split $\boldsymbol{b}_q$ into two sub-granular balls: $\boldsymbol{b}_\alpha$ and $\boldsymbol{b}_\beta$;
    Compute $\zeta_{\boldsymbol{b}_q}$, $\zeta_{\boldsymbol{b}_\alpha}$, $\zeta_{\boldsymbol{b}_\beta}$ and $\varepsilon_{\boldsymbol{b}_q}$;
    if $\zeta_{\boldsymbol{b}_q} \geq \varepsilon_{\boldsymbol{b}_q}$ **then**
      `// Expansion operation`
      $\boldsymbol{\Phi}_q \leftarrow \boldsymbol{\Phi}_q \cup \boldsymbol{\Theta}_i$;
      Update $\boldsymbol{b}_q$'s other MBF attributes using current $\boldsymbol{\Phi}_q$;
    else
      `// Splitting operation`
      $\boldsymbol{B} \leftarrow \boldsymbol{B} \setminus \{\boldsymbol{b}_q\}$;
      $l \leftarrow l - 1$;
      Enqueue $\boldsymbol{b}_\alpha$ and $\boldsymbol{b}_\beta$ into $\boldsymbol{Q}$;
      while $\boldsymbol{Q}$ *is not empty* **do**
        Dequeue micro ball $\boldsymbol{b}$ from $\boldsymbol{Q}$;
        Apply FCM to split $\boldsymbol{b}$ into two sub-micro balls: $\boldsymbol{b}_{\alpha'}$ and $\boldsymbol{b}_{\beta'}$;
        Compute $\zeta_{\boldsymbol{b}}$, $\zeta_{\boldsymbol{b}_{\alpha'}}$, $\zeta_{\boldsymbol{b}_{\beta'}}$;
        Compute $\varepsilon_{\boldsymbol{b}}$;
        if $\zeta_{\boldsymbol{b}} < \varepsilon_{\boldsymbol{b}}$ **then**
          Enqueue $\boldsymbol{b}_{\alpha'}$ and $\boldsymbol{b}_{\beta'}$ into $\boldsymbol{Q}$;
        else
          $\boldsymbol{b}_l \leftarrow \boldsymbol{b}$;
          Compute MBF attributes of $\boldsymbol{b}_l$;
          $\boldsymbol{B} \leftarrow \boldsymbol{B} \cup \{\boldsymbol{b}_l\}$;
          $l \leftarrow l + 1$;
        end
      end
    end
  else
    `// Generation operation`
    Initialize new micro ball $\boldsymbol{b}_l$ from $\boldsymbol{g}_i$;
    Compute MBF attributes of $\boldsymbol{b}_l$;
    $\boldsymbol{B} \leftarrow \boldsymbol{B} \cup \{\boldsymbol{b}_l\}$;
    $l \leftarrow l + 1$;
  end
end
return $\boldsymbol{B}$

Algorithm 3: MB Pruning Algorithm.

Input: current micro balls $\boldsymbol{B}$, expired data points $\check{\boldsymbol{X}}^{(\omega)}$, activity threshold τ
Output: updated micro balls $\boldsymbol{B}$

foreach $\boldsymbol{b}_j \in \boldsymbol{B}$ **do**
  $\boldsymbol{E}_j^{(\omega)} \leftarrow \check{\boldsymbol{X}}^{(\omega)} \cap \boldsymbol{b}_j$;
  if $\boldsymbol{E}_j^{(\omega)} \neq \emptyset$ **then**
    if $|\boldsymbol{b}_j| = |\boldsymbol{E}_j^{(\omega)}|$ **then**
      // Removal operation
      $\boldsymbol{B} \leftarrow \boldsymbol{B} \setminus \{\boldsymbol{b}_j\}$;
    else
      // Shrinkage operation
      $\boldsymbol{\Phi}_j \leftarrow \boldsymbol{\Phi}_j \setminus \boldsymbol{E}_j^{(\omega)}$;
      Update $\boldsymbol{b}_q$'s other MBF attributes using current $\boldsymbol{\Phi}_j$;
    end
  end
end
foreach $\boldsymbol{b}_j \in \boldsymbol{B}$ **do**
  Compute $\eta_{\boldsymbol{b}_j}^{(\omega)}$ via Eq. (12);
  if $\eta_{\boldsymbol{b}_j}^{(\omega)} < \tau$ **then**
    // Elimination operation
    $\boldsymbol{B} \leftarrow \boldsymbol{B} \setminus \{\boldsymbol{b}_j\}$;
  end
end
return $\boldsymbol{B}$

To handle data removal caused by the sliding window mechanism, the model performs shrinkage and removal operations, ensuring MBs maintain their structural validity. Specifically, each time the window slides forward, all existing MBs are examined to identify whether any of their constituent data points have been discarded. If only a subset of data points is discarded, the MB undergoes a shrinkage operation that updates its MBF attributes (e.g., the data subset, center, radius, etc.) based on the remaining valid points. If all data points within a MB have been removed, it is promptly deleted through a removal operation.

To manage outdated but not yet structurally invalid MBs, the model introduces an elimination operation guided by a window-aware activity score. Unlike removal operations triggered by data expiration, this operation targets MBs that may still contain data but have become temporally irrelevant due to prolonged inactivity. Specifically, after each window slide, the model recalculates the activity score of every existing MB using the proposed window-aware temporal decay mechanism. The updated score is then compared against a predefined threshold τ: if the score exceeds τ, the MB is retained as active; otherwise, it is eliminated from the system. This selective elimination strategy improves its responsiveness to sudden pattern shifts by prioritizing MBs with recent contributions while phasing out inactive ones.

The operations induced by data expiration (removal, shrinkage, and elimination) constitute the MB pruning algorithm, whose complete procedure is specified in Algorithm 3.

D. Clustering

Traditional offline clustering methods typically require access to the entire dataset, leading to high computational costs. To address this issue, our method's final clustering step utilizes microballs (i.e., micro balls' centers) as input to the DPC algorithm, rather than using all data points.

Algorithm 4: DPC on MBs.

Input: micro-ball set $\boldsymbol{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_L\}$ with centers $\{\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_L\}$
Output: cluster assignment vector $\boldsymbol{C} \in \mathbb{N}^L$

// Density peak selection
Compute distance matrix $\boldsymbol{D} \in \mathbb{R}^{L \times L}$ where $D_{ij} = \|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|$;
Calculate local densities $\boldsymbol{\rho} \in \mathbb{R}^L$: $\rho_l = \sum_j \exp(-(D_{lj}/\sigma)^2)$;
Determine minimum distances $\boldsymbol{\delta} \in \mathbb{R}^L$: $\delta_l = \min_{j:\rho_j > \rho_l}(D_{lj})$;
Identify cluster centers as $\{\boldsymbol{\phi}_k\}$ with both $\rho_l > \rho_{thre}$ and $\delta_l > \delta_{thre}$;
// Cluster assignment
Initialize $\boldsymbol{C} \leftarrow \boldsymbol{0}$;
for $l \leftarrow 1$ **to** L **do**
  if $\boldsymbol{\phi}_l$ *is a cluster center* **then**
    $C_l \leftarrow$ new cluster ID;
  end
  else
    $C_l \leftarrow C_{j^*}$ where $j^* = \text{argmin}_{j:\rho_j > \rho_l} D_{lj}$;
  end
end
return $\boldsymbol{C}$;

TABLE I
DATASETS USED IN EXPERIMENTS

Data Streams		#Instances	#Features	#Clusters
Synthetic	Benchmark1	5,500	2	2
	Benchmark2_1	10,000	2	3
	Benchmark2_2	20,000	2	3
	R_1	35,946	2	3
	R_2	33,976	2	7
	R_3	35,233	2	7
Real-world	Powersupply	29,928	2	24
	NOAAweather	18,159	8	2
	Electricity	45,312	6	2
	Rialto	82,250	27	10
	Covertype	581,012	54	7
	Sensor	2,219,803	5	54

As shown in Algorithm 4, the clustering process is divided into two stages: the density peak selection stage identifies representative cluster centers from the microball centers using a combined density-distance criterion, and the cluster assignment stage propagates labels based on the nearest high-density associations. This approach not only reduces computational complexity but also mitigates the negative effects of density variations, enabling more efficient and robust identification of cluster centers.

E. Complexity Analysis

The computational complexity of the proposed clustering framework can be analyzed by examining the major procedures, including window-guided data granulation, MB growth,

MB pruning, and DPC. The window-guided data granulation, which recursively splits data points into GB using a two-class FCM algorithm, exhibits a complexity of $O(S\log(K)MI)$ per window, where S is the number of new data points, M is the dimensionality, K is the number of GB, and I is the number of FCM iterations. The subsequent MB growth involves operations, such as generation, expansion, and potential recursive splitting, contributing $O(LKM)$, where L is the number of MB. It is worth noting that MB splitting events occur very infrequently and can be regarded as negligible. The MB pruning, which processes expired data and inactive MBs, operates in $O(LM)$ per window. Finally, the DPC applied to L MB centers incurs $O(L^2\,M)$ due to pairwise distance computations. By combining the above analysis, the overall complexity of the proposed method is dominated by the most expensive steps, resulting in $O(S\log(K)MI + L^2\,M)$ per window update.

IV. Experimental Results

A. Experimental Setup

To evaluate the performance of the AFGBStream method, we conduct experiments on different types of datasets and compare it with several stream clustering methods. The experiments are carried out on a Lenovo Y9000P personal computer equipped with a 3rd Gen Intel Core i9-13900HX processor and 16 GB of RAM. All experiments are implemented in Python 3.7 and involve both the AFGBStream method and other comparison methods.

1) Datasets: In this study, we use six synthetic datasets and six real datasets to comprehensively evaluate the performance of AFGBStream (including the noisy dataset), as summarized in Table I and Table IV. Due to space limitations, descriptions of the synthetic and real datasets are provided in the Supplementary Material. The distributions of the six synthetic datasets are shown in Fig. 2 .

2) Parameter Setup: In the parameter sensitivity analysis of the AFGBStream method, we focus on four key parameters: window size (W), slide step (S), decay rate (λ), and MB removal threshold (τ), and investigate their impact on the clustering performance. A detailed analysis of the influence of each parameter is provided, along with recommendations for their appropriate configurations. All parameter ranges for the methods are listed in the supplementary materials (see Figs. S1 and S2 of the Supplementary Material). A detailed analysis of the influence of each parameter is provided, along with recommendations for their appropriate configurations.

3) Comparison Methods: In the experiments, we select several mainstream stream clustering methods as baselines, including DenForest [12] and EmCStream [16] (based on sliding windows), DenStream [6] (based on density), D-Stream [13][14] (based on grids), and GB-FuzzyStream [27] (a fuzzy clustering method based on GB). DenStream is sensitive to input parameters, so we determine its optimal parameter configuration through grid search. For EmCStream, we set the expected number of clusters k to match the actual number of classes.

4) Evaluation Metrics: To evaluate clustering performance, we use Adjusted Mutual Information (AMI), Rand Index (RI), and Purity as evaluation metrics [29]. The clustering quality of AFGBStream is obtained by averaging the clustering results over all sliding windows. The results of all baseline methods are averaged over multiple runs to reduce the influence of randomness.

B. Comparison Experiments

Table II compares the performance of different methods across various datasets, with bold values indicating the best results for each metric. Overall, AFGBStream outperforms the baseline methods in terms of AMI, RI, and Purity on most datasets. Fig. 3 illustrates its clustering performance on selected synthetic datasets, showing that AFGBStream can effectively reveal the true class distributions.

In the Bench1 dataset, classes move along the diagonal in a 2-D space and overlap in the central region. Fig. 3(a)–(c) shows the clustering results of AFGBStream on this dataset, demonstrating its ability to accurately track their movement and effectively separate the overlapping classes. In contrast, other methods struggle to distinguish overlapping classes, leading to lower AMI and RI scores (see Table II).

The Bench2 dataset consists of two static Gaussian-distributed classes and one dynamically moving class, where slight interclass overlap increases clustering difficulty. As shown in Fig. 3(d)–(f), AFGBStream maintains clear class separation even with increasing overlap, demonstrating its handling of complex distributions.

The R-series dataset simulates a dynamically evolving data stream environment with partial class overlap, posing a more complex clustering challenge. Fig. 3(g)–(i) illustrates the performance of AFGBStream on the R series datasets, where it not only accurately identifies the three true class distributions in R_1 but also maintains stable clustering performance under class addition, deletion, and positional changes.

On real-world datasets, AFGBStream also exhibits strong performance. On the Powersupply dataset, it achieves the highest AMI, RI, and Purity scores. On the NOAAweather dataset, although its RI is slightly lower than that of D-Stream, it still attains the highest AMI and Purity. On the Electricity dataset, AFGBStream ranks first across all evaluation metrics. On the Covertype, Rialto, and Sensor datasets, AFGBStream continues to deliver high clustering quality, with an AMI significantly surpassing all competing methods. In addition, we evaluate the runtime comparison of the related sliding window methods, which can be found in the Supplementary Materials (see Table S1 and Fig. S3 of the Supplementary Material).

In summary, the experimental results on both synthetic and real-world datasets demonstrate that AFGBStream outperforms baseline methods in dynamic class identification and overlapping class discrimination, demonstrating its effectiveness and robustness for data stream clustering tasks.

C. Ablation Experiments

To evaluate the contribution of each key component in the proposed AFGBStream framework, we perform ablation studies by constructing three simplified variants: w/o DG (removing the

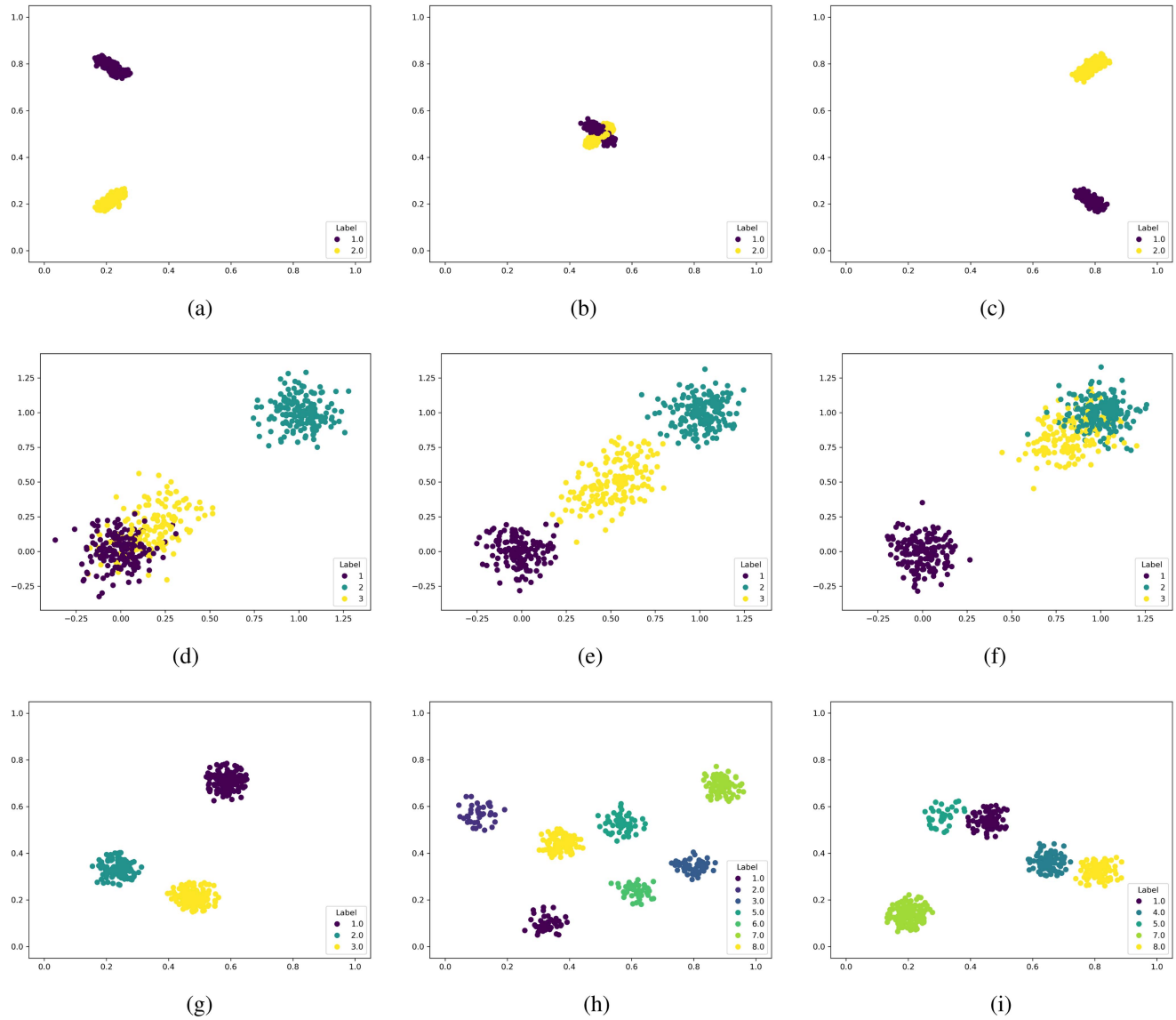

Fig. 2. Distributions of synthetic datasets. (a) Bench1. (b) Bench1. (c) Bench1. (d) Bench2_1. (e) Bench2_1. (f) Bench2_1. (g) R_1. (h) R_2. (i) R_3.

window-guided data granulation strategy), w/o FCM (replacing FCM with a hard partitioning algorithm), and w/o TD (removing the window-aware temporal decay mechanism). The clustering results of the full model and its ablated variants across multiple datasets are reported in Table III, with the best scores in each setting highlighted in bold.

After removing the window-guided data granulation strategy (w/o DG), the clustering process operates directly on the raw data points within each sliding window using the DPC algorithm. This variant shows a noticeable decline in performance, highlighting the importance of the GB module in enhancing clustering quality. The performance deterioration occurs because DPC is more vulnerable to parameter sensitivity, noise interference, and irregular cluster shapes, and often struggles to cope with overlapping clusters. By abstracting and smoothing local structures, the GB mechanism significantly improves the stability and robustness of the overall framework.

After removing the FCM mechanism, we introduce two variants: w/o FCM, which employs a geometry-based hard partitioning strategy, and w/o FCM_2, which utilizes K-means to generate GB. Compared with the full model, both variants perform significantly worse on datasets with overlapping or ambiguous class boundaries. This demonstrates that the fuzzy partitioning ability of FCM is essential for generating robust GB in our method, effectively improving clustering performance.

Finally, after removing the window-aware temporal decay mechanism (w/o TD), the method ceases to compute activation scores for MBs and no longer eliminates inactive ones. The results show that this variant performs worse, especially on data streams with frequent changes, underscoring the crucial role of the window-aware temporal decay mechanism. The performance degradation arises from accumulating obsolete MBs that distort cluster structures while failing to properly weight recent data or detect emerging stream patterns. The temporal decay

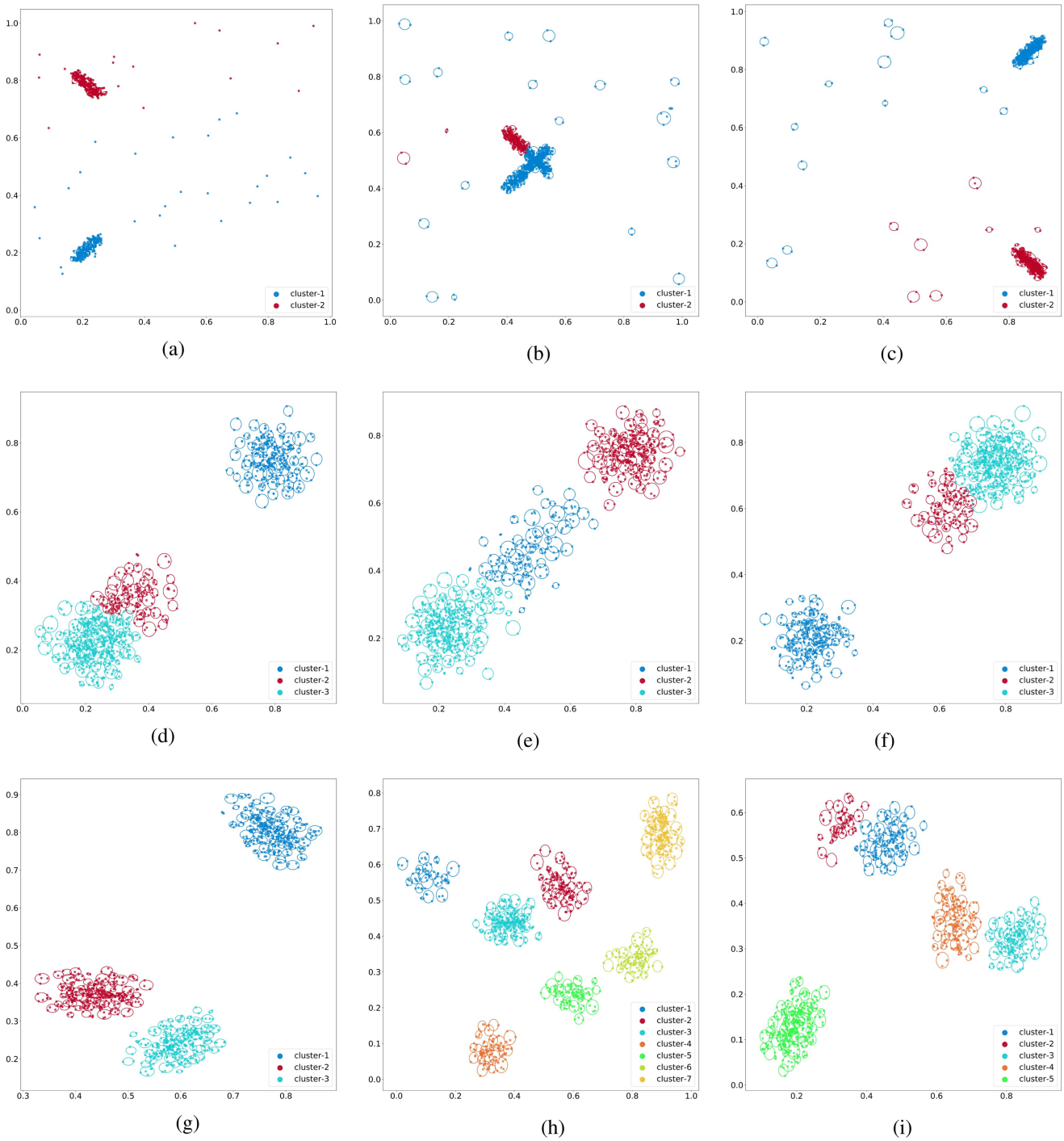

Fig. 3. Clustering of synthetic datasets. (a) Bench1. (b) Bench1. (c) Bench1. (d) Bench2_1. (e) Bench2_1. (f) Bench2_1. (g) R_1. (h) R_2. (i) R_3.

mechanism addresses these issues by automatically phasing out outdated information while preserving relevant historical context, thereby maintaining the framework's responsiveness to stream evolution.

The ablation studies collectively demonstrate that each component plays a vital role in enhancing the adaptability, stability, and overall effectiveness of the proposed stream clustering framework.

D. Noise Robustness Experiments

To evaluate the robustness of different stream clustering methods in noisy environments, we introduce noise points into the R-series and real-world datasets (R_1, R_2, R_3, Powersupply, NOAAweather; see Table I) to construct corresponding noisy versions: R_1_N, R_2_N, R_3_N, Powersupply_N, and NOAAweather_N, as shown in Table IV.

TABLE II
PERFORMANCE COMPARISON OF METHODS ON SYNTHETIC AND REAL-WORLD DATASETS

Algorithm	Metric	Bench1	Bench2_1	Bench2_2	R_1	R_2	R_3	Powersupply	NOAAweather	Electricity	Rialto	Covertype	Sensor
GB-FuzzyStream	AMI	0.649	0.500	0.474	0.724	0.739	0.686	0.205	0.008	0.043	0.256	0.142	0.309
	RI	0.817	0.746	0.734	0.876	0.895	0.868	0.802	0.507	0.513	0.627	0.558	0.616
	Purity	0.840	0.733	0.719	0.888	0.842	0.801	0.146	0.694	0.623	0.311	0.675	0.132
Denstream	AMI	0.646	0.726	0.728	0.799	0.807	0.747	0.240	0.042	0.058	0.141	0.152	0.150
	RI	0.850	0.826	0.825	0.884	0.897	0.881	0.733	0.464	0.511	0.559	0.551	0.310
	Purity	0.873	0.821	0.818	0.908	0.846	0.852	0.137	0.692	0.663	0.203	0.664	0.049
D-stream	AMI	0.000	0.031	0.016	0.000	0.000	0.000	0.000	0.000	0.000	0.022	0.002	0.009
	RI	0.423	0.352	0.343	0.334	0.239	0.226	0.041	0.526	0.519	0.188	0.482	0.214
	Purity	0.470	0.366	0.358	0.350	0.294	0.264	0.042	0.691	0.582	0.350	0.597	0.031
Denforest	AMI	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000
	RI	0.423	0.334	0.333	0.333	0.235	0.226	0.041	0.572	0.519	0.091	0.481	0.016
	Purity	0.481	0.384	0.382	0.381	0.307	0.291	0.042	0.687	0.583	0.100	0.597	0.040
EmCStream	AMI	0.046	0.017	0.084	0.013	0.028	0.017	0.001	0.005	0.000	0.005	0.001	0.012
	RI	0.513	0.564	0.591	0.562	0.613	0.608	0.648	0.481	0.497	0.611	0.538	0.638
	Purity	0.500	0.416	0.469	0.409	0.276	0.266	0.050	0.680	0.567	0.120	0.491	0.039
AFGBStream	AMI	**0.907**	**0.801**	**0.788**	**0.945**	**0.936**	**0.883**	**0.291**	**0.062**	**0.105**	**0.480**	**0.214**	**0.344**
	RI	**0.960**	**0.868**	**0.870**	**0.970**	**0.965**	**0.946**	**0.805**	**0.536**	**0.530**	**0.717**	**0.577**	**0.649**
	Purity	**0.973**	**0.850**	**0.855**	**0.974**	**0.947**	**0.925**	**0.182**	**0.716**	**0.674**	**0.469**	**0.712**	**0.136**

TABLE III
ABLATION EXPERIMENT

Algorithm	Metric	Bench1	Bench2_1	Powersupply	NOAAweather
w/o DG	AMI	0.3644	0.7135	0.1219	0.0414
	RI	0.6288	0.7954	0.2828	0.5411
	Purity	0.7021	0.7754	0.1686	0.6951
w/o FCM	AMI	0.579	0.733	0.283	0.028
	RI	0.770	0.832	0.794	0.539
	Purity	0.845	0.824	0.168	0.688
w/o FCM_2	AMI	0.522	0.731	0.278	0.026
	RI	0.739	0.828	0.779	0.541
	Purity	0.817	0.809	0.163	0.692
w/o TD	AMI	0.622	0.735	0.276	0.040
	RI	0.807	0.835	0.777	0.546
	Purity	0.865	0.832	0.162	0.703
AFGBStream	AMI	**0.907**	**0.780**	**0.291**	**0.045**
	RI	**0.960**	**0.863**	**0.805**	**0.549**
	Purity	**0.973**	**0.859**	**0.182**	**0.706**

TABLE IV
DATASETS USED IN THE NOISE EXPERIMENT

Data Streams	Real/Synthetic	#Instances	#Features	#Clusters
R_1_N	S	40000	2	3
R_2_N	S	40000	2	7
R_3_N	S	40000	2	7
Powersupply_N	R	34117	2	24
NOAAweather_N	R	20701	8	2

Table V presents the experimental results of all methods on both the original and noisy datasets. AFGBStream achieves RI scores of 0.96, 0.942, and 0.935 on R_1_N, R_2_N, and R_3_N, respectively, showing only slight decreases compared to the corresponding nonnoisy datasets (0.97, 0.965, and 0.946). AFGBStream also maintains stable performance in terms of AMI and Purity, indicating strong noise resilience. As observed in Table V, the clustering performance of all baseline methods declines to varying degrees after the addition of noise. Particularly, GB-FuzzyStream, DenStream, and EmCStream exhibit notable declines across AMI, RI, and Purity metrics when tested on noisy datasets (R_1_N, R_2_N, and R_3_N), indicating that noise adversely affected their clustering performance. Notably, while GB-FuzzyStream and DenStream show relatively stable RI scores, their AMI scores drop significantly on datasets with high class overlap, such as R_2_N. When tested on real-world datasets (Powersupply_N and NOAAweather_N), AFGBStream exhibits only a slight decline in each metric, maintaining stable performance.

Furthermore, D-Stream and DenForest exhibit unsatisfactory performance on the original datasets, and their clustering quality degrades significantly with added noise, reaching zero AMI scores in several cases. This indicates that these methods lack effective mechanisms for handling noise and exhibit insufficient robustness.

In summary, the experimental results demonstrate that AFGBStream exhibits the strongest robustness among all compared methods when facing noisy data streams. This resilience can be attributed to the structural characteristics of GB, which mitigate the influence of local noise through collective decision-making.

TABLE V
NOISE COMPARISON TEST

Algorithm	Metric	R_1_N	R_1	R_2_N	R_2	R_3_N	R_3	Powersupply_N	Powersupply	NOAAweather_N	NOAAweather
GB-FuzzyStream	AMI	0.682	0.724	0.689	0.739	0.663	0.686	0.071	0.205	0.001	0.008
	RI	0.870	0.876	0.880	0.895	0.871	0.868	0.790	0.802	0.413	0.507
	Purity	0.830	0.888	0.801	0.842	0.774	0.801	0.131	0.146	0.602	0.694
Denstream	AMI	0.712	0.799	0.695	0.807	0.665	0.747	0.203	0.240	0.003	0.042
	RI	0.891	0.884	0.879	0.897	0.879	0.881	0.725	0.733	0.460	0.464
	Purity	0.896	0.908	0.798	0.846	0.837	0.852	0.120	0.137	0.568	0.692
D-stream	AMI	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	RI	0.280	0.334	0.200	0.239	0.192	0.226	0.005	0.041	0.496	0.526
	Purity	0.314	0.350	0.264	0.294	0.239	0.264	0.001	0.042	0.592	0.691
Denforest	AMI	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	RI	0.279	0.333	0.200	0.235	0.193	0.226	0.037	0.041	0.456	0.572
	Purity	0.345	0.381	0.279	0.307	0.264	0.291	0.012	0.042	0.604	0.687
EmCStream	AMI	0.002	0.013	0.128	0.028	0.101	0.017	0.000	0.001	0.004	0.005
	RI	0.557	0.562	0.628	0.613	0.615	0.608	0.643	0.648	0.499	0.481
	Purity	0.351	0.409	0.495	0.276	0.318	0.266	0.123	0.050	0.597	0.680
AFGBStream	AMI	**0.906**	**0.945**	**0.875**	**0.936**	**0.858**	**0.883**	**0.222**	**0.291**	**0.018**	**0.062**
	RI	**0.960**	**0.970**	**0.942**	**0.965**	**0.935**	**0.946**	**0.783**	**0.805**	**0.503**	**0.536**
	Purity	**0.954**	**0.974**	**0.898**	**0.947**	**0.886**	**0.925**	**0.155**	**0.182**	**0.617**	**0.716**

V. CONCLUSION

This article presents AFGBStream, a sliding window-based data stream clustering method that integrates an adaptive incremental update framework built upon sliding windows and GB. In the data ingestion phase, we introduce a window-guided data granulation strategy that effectively alleviates the sensitivity to microcluster radius parameters observed in existing methods, thereby improving the robustness of data representation. During the model update phase, a window-aware temporal decay mechanism is incorporated to strengthen the model's responsiveness to sudden pattern changes in evolving data streams. Experimental results across multiple benchmarks demonstrate the effectiveness, adaptability, and robustness of AFGBStream.

In the future, we aim to utilize a center consistency metric based on GB to enable efficient and flexible representation and coverage of the sample space. In addition, we plan to extend AFGBStream to distributed or federated learning settings, facilitating scalable streaming clustering for large-scale, decentralized data sources.

REFERENCES

[1] J. Sun, M. Du, C. Sun, and Y. Dong, "Efficient online stream clustering based on fast peeling of boundary micro-cluster," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 3, pp. 5680–5693, Mar. 2025.

[2] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2057–2068, 2011.

[3] Y. Li, M. Chi, W. Lu, X. Liu, and W. Pedrycz, "Data stream clustering via fuzzy similarity and diffusion-enhanced contextual affinity," *Inf. Sci.*, vol. 723, 2026, Art. no. 122690.

[4] J. Sun, M. Du, Z. Lew, and Y. Dong, "TWStream: Three-way stream clustering," *IEEE Trans. Fuzzy Syst.*, vol. 32, no. 9, pp. 4927–4939, Sep. 2024.

[5] A. Urio-Larrea et al., "Data stream clustering: Introducing recursively extendable aggregation functions for incremental cluster fusion processes," *IEEE Trans. Cybern.*, vol. 55, no. 3, pp. 1421–1435, Mar. 2025.

[6] F. Cao, M. Ester, W. Qian, and A. zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SIAM Int. Conf. Data Mining*, 2006, pp. 328–339.

[7] J. Youn, J. Shim, and S. G. Lee, "Efficient data stream clustering with sliding windows based on locality-sensitive hashing," *IEEE Access*, vol. 6, pp. 63757–63776, 2018.

[8] B. Kim, K. Koo, J. Kim, and B. Moon, "DISC: Density-based incremental clustering by striding over streaming data," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 828–839.

[9] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Adaptive clustering for dynamic IoT data streams," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 64–74, Feb. 2017.

[10] M. Hahsler and M. Bolaños, "Clustering data streams based on shared density between micro-clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1449–1461, Jun. 2016.

[11] D. Cheng, J. Huang, S. Zhang, X. Zhang, and X. Luo, "A novel approximate spectral clustering algorithm with dense cores and density peaks," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 4, pp. 2348–2360, Apr. 2022.

[12] B. Kim, K. Koo, U. Enkhbat, and B. Moon, "DenForest: Enabling fast deletion in incremental density-based clustering over sliding windows," in *Proc. 2022 Int. Conf. Manag. Data*, 2022, pp. 296–309.

[13] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2007, pp. 133–142.

[14] L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 3, 2009,Art. no. 12.

[15] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 685–694.

[16] A. Zubaroğlu and V. Atalay, "Online embedding and clustering of evolving data streams," *Statist. Anal. Data Mining: ASA Data Sci. J.*, vol. 16, no. 1, pp. 29–44, 2023.

[17] J. Liu, M. Wu, M. Lin, and Z. Xu, "Anisotropic-Gaussian-kernel-based fuzzy clustering algorithm for feature selection," *IEEE Trans. Fuzzy Syst.*, vol. 33, no. 9, pp. 3061–3075, Sep. 2025.

[18] X. Shuyin, L. Yunsheng, D. Xin, W. Guoyin, Y. Hong, and L. Yuoguo, "Granular ball computing classifiers for efficient, scalable and robust learning," *Inf. Sci.*, vol. 483, pp. 136–152, 2019.

[19] S. Xia et al., "Ball k-means: Fast adaptive clustering with no bounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 87–99, Jan. 2022.

[20] S. Xia, J. Xie, and G. Wang, "Gbc: An efficient and adaptive clustering algorithm based on granular-ball," 2022, *arXiv:2205.14592*.

[21] J. Yang et al., "Constructing three-way decision with fuzzy granular-ball rough sets based on uncertainty invariance," *IEEE Trans. Fuzzy Syst.*, vol. 33, no. 6, pp. 1781–1792, Jun. 2025.

[22] J. Yang et al., "3WC-GBNRS : A novel three-way classifier with granular-ball neighborhood rough sets based on uncertainty," *IEEE Trans. Fuzzy Syst.*, vol. 32, no. 8, pp. 4376–4387, Aug. 2024.

[23] J. Yang et al., "CS3W-GBG: A cost-sensitive three-way granular-ball generation method," *IEEE Trans. Fuzzy Syst.*, vol. 33, no. 10, pp. 3681–3694, Oct. 2025.

[24] D. Cheng, Y. Li, S. Xia, G. Wang, J. Huang, and S. Zhang, "A fast granular-ball-based density peaks clustering algorithm for large scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 12, pp. 17202–17215, Dec. 2024.

[25] J. Xie, W. Kong, S. Xia, G. Wang, and X. Gao, "An efficient spectral clustering algorithm based on granular-ball," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9743–9753, Sep. 2023.
[26] S. Xia, B. Shi, Y. Wang, J. Xie, G. Wang, and X. Gao, "GBCT: Efficient and adaptive clustering via granular-ball computing for complex data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 7, pp. 12159–12172, Jul. 2025.
[27] J. Xie, M. Dai, S. Xia, J. Zhang, G. Wang, and X. Gao, "An efficient fuzzy stream clustering method based on granular-ball structure," in *Proc. IEEE 40th Int. Conf. Data Eng.*, 2024, pp. 901–913.
[28] G. Lang, L. Zhao, D. Miao, and W. Ding, "Granular-ball computing-based fuzzy twin support vector machine for pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 33, no. 7, pp. 2148–2160, Jul. 2025.
[29] D. Cheng, J. Huang, S. Zhang, S. Xia, G. Wang, and J. Xie, "K-means clustering with natural density peaks for discovering arbitrary-shaped clusters," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 8, pp. 11077–11090, Aug. 2024.

Qijia Wang received the bachelor's degree in computer science from Suzhou City University, Suzhou, China, in 2021. He is currently working toward the master's degree in electronic information from the School of Artificial Intelligence and Computer Science, Jiangsu Normal University, Xuzhou, China.

His research interests include Granular-ball computing, clustering, and data stream.

Mingjing Du (Member, IEEE) received the Ph.D. degree in computer science from the China University of Mining and Technology, Xuzhou, China, in 2018.

He is currently an Associate Professor with the School of Artificial Intelligence and Computer Science, Jiangsu Normal University, Xuzhou, China. His current research interests include cluster analysis and three-way decisions.

Weiping Ding (Senior Member, IEEE) received the Ph.D. degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013.

From 2014 to 2015, he was a Postdoctoral Researcher with the Brain Research Center, National Chiao Tung University, Hsinchu, Taiwan. In 2016, he was a Visiting Scholar with the National University of Singapore, Singapore. From 2017 to 2018, he was a Visiting Professor with the University of Technology Sydney, Ultimo, NSW, Australia. He is currently the Full Professor with Nantong University, Nantong, China. He has authore or coauthored more than 480 articles, including more than 260 IEEE Transactions papers, with more than 24 000 citations and an h-index of 75+ (Google Scholar, May 2026). His 25 authored/coauthored papers have been selected as ESI Highly Cited Papers. His main research interests include granular data mining and multimodal machine learning.

Dr. Ding was featured on the "World's Top 2% Scientists List" every year from 2020 to 2025. He serves/served as an Associate Editor/Area Editor/Editorial Board member of more than ten international prestigious journals, such as IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE/CAA Journal of Automatica Sinica,IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *Information Fusion*, *Neurocomputing*, *Applied Soft Computing*. He was the Leading Guest Editor for Special Issues in several prestigious journals, including IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON FUZZY SYSTEMS, and *Information Fusion*.