Data stream clustering: introducing recursively extendable aggregation functions for incremental cluster fusion processes

A. Urio-Larrea *IEEE Member*, H. Camargo *IEEE Member*, G. Lucca, T. Asmus, C. Marco-Detchart, L. Schick, C. Lopez-Molina, J. Andreu-Perez *Senior Member, IEEE*, H. Bustince *Fellow, IEEE*, G.P. Dimuro *IEEE Member*

***Abstract*—In Data Stream (DS) learning, the system has to extract knowledge from data generated continuously, usually at high speed and in large volumes, making it impossible to store the entire set of data to be processed in batch mode. Hence, machine learning models must be built incrementally by processing the incoming examples, as data arrive, while updating the model to be compatible with the current data. In fuzzy DS clustering, the model can either absorb incoming data into existing clusters or initiate a new cluster. As the volume of data increases, there is a possibility that the clusters will overlap to the point where it is convenient to merge two or more clusters into one. Then, a cluster comparison measure (CM) should be applied, to decide whether such clusters should be combined, also in an incremental manner. This defines an incremental fusion process based on aggregation functions that can aggregate the incoming inputs without storing all the previous inputs. The objective of this paper is to solve the fuzzy DS clustering problem of incrementally comparing fuzzy clusters on a formal basis. Firstly, we formalize and operationalize incremental fusion processes of fuzzy clusters by introducing Recursively Extendable (RE) aggregation functions, studying construction methods and different classes of such functions. Secondly, we propose two approaches to compare clusters: similarity and overlapping between clusters, based on RE aggregation functions. Finally, we analyze the effect of those incremental CMs on the online and offline phases of the well-known fuzzy clustering algorithm d-FuzzStream, showing that our new approach outperforms the original algorithm and presents better or comparable performance to other state-of-the-art DS clustering algorithms found in the literature.**

***Index Terms*—Data streams, fuzzy clustering, similarity measures, overlap indices, aggregation functions.**

I. Introduction

LEARNING from Data Streams (DSs) is a research field aiming to extract knowledge from data that are generated continuously and may have changes in their distribution over time. In many applications, data are generated at high speed and in large volumes, making it impossible to store entire datasets to be processed in batch mode [1]. Traditional Machine Learning (ML) strategies must be adapted to address the challenges posed by these dynamic data sources. The main supervised and unsupervised ML tasks have been addressed in the context of DSs. See [2] for a survey of methods, applications, and open problems, and also [3], [4] and the references therein, for recently published works.

In DSs analysis, since data arrive continuously and cannot be stored, ML models must be built incrementally. A common approach for the processing of data is to analyze the incoming examples one by one, updating an initial model to make it compatible with the current data. An important feature of such methods is being able to identify whether incoming data is (a) represented by the current structure that should, therefore, only be updated by absorbing this new data or (b) a novelty that should cause some modification in the cluster structure. Additionally, ML models for DSs ideally have the ability to forget representations extracted from old data, which shall no longer be compatible with current data. If the model is rigid and does not allow for such changes, it might eventually become incapable of performing its task properly.

As DSs pose additional challenges regarding labelling cost, clustering analysis attracts attention for extracting knowledge in an unsupervised way. Relevant applications of stream clustering can be found in fields such as power load demand management [5], network intrusion detection [6], social media [7], and IoT [8], among others.

In this work, we focus on fuzzy clustering methods for DSs [9], *i.e.* methods in which an instance may partially belong to several fuzzy clusters with different membership degrees, according to fuzzy set theory [10]. In particular, we consider the state-of-the-art *d-FuzzStream* clustering algorithm [11], which consists of two phases: the online phase, responsible for incrementally updating a model, and the offline one, which generates a snapshot clustering of the model when requested.

Regarding the online phase, due to the dynamic nature of the data, the number and size of fuzzy clusters may vary as the DS progresses. As the data and their membership degrees to the fuzzy clusters are not stored for the reasons above, the clustering is represented by a summary structure that maintains statistical components to identify each cluster. Incoming data can be absorbed by existing clusters or start a new cluster. For each arriving instance, the system shall check whether it is inside the fuzzy radius of some cluster, in which case the data is included in the existing structure, by updating the

Supported by FAPERGS (23/2551-0001865-9, 23/2551-0000126-8), CNPq (304118/2023-0, 407206/2023-0), CAPES, FAPESP (2022/09136-1), MCIN/AEI/10.13039/501000011033/FEDER,UE (PID2022-136627NB-I00) Grant Santander-UPNA.

A. Urio, C. Lopez-Molina, C. Marco-Detchart and H. Bustince are with Universidad Pública de Navarra, Spain, e-mail: {asier.urio, carlos.lopez, cedric.marco, bustince}@unavarra.es; H. Camargo and L. Schick are with Universidade Federal de São Carlos, Brasil, e-mail: heloisacamargo@ufscar.br; G. Lucca, is with Universidade Católica de Pelotas, Brazil, e-mail: giancarlo.lucca@ucpel.edu.br; T. Asmus and G.P. Dimuro are with Universidade Federal do Rio Grande, Brazil, e-mail: {tiagoasmus,gracalizdimuro}@furg.br; J. Andreu-Perez is with the University of Essex, UK, e-mail: j.andreu-perez@essex.ac.uk.

statistical components of each cluster. Otherwise, a new cluster is created. As the DS progresses, the clusters may overlap to the point where it is convenient to merge two or more clusters together. For this, after the initial processing of an instance, a comparison measure (CM) between pairs of clusters is applied to decide if they should be combined. This CM depends on the membership degrees of the examples for each fuzzy cluster. However, since examples and their membership degrees are not stored individually, cluster comparison must be done *incrementally*.

In this context, this work firstly focuses on a key issue in the definition of clustering models for DSs:

How to incrementally compare two fuzzy clusters according to the similarity or overlapping between them?

Formally, the comparison measure (CM) proposed in this work is based on an aggregation function that needs to increase its arity at each stage of the clustering process. Operationally, the calculus of this function at the n+1-th stage (i.e., when the n+1-th value is yielded by the DS) only considers the result of the previous fusion process, not the explicit representation of the previous n inputs, which are not stored. This process is referred to as "incremental fusion process".

Then, the second main focus in this work is:

How to provide an adequate axiomatization, on an operational approach, of incremental fusion processes?

This process can be easily performed by associative aggregation functions [12]. Nevertheless, there are a plethora of interesting non-associative aggregation functions for the task, e.g. overlap/grouping functions [13], which have been successfully applied in several applications (e.g, [14]). However, incremental fusion processes may not be trivially performed by such non-associative functions [15].

In the literature, some authors have studied the problem of applying functions with different possible arities. For example, Montero et al. [16] raised the problem of computational aggregation processes in a big data environment, where aggregation operators must be successfully computed by parts in a distributed parallel calculus in many nodes (mapping) before being reduced. The authors studied families of aggregation operators that can be defined through an algorithm or a function that can solve all possible declared aggregations, in particular, considering the possibility of partial reuse of previous computations when new values arrive. The functions designed according to this approach, called computable aggregation functions (CAFs), are operational, and can be subdivided by programming paradigms (e.g., hierarchical CAFs [17]). The same approach was considered to define computable aggregations of random variables [18]. Magdalena et al. [19], [20] analysed the monotonicity in non-deterministic CAFs.

The earliest notion of a family of aggregation operators (FAO) or extended aggregation functions (EAF) $\{A_n{:}[0,1]^n{\to}[0,1]\}$ appeared in the literature in 2013 (e.g., [21], [22], [23]). Rojas et al. [24] studied the strict stability of such operators, *i.e.* the property that enables the aggregation of sets of items with different dimensions in a consistent way. In 2019, Olaso et al. [25] discussed some problems of this notion, introducing a more flexible generalization of stability for FAO. Beliakov et al. [26] studied pre-EAFs with cardinality-limiting behaviour limiting the contribution of repeated inputs.

In the same line of work, Mesiar et al. [27] presented an interesting discussion on FAO, observing that, in most aggregation problems, the number of values to be aggregated cannot be fixed a priori. Then, to be able to consistently manage values within a wide cardinality range, they introduced the set-extended aggregation functions $A : \bigcup_{n\in\mathbb{N}}[0,1]^n \to [0,1]$. Although this concept allows aggregation functions to vary their arity by a family of aggregation functions, this approach does not address the characterization of incremental aggregation processes. Specifically, it is not considered the fact that n-ary aggregation functions do not necessarily need to "remember" the n initial values when adding a potential $(n+1)$-th input.

Objectives of the work

The objective of this paper is to solve the problem of incrementally comparing fuzzy clusters in fuzzy data stream clustering, on a formal basis. For that, we have the following goals:

1) To formalize and operationalize incremental fusion processes of fuzzy clusters by introducing the concept of *recursively extendable* (RE) aggregation functions, presenting construction methods and studying different classes of such functions to be applied in fuzzy data stream clustering;
2) To propose two different approaches to compare clusters in the fuzzy data stream clustering problem, namely, similarity and overlapping between clusters, in which RE aggregation functions are applied.
3) To study, by means of experiments, the behaviour of a set of measures of the types proposed in this paper within the scope of the *d-FuzzStream* clustering algorithm, analyzing their effects in the online phase, specifically in micro clusters merging, and in the offline phase, with comparisons with similar methods.

The paper is organized as follows: in Sect. II, we discuss the basic concepts regarding fuzzy data stream clustering; Section III introduces RE aggregation functions and a general construction method, studying different classes and examples; in Sect. IV, we present our new approach for fuzzy data stream clustering, introducing two methods to compare clusters (similarity and overlapping between clusters), defined in terms of RE aggregation functions; the experiments are detailed in Sections V, with the analysis of the results and the related discussions presented in Sect. VI (for the online phase) and VII (for the offline phase); Section VIII is the Conclusion. The source code, complete experiments, additional results and the table of abbreviations/notations are available at https://github.com/asieriko/REFuzzStream.

II. Fuzzy data stream clustering

In data streams, due to its continuous nature, instances are seen only once and cannot be explicitly stored. Additionally,

since the data distribution can change, the main challenge of learning from DSs is to detect changes and update the model being generated incrementally [28].

Several DS clustering models have been proposed in recent years. Chen et al. [3] presented a Two-Stage Sparse Representation Clustering (TSSRC) method for clustering high-dimensional DSs incorporating the concept of landmark window. Another work that also explores the challenges of sparse representations in DS clustering is [4], in which an algorithm was proposed to address the time-varying nature of the subspaces underlying the evolving DS. Zhang et al. [5] introduced a dynamic conditional score model to deal with time-varying multidimensional time series of power load data and improve the clustering performance. Other algorithms handling DS clustering are EvolveCluster [29], which follows a partition algorithm approach, OSRC [30], which constructs sparse representations by reducing the dimensionality, and ACSC [31], a stream version of Ant Colony Clustering.

In particular, the non-parametric Bayesian models [32], [33] are theoretically suitable for online clustering, due to their ability to relax the complexity of the model according to data observed over time and to automatically infer the number of mixture components. Although sharing the same objective as the method studied in our research, such methods are based on a very different theoretical foundation. We focus on online clustering using Fuzzy C-Means (FCM) clustering [34], since it is a simple and intuitive way of maintaining the summary structure. See [28] for a comprehensive survey of DS clustering algorithms.

Fuzzy Clustering is a type of clustering that allows a data point to belong partially to more than one cluster in different degrees, using concepts from the Fuzzy Set Theory [10]. In this theory, given a universe, X, an element $x \in X$ may partially belong to a fuzzy set $FS \subseteq X$. The function $\mu_{FS} : X \rightarrow [0, 1]$ that measures the degree to which each element $x \in X$ belongs to FS is called the membership function of FS. Hence, $\mu_{FS}(x) = 1$ (resp., $\mu_{FS}(x) = 0$) represents that the element x completely belongs to (resp., is disassociated from) the fuzzy set FS. Membership degrees $\mu_{FS}(x) \in]0, 1[$ indicate a partial membership of x to FS. For simplicity, the membership degree of an element x to a fuzzy set FS will be denoted by $FS(x)$. Fuzzy Clustering has also been extended to the context of DSs.

Among the clustering algorithms used as inspirations for dealing with DSs, partitioning-based algorithms, mainly FCM [34], were the first to be used. Also, they are the basis of most of the works so far [9]. Another category used as the basis for proposals, leading to effective approaches, is density-based clustering, e.g. the DBSCAN algorithm [35].

Since the beginning of this research field, more than two decades ago, most works have been based on two main approaches: (a) algorithms based on chunks and (b) two-step algorithms. Chunk-based algorithms use time windows on the DS data to define the chunks of examples [36] that are clustered. As the window moves forward, data from previous chunks are remembered using the cluster centroids weighted by the sum of the membership degrees of each example to the cluster, so as to initialize the centroids of the clustering

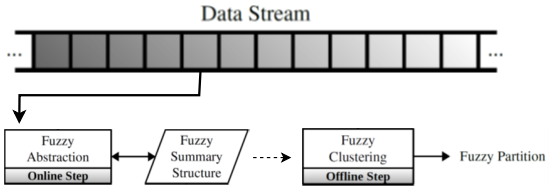


Fig. 1: *Fuzzy Online-Offline Framework* [39]

of the next chunk. In two-step algorithms, the first step of the algorithm is designed to process the input examples one by one as they arrive and, as the examples in DSs cannot be stored, to keep a summary structure containing statistics describing the examples. The statistics evolve to update the representation structure incrementally. The second step, which is performed offline, either under the request of the user or at predefined time points, performs clustering using the information in the online step.

The seminal idea of a summary structure to represent data sets in DSs was introduced by Zang et al. [37], with the proposal of a Feature Vector (FV). Aggarwal et al. [38] used and extended FV by creating *CluStream*, which is a two-step algorithm that runs in the realm of *Online-Offline Framework* (OOF) [11]. In *CluStream*, the summary structure is composed of a set of Micro-Clusters (*MiCs*), which are cluster feature vectors composed by statistics that can be used to calculate the radius and the centroid of clusters, while also storing a timestamp to represent their time relevance. Using *MiCs* is appropriate for DSs due to the low memory requirements, the possibility of being updated incrementally, and supporting an ageing process consisting of discarding the older *MiCs* when the structure reaches its maximum size. When the offline step is applied, the set of *MiCs* is converted into a set of weighted examples (one for each *MiC*) and is then clustered by a variant of the k-means algorithm, resulting in Macro-Clusters.

A. The d-FuzzStream algorithm

FuzzStream [39] is one of the first fuzzy clustering algorithms proposed as a fuzzy extension of the OOF, under the *Fuzzy Online-Offline Framework*, depicted in Fig. 1. A fuzzy summary structure consisting of a set of *Fuzzy Micro-Clusters* (*FMiCs*) is maintained during the online step and clustered in the offline step using the Weighted FCM (WFCM) algorithm [40]. A version that uses fuzzy dispersion to update the summary structure, *d-FuzzStream*, was introduced in [11].

In the context of the *d-FuzzStream* algorithm, the fuzzy dispersion of a cluster is defined as a measure based on the Root-Mean-Square Deviation that can be used to represent the radius of a fuzzy cluster [41]. Let X be a dataset, $x_j \in X$. Let $C_1, ..., C_k$ be a set of k fuzzy clusters on X and N_i the number of examples assigned to a cluster C_i. The *fuzzy dispersion* of a cluster C_i is given by $disp_i = \sqrt{\frac{\sum_{x_j \in C_i} \mu_j \|x_j - c_i\|^2}{N_i}}$, where μ_j is the membership degree of the example x_j to the cluster C_i, c_i is the prototype of C_i and $\|x_j - c_i\|$ represents the distance between the example x_j and the prototype c_i.

A *Fuzzy Cluster Similarity Matrix* (FCSM) R is a $k \times k$ matrix where each cell $R_{ij}, i, j = 1, 2, \ldots, k$ represents the

TABLE I: FMiC Structure

Component	Definition
$\overline{CF}$	Linear sum of examples weighted by their membership degrees to the FMiC
SSD	Quadratic sum of distances between the examples and the FMiC prototypes weighted by example membership degree to the FMiC
M	Sum of membership degrees of the examples assigned to the FMiC
N	Number of examples assigned to the FMiC
t	Timestamp of the last example assigned to the FMiC

similarity between fuzzy clusters C_i and C_j calculated as the ratio of the sum of the fuzzy dispersion of C_i and C_j to the distance between their prototypes [41], as:

$$R_{ij} = \frac{disp_i + disp_j}{\|c_i - c_j\|}. \quad (1)$$

As in all DS clustering methods based on OOF, in *d-FuzzStream* a summary structure is built and maintained in the online step. This structure is composed of a set of ***FMiCS***, which are represented by vectors $(\overline{CF}, SSD, M, N, t)$, whose components are in Table I.

The online step of *d-FuzzStream* consists of receiving and processing examples from the DS, one by one, to perform the maintenance of the summary structure. Since the examples themselves are not stored, the ***FMiC*** components must be updated incrementally, as each example is processed. The parameters required during this process are the fuzzification parameter m for the FCM algorithm, the minimum/maximum number of ***FMiCs*** allowed in the structure, and the merging threshold τ.

The summary structure is initially empty. The first examples in the DS are used to create ***FMiCs*** until the structure reaches the minimum number required. When the summary structure already has the minimum number of ***FMiCs***, the algorithm proceeds to evaluate whether the example is an outlier or not, calculating the Euclidean distances between the example and all ***FMiCs*** prototypes (centroid), as well as the fuzzy radius for each ***FMiC***. As ***FMiCs*** can change with processing of each example, the centroid c_i and the fuzzy radius $disp_i$ must be calculated using the components of the vector ***FMiC*** before the distance is calculated, as $c_i = \frac{\overline{CF}_i}{M_i}$ and $disp_i = \sqrt{\frac{SSD_i}{N_i}}$.

When an example falls into one or more ***FMiC*** radius, it is absorbed by the summary structure. For that, the membership degrees of the example are calculated, as in the traditional FCM algorithm, and the components of the ***FMiCs*** are updated, by simple addition and multiplication operations. If the example does not fall into the radius of any ***FMiC***, it is considered an outlier and a new ***FMiC*** is created with the example as its centroid and added to the summary structure. In this case, if the structure reaches its maximum size, the oldest ***FMiC*** is replaced by the new one. This strategy provides a drift detection capability to the summary structure.

After processing the example, the merge step is initiated. The FCSM R (Eq. (1)) is calculated for the ***FMiCs*** currently available using the ***FMiCs*** components. The criterion adopted to decide on the merge of ***FMiCs*** is the *Similarity-Driven Merging Criterion* [41]. If the R_{ij} value is greater than τ, the two ***FMiCs*** can be merged. Otherwise, the two ***FMiCs*** are considered to be separated and not similar enough to be merged. The threshold τ can assume values in the interval $[0, +\infty[$, where $\tau < 1$ considers non-overlapping ***FMiCs*** and $\tau >= 1$ only considers overlapping ***FMiCs***.

The behaviour of the online phase of *d-FuzzStream* addresses the main challenges in DS processing: (i) incremental model learning, with the continuous update of the summary structure when each new example is processed; (ii) new group discovering, with the creation of new FMiCs when the new example is found to be an outlier; and (iii) forgetfulness of old data that no longer represents the stream, with the exclusion of the oldest FMiCs when the summary structure is full. Besides that, the merging operation avoids the removal of an excessively high number of FMiCs, which would make the forgetting event happen too fast.

The offline step is executed when requested by the user. The set of ***FMiCs*** in the summary structure at this specific moment is transformed into a set of examples, formed by the centroid of each ***FMiC***, weighted by the sum of the membership degrees of the examples to this ***FMiC***. The weighted centroids are then clustered using the WFCM algorithm [40] to generate the fuzzy partition. The initial cluster prototypes are the examples with the greatest weights.

In this work, we focus on the merging decision. In [42], it was presented a previous study to experiment with different similarity-like measures between fuzzy clusters, besides the one used in the original implementation of *d-FuzzStream*. However, the selection of the adopted measures did not take into account the properties they meet, if any.

In the next section, we formalize and operationalize the incremental fusion process of fuzzy clusters (which is our first objective posed in the Introduction) by introducing the concept of recursively extendable aggregation functions, which are then used to formally define cluster CM.

III. Recursively extendable functions

As previously discussed, in online clustering, the fuzzy sets to be compared are updated in real time, usually increasing their cardinality. This means that the applied CM has to be calculated repeatedly considering the addition of new inputs, which characterizes an incremental fusion process. This increase in dimension may pose a challenge since not every CM is based on associative functions. Also, when the application consists of a large number of inputs that are added "one at a time", it is not computationally feasible to store all the inputs to calculate the CM each time a new input is added to the system.

To address this type of situation, here we introduce the concept of recursively extendable aggregation functions. In the rest of the paper, consider $\mathrm{N} = \{1, 2, \ldots\}$ and $\mathcal{N} = \{2, 3, \ldots\}$.

First, let us recall the concept of *aggregation function* [23], which is an increasing function $A{:}[0,1]^n \to [0,1]$ such that $A(0,\ldots,0){=}0$ and $A(1,\ldots,1){=}1$.

Definition 3.1: An n-ary aggregation function $M_n : [0,1]^n \to [0,1]$, with $n \in \mathcal{N}$, is said to be recursively extendable (RE) to the arity $n{+}1$ if there exists an aggregation function $F : [0,1]^2 \to [0,1]$ and a family of parametric aggregation functions $H_{(\rho)}{:}[0,1]^2 \to [0,1]$, with $\rho \in \mathcal{N}$, such that

$$\begin{aligned} M_2(x_1, x_2) &= F(x_1, x_2) \\ M_{n+1}(x_1, \ldots, x_{n+1}) &= H_{(n)}(M_n(x_1, \ldots, x_n), x_{n+1}), \end{aligned} \quad (2)$$

for all $x_1, \ldots, x_{n+1} \in [0,1]$. For simplicity, M_n is called a recursively extendable aggregation function.

Definition 3.1 is conceptualizing the property of a function M_n of being able to extend its arity from n to $n+1$, stating the conditions for such extension to take place. Obviously:

Proposition 3.1: Any n-ary associative aggregation function is also recursively extendable, but the converse may not hold.

The advantage of Def. 3.1 is that it allows for one to define an n-ary aggregation function M that is not associative in an incremental manner, aggregating the inputs "one at a time". For practical applications, this means that when a new input x_{n+1} has to be aggregated, the system does not need to access the data of all the previous inputs $(x_1, \ldots, x_n)$, only the result of the current aggregation of them, $M_n(x_1, \ldots, x_n)$, and, maybe, some parameters, like the number of inputs n that have been aggregated up until the inclusion of x_{n+1} in the aggregation process. Furthermore, this avoids the need for the system to have particular expressions of M for different values of n when M is not associative.

Example 3.1: The arithmetic mean $AM{:}[0,1]^n \to [0,1]$, given, for all $x_1, \ldots, x_n \in [0,1]$, by

$$AM(x_1, \ldots, x_n) = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i. \tag{3}$$

is an aggregation function that is not associative, but it is RE. In fact, let $F{:}[0,1]^2 \to [0,1]$ be the bivariate AM, defined, for all $x_1, x_2 \in [0,1]$, by $F(x_1, x_2) = \frac{x_1+x_2}{2}$, and $H_{(\rho)}{:}[0,1]^2 \to [0,1]$ be a family of parametric aggregation functions, with $\rho \in \mathcal{N}$, given, for all $y_1, y_2 \in [0,1]$, by $H_{(\rho)}(y_1, y_2) = \frac{\rho \cdot y_1 + y_2}{\rho+1}$. The function $AM_n{:}[0,1]^n \to [0,1]$ is RE to the arity $n+1$, since, one can define, for all $x_1, \ldots, x_{n+1} \in [0,1]$:

$$\begin{array}{rcl} AM_2(x_1, x_2) & = & F(x_1, x_2) \\ AM_{n+1}(x_1, \ldots, x_{n+1}) & = & H_{(n)}(AM_n(x_1, \ldots, x_n), x_{n+1}), \end{array}$$

which provides a definition to the n-dimensional AM in a recursive manner. Thus, AM_n is an RE aggregation function.

We present a construction method for such functions:

Theorem 3.1: For $k \in \mathrm{N}$ and $i \in \{1, \ldots, k\}$, take an aggregation function $C{:}[0,1]^k \to [0,1]$, k RE aggregation functions $F_n^i{:}[0,1]^n \to [0,1]$ and a family of parametric aggregation functions $H_{(\rho)}{:}[0,1]^2 \to [0,1]$, with $\rho \in \mathcal{N}$. The n-ary function $M_n{:}[0,1]^n \to [0,1]$ is an aggregation function that is RE to the arity $n+1$, where, for all $x_1, \ldots, x_{n+1} \in [0,1]$:

$$\begin{array}{c} M_2(x_1, x_2) = C(F_2^1(x_1, x_2), \ldots, F_2^k(x_1, x_2)) \\ M_{n+1}(x_1, \ldots, x_{n+1}) = H_{(n)}(M_n(x_1, \ldots, x_n), x_{n+1}), \text{with} \\ M_n(x_1, \ldots, x_n) = C(F_n^1(x_1, \ldots, x_n), \ldots, F_n^k(x_1, \ldots, x_n)). \end{array}$$

Proof: Since $C, F_n^1, \ldots, F_n^k$ are aggregation functions, and $H_{(\rho)}$ is a family of aggregation functions, it is immediate that M_n is also an aggregation function. Also, one may observe that the definition of $M_n : [0,1]^n \to [0,1]$ is that of an RE aggregation function (Eq. (2)), with $F(x_1, x_2) = C(F_2^1(x_1, x_2), \ldots, F_2^k(x_1, x_2))$, completing the proof. ■

Example 3.2: A well-known class of aggregation functions is that of n-dimensional overlap functions [15] $On{:}[0,1]^n \to [0,1]$, for which the following conditions should hold, for all $x_1, \ldots, x_n \in [0,1]$: **(On1)** On is symmetric; **(On2)** $On(x_1, \ldots, x_n) = 0 \Leftrightarrow \prod_{i=1}^n x_i = 0$; **(On3)** $On(x_1, \ldots, x_n) = 1 \Leftrightarrow \prod_{i=1}^n x_i = 1$; **(On4)** On is increasing and **(On5)** continuous. When $n = 2$, they are just called overlap functions [13]. Some examples of overlap functions are shown in Table II, as the geometric mean GM, the product $PROD$, the minimum MIN and OB. Observe that $PROD$ and MIN are associative, so they can be easily extended to the n-dimensional case, denoted by $PROD_n$ and MIN_n, respectively. Additionally, by Proposition 3.1, they are recursively extendable. However, OB is not associative. Its n-dimensional version is the function $OnB : [0,1]^n \to [0,1]$, given, for all $x_1, \ldots, x_n \in [0,1]$, by

TABLE II: Examples of overlap functions

Name	Definition
Product	$PROD(x,y) = x \cdot y$
Minimum	$MIN(x,y) = \min\{x,y\}$
Geometric Mean	$GM(x,y) = \sqrt{x \cdot y}$
OB Overlap	$OB(x,y) = \sqrt{xy \cdot \min\{x,y\}}$
ODIV Overlap	$ODIV(x,y) = \frac{xy + \min\{x,y\}}{2}$

TABLE III: Examples of n-dimensional grouping functions

Name	Definition
Probabilistic Sum	$PS(x_1, \ldots, x_n) = 1 - \prod_{i=1}^n (1 - x_i)$
Maximum	$MAX(x_1, \ldots, x_n) = \max\{x_1, \ldots, x_n\}$
Dual of GM	$GM^d(x_1, \ldots, x_n) = 1 - [\prod_{i=1}^n (1 - x_i)]^{\frac{1}{n}}$
GB Grouping	$GB(x_1, \ldots, x_n) = 1 - \sqrt{\prod_{i=1}^n (1 - x_i) \cdot \min\{1 - x_1, \ldots, 1 - x_n\}}$
GDIV Grouping	$GDIV(x_1, \ldots, x_n) = 1 - \frac{\prod_{i=1}^n (1-x_i) + \min\{1-x_1, \ldots, 1-x_n\}}{2}$

$$OnB(x_1, \ldots, x_n) = \sqrt{\left(\prod_{i=1}^{n} x_i\right) \cdot \min\{x_1, \ldots, x_n\}}. \tag{4}$$

Nevertheless, OnB can be defined through Theorem 3.1, showing that it is a RE aggregation function. Take C as the geometric mean GM, F_n^1 and F_n^2 as the product $PROD_n$ and the minimum MIN_n, respectively, and the family of parametric aggregation functions $H_{(\rho)} : [0,1]^2 \to [0,1]$, with $\rho \in \mathcal{N}$, given, for all $y_1, y_2 \in [0,1]$ and $x_1, \ldots, x_\rho \in [0,1]$, by

$$H_{(\rho)}(y_1, y_2) = \begin{cases} 0, \text{ if } y_1 = 0; \\ \left(\frac{y_1^2}{MIN_\rho(x_1, \ldots, x_\rho)} \cdot y_2 \cdot \min\left\{ \frac{y_1^2}{PROD_\rho(x_1, \ldots, x_\rho)}, y_2 \right\} \right)^{\frac{1}{2}}, \\ \qquad \text{otherwise.} \end{cases}$$

Thus, one can define a recursively extendable function $OB_n : [0,1]^n \to [0,1]$, given, for all $x_1, \ldots, x_{n+1} \in [0,1]$, by

$$\begin{array}{c} OB_2(x_1, x_2) = GM(PROD_2(x_1, x_2), MIN_2(x_1, x_2)) \\ OB_{n+1}(x_1, \ldots, x_{n+1}) = H_{(n)}(GM(PROD_n(x_1, \ldots, x_n), \\ \qquad MIN_n(x_1, \ldots, x_n)), x_{n+1}). \end{array}$$

Also, the value of $OB_n(x_1, \ldots, x_n)$ coincides with the value of $OnB(x_1, \ldots, x_n)$ (Eq. (4)), for all $x_1, \ldots, x_n \in [0,1]$.

A. Classes of recursively extendable aggregation functions

Observe that Def. 3.1 can be adapted for any subclass of aggregation functions by stating the function F to be of this subclass when $n = 2$. Now, we present the definitions of recursively extendable n-dimensional overlap functions.

Definition 3.2: A function $O_n : [0,1]^n \to [0,1]$, with $n \in \mathcal{N}$, is said to be an RE n-dimensional overlap function if there exist an overlap function $O : [0,1]^2 \to [0,1]$ and a family of parametric aggregation functions $H_{(\rho)} : [0,1]^2 \to [0,1]$, with $\rho \in \mathcal{N}$, such that, for all $x_1, \ldots, x_{n+1} \in [0,1]$:

$$\begin{array}{rcl} O_2(x_1,x_2) & = & O(x_1,x_2) \\ O_{n+1}(x_1,\ldots,x_{n+1}) & = & H_{(n)}(O_n(x_1,\ldots,x_n),x_{n+1}). \end{array}$$

Example 3.3: The geometric mean GM (Table II) is an overlap function that is not associative, but it is RE. To show that, let $F:[0,1]^2 \to [0,1]$ be an aggregation function, defined, for all $x_1,x_2 \in [0,1]$, by $F(x_1,x_2)=\sqrt{x_1 \cdot x_2}$ and $H_{(\rho)}:[0,1]^2 \to [0,1]$ be a family of parametric aggregation functions, with $\rho \in \mathcal{N}$, given, for all $y_1,y_2 \in [0,1]$, by $H_{(\rho)}(y_1,y_2)=(y_1^\rho \cdot y_2)^{\frac{1}{\rho+1}}$. The function $GM_n:[0,1]^n \to [0,1]$ is RE to the arity $n+1$, since, for $x_1,\ldots,x_{n+1} \in [0,1]$, we can define

$$\begin{array}{rcl} GM_2(x_1,x_2) & = & F(x_1,x_2) \\ GM_{n+1}(x_1,\ldots,x_{n+1}) & = & H_{(n)}(GM_n(x_1,\ldots,x_n),x_{n+1}), \end{array}$$

providing a definition to the n-dimensional GM in a recursive manner, i.e., GM_n is an RE n-dimensional overlap function.

To study the duality property, let us first recall the concept of *fuzzy negation* [23], which is a function $N:[0,1] \to [0,1]$ such that: **(N1)** $N(0)=1$ and $N(1)=0$; **(N2)** If $x \leq y$ then $N(y) \leq N(x)$, for all $x,y \in [0,1]$. If the involutive property, given by **(N3)** $N(N(x))=x$, for all $x \in [0,1]$, is also satisfied, then N is said to be a strong fuzzy negation.

The *Zadeh negation* given, for all $x \in [0,1]$, by $N_Z(x)=1-x$, is an example of strong fuzzy negation.

Given a strong fuzzy negation $N:[0,1] \to [0,1]$ and $H:[0,1]^n \to [0,1]$, then $H^N:[0,1]^n \to [0,1]$, defined, for all $x_1,\ldots,x_n \in [0,1]$, by $H^N(x_1,\ldots,x_n)=N(H(N(x_1),\ldots,N(x_n)))$, is the N-dual of H. Here, the N_Z-dual function (dual with respect to the Zadeh negation) of H is just called dual of F.

The dual class of n-dimensional overlap functions is that of n-dimensional grouping functions [15] $Gn:[0,1]^n \to [0,1]$, for which, for all $x_1,\ldots,x_n \in [0,1]$, the following conditions hold: **(Gn1)** Gn is symmetric; **(Gn2)** $Gn(x_1,\ldots,x_n)=0 \Leftrightarrow x_i=0$ for all $i \in \{1,\ldots,n\}$; **(Gn3)** $Gn(x_1,\ldots,x_n)=1 \Leftrightarrow$ there exists $i \in \{1,\ldots,n\}$ such that $x_i=1$; **(Gn4)** Gn is increasing; **(Gn5)** Gn is continuous. When n = 2, they are just called grouping functions. Table III shows examples of n-dimensional grouping functions obtained by duality from the overlap functions of Table II.

Definition 3.3: A function $G_n:[0,1]^n \to [0,1]$, with $n \in \mathcal{N}$, is said to be an RE n-dimensional grouping function if there exist a grouping function $G:[0,1]^2 \to [0,1]$ and a family of parametric aggregation functions $H_{(\rho)}:[0,1]^2 \to [0,1]$, with $\rho \in \mathcal{N}$, such that, for all $x_1,\ldots,x_{n+1} \in [0,1]$:

$$\begin{array}{rcl} G_2(x_1,x_2) & = & G(x_1,x_2) \\ G_{n+1}(x_1,\ldots,x_{n+1}) & = & H_{(n)}(G_n(x_1,\ldots,x_n),x_{n+1}), \end{array}$$

Proposition 3.2: Let $M_n:[0,1]^n \to [0,1]$ be an RE aggregations function, related to the function $F:[0,1]^2 \to [0,1]$ and the family of parametric aggregation functions $H_{(\rho)}:[0,1]^2 \to [0,1]$, with $\rho \in \mathcal{N}$. Then the dual of M_n, $M_n^d:[0,1]^n \to [0,1]$, is an RE aggregation function, given, for all $x_1,\ldots,x_{n+1} \in [0,1]$, by:

$$\begin{array}{rcl} M_2^d(x_1,x_2) & = & F^d(x_1,x_2) \\ M_{n+1}^d(x_1,\ldots,x_{n+1}) & = & H_{(n)}^d(M_n(x_1,\ldots,x_n),x_{n+1}), \end{array}$$

where F^d and $H_{(\rho)}^d$ are the dual of F and H_ρ, respectively.

Proposition 3.2 shows that one can define RE aggregation functions by the duality property. For instance, in the following example, we define an RE n-dimensional grouping function by means of an RE n-dimensional overlap function.

Example 3.4: The dual of the geometric mean, GM^d (Table III), is a grouping function that is not associative, but it is RE. Considering F, $H_{(\rho)}$ and GM_n (Ex. 3.3), then, following Prop. 3.2, the function $GM_n^d:[0,1]^n \to [0,1]$ is an n-dimensional grouping function that is RE to $n+1$ and it is also the dual of GM_n. In fact, for all $x_1,\ldots,x_{n+1},y_1,y_2 \in [0,1]$, we define

$$\begin{array}{c} GM_2^d(x_1,x_2)=F^d(x_1,x_2) \\ GM_{n+1}^d(x_1,\ldots,x_{n+1})=H_{(n)}^d(GM_n^d(x_1,\ldots,x_n),x_{n+1}), \\ \text{where} \quad F^d(x_1,x_2)=1-[(1-x_1)\cdot(1-x_2)]^{\frac{1}{2}}, \\ \text{and} \quad H_{(\rho)}^d(y_1,y_2)=1-((1-y_1)^\rho \cdot (1-y_2))^{\frac{1}{\rho+1}} \end{array}$$

Thus, GM_n^d is a RE n-dimensional grouping function and is the dual of GM_n, defined in Ex. 3.3.

Example 3.5: We present examples of RE n-dimensional grouping functions that are applied in experiments in Sect. V: **(a)** Consider $F{=}OB$ (Table II) and $H_{(\rho)}$ from Ex. 3.2. From Prop. 3.2, the function $OB_n^d=GB_n:[0,1]^n \to [0,1]$ is an RE n-dimensional grouping function (the dual of OB_n). In fact, for $x_1,\ldots,x_{n+1} \in [0,1]$ and $y_1,y_2,x_1,\ldots,x_\rho \in [0,1]$, we define

$$\begin{array}{l} GB_2(x_1,x_2)=F^d(x_1,x_2) \\ GB_{n+1}(x_1,\ldots,x_{n+1})=H_{(n)}^d(GB_n(x_1,\ldots,x_n),x_{n+1}), \\ \text{where } F^d(x_1,x_2)= \\ \quad 1-[(1-x_1)\cdot(1-x_2)\cdot\min\{(1-x_1),(1-x_2)\}]^{\frac{1}{2}}, \\ \text{and } H_{(\rho)}^d(y_1,y_2)= \\ \quad \begin{cases} 1 \text{ if } y_1=1; \\ 1-\left(\frac{(1-y_1)^2}{MIN_\rho(1-x_1,\ldots,1-x_\rho)}\cdot(1-y_2)\cdot \right. \\ \qquad \left. \min\left\{\frac{(1-y_1)^2}{PROD_\rho(1-x_1,\ldots,1-x_\rho)},(1-y_2)\right\}\right)^{\frac{1}{2}}, \\ \qquad\qquad \text{otherwise.} \end{cases} \end{array}$$

Thus, GB_n is the dual of OB_n (Ex. 3.2). Also, GB_n coincides with GB (Table III).

(b) Consider $F{=}ODIV$ (Table II) and let $H_{(\rho)}$ be a family of parametric aggregation functions, with $\rho \in \mathcal{N}$, given by

$$\begin{array}{l} H_{(\rho)}(y_1,y_2)= \\ \quad \begin{cases} 1, \text{ if } y_1=1; \\ 1-\frac{1}{2}\cdot\left(\frac{2\cdot(1-y_1)}{MIN_\rho(1-x_1,\ldots,1-x_\rho)}\cdot(1-y_2)\cdot \right. \\ \qquad \left. \min\left\{\frac{2\cdot(1-y_1)}{PROD_\rho(1-x_1,\ldots,1-x_\rho)},(1-y_2)\right\}\right), \\ \qquad\qquad \text{otherwise.} \end{cases} \end{array}$$

for all $y_1,y_2 \in [0,1]$ and $x_1,\ldots,x_\rho \in [0,1]$.

Then, we have that the function $GDIV_n:[0,1]^n \to [0,1]$ is a recursively extendable n-dimensional grouping function. In fact, for all $x_1,\ldots,x_{n+1} \in [0,1]$, we define

$$\begin{array}{l} GDIV_2(x_1,x_2)=F^d(x_1,x_2) \\ GDIV_{n+1}(x_1,\ldots,x_{n+1})=H_{(n)}(GDIV_n(x_1,\ldots,x_n),x_{n+1}), \\ \text{where } F^d(x_1,x_2)= \\ \quad 1-\frac{1}{2}\cdot[(1-x_1)\cdot(1-x_2)\cdot\min\{(1-x_1),(1-x_2)\}]. \end{array}$$

Thus, the value of $GDIV_n$ coincides with that of $GDIV$ (Table III), for all $x_1,\ldots,x_n \in [0,1]$.

Since the maximum and probabilistic sum are associative grouping functions, we can construct all the n-dimensional grouping functions from Table III in a recursive manner, as shown in Examples 3.4 and 3.5.

In the next section, we apply the results on RE aggregation functions to develop two different approaches for defining incremental cluster CM to be used in the cluster merging decision of the d-FuzzStream algorithm, as explained in Sect. II-A, which is, in fact, the 2nd objective posed in the Introduction.

IV. New Approaches for cluster merging decision

Following the discussion on Sect. II, the merge operation of pairs of *FMiCs* is performed in the context of DS clustering in order to make the summary structure more compact and representative. Assuming that clusters that are very similar or with high overlapping represent the same information, this operation tends to generate a representation that is more compatible with the real structure of the data.

Based on the previously defined RE aggregation functions (Sect. III), in this study, we will attack the problem by proposing two different ways to compare clusters: (1) similarity between clusters and (2) overlapping between clusters.

To maintain the compatibility of these proposed approaches with the used clustering method, we introduce the concept of comparison matrix as a generalization of the similarity matrix R adopted in the original *d-FuzzStream* algorithm. A Comparison Matrix (CMX) is a $k \times k$ matrix, where k is the number of *FMiCs* present in the current summary structure, and each cell $CMX(i,j)$, $i,j = 1,2,\ldots,k$, contains a CM between the *FMiCs* i and j.

In the original version of the *d-FuzzStream* algorithm, the similarity between *FMiCs* can be calculated using only the components stored in the vector structure that represents the *FMiC* (Eq. (1)). For the CM calculations proposed in this paper, different partial values must be stored so that the comparison value between pairs of *FMiCs* can be updated recursively. For that, the structure used to represent the comparison matrix must be more complex than in the case of the original version of the algorithm and contain different auxiliary partial values properly defined for each CM.

A. Similarity between clusters

To formally construct cluster similarity measures, first, let us recall some important concepts. For that, denote by $FS(X)$ the space of all fuzzy sets defined over a finite set X.

A set function $SM : FS(X)^2 \to [0,1]$ is called a *similarity measure* [43] on $FS(X)$ if, for all $A,B,C \in FS(X)$, it holds that: **(SM1)** $SM(A,B) = SM(B,A)$; **(SM2)** $SM(A,B) = 0$ if and only if $\{A(x),B(x)\} = \{0,1\}$, for all $x \in X$; **(SM3)** $SM(A,B) = 1$ if and only if $A = B$; **(SM4)** if $A \leq B \leq C$ then $SM(A,B) \geq SM(A,C)$ and $SM(B,C) \geq SM(A,C)$.

A function $R : [0,1]^2 \to [0,1]$ is said to be a *restricted equivalence function* (REF) [44] associated with a strong negation $N{:}[0,1]{\to}[0,1]$, if, for all $x,y,z{\in}[0,1]$, it holds that: **(1)** R is symmetric; **(2)** $R(x,y){=}1$ iff $x{=}y$; **(3)** $R(x,y){=}0$ if and only if $\{x,y\}{=}\{0,1\}$; **(4)** $R(x,y) = R(N(x),N(y))$; **(5)** if $x \leq y \leq z$ then $R(x,y) \geq R(x,z)$ and $R(y,z) \geq R(x,z)$.

A continuous, strictly increasing function $\varphi : [0,1] \to [0,1]$ with $\varphi(0) = 0$, $\varphi(1) = 1$ is called an *automorphism* of the interval $[0,1]$. Given automorphisms $\varphi_1,\varphi_2{:}[0,1]{\to}[0,1]$ and the strong fuzzy negation $N{:}[0,1]{\to}[0,1]$, defined, for $x{\in}[0,1]$, by $N(x) = \varphi_2^{-1}(1{-}\varphi_2(x))$, then the function $R : [0,1]^2 \to [0,1]$, given, for $x,y \in [0,1]$, by $R(x,y){=}\varphi_1^{-1}(1{-}|\varphi_2(x){-}\varphi_2(y)|)$, is a REF associated with N. [44]

Example 4.1: Let $\varphi_1,\varphi_2 : [0,1] \to [0,1]$ be the automorphisms given, for all $x \in [0,1]$, respectively, by $\varphi_1(x) = x^r, \varphi_2(x) = x^t$, with $r,t \in (0,+\infty)$, and $N^t{:}[0,1]{\to}[0,1]$ be the fuzzy negation, defined, for $x{\in}[0,1]$, by $N^t(x){=}(1{-}x^t)^{\frac{1}{t}}$. The function $R_{r,t}{:}[0,1]^2{\to}[0,1]$, given, for all $x,y{\in}[0,1]$, by

$$R_{r,t}(x,y) = (1 - |x^t - y^t|)^{\frac{1}{r}}, \tag{5}$$

is a REF associated with N^t.

We recall a construction method for similarity measures:

Proposition 4.1: [44] Let $R : [0,1]^2 \to [0,1]$ be a REF and $M : [0,1]^n \to [0,1]$ be an aggregation function satisfying: **(M1)** $M(x_1,\ldots,x_n) = 0$ if and only if $x_1 = \ldots = x_n = 0$, **(M2)** $M(x_1,\ldots,x_n) = 1$ if and only if $x_1 = \ldots = x_n = 1$, for all $x_1,\ldots,x_n \in [0,1]$. Then, the function $SM_R^M : FS(X)^2 \to [0,1]$, defined, for all $A,B \in FS(X)$, by $SM_R^M(A,B) = M(R(A(x_1),B(x_1)),\ldots,R(A(x_n),B(x_n)))$, is a similarity measure on $FS(X)$.

Now we introduce our approach of cluster similarity measure, based on RE aggregation functions:

Definition 4.1: Let $R : [0,1]^2 \to [0,1]$ be a REF and $M : [0,1]^n \to [0,1]$ be a RE aggregation function, satisfying the constraints required by Prop. 4.1. Then, the function SM_R^M obtained by the construction method in Prop. 4.1 is called a cluster similarity measure (CSM).

Example 4.2: Here, to analyze the similarity between *FMiCs*, we use the CSM defined by Def. 4.1, adopting as the function M the RE arithmetic mean AM_n (Ex. 3.1), and as the function R the REF $R_{r,t}$ (Ex. 4.1, Eq. (5)), obtaining the CSM $SM_{R_{r,t}}^{AM_n}$, where $r,t \in (0,+\infty)$. In such a case, to calculate the similarity recursively, as each example is processed, it is necessary to store the current values of AM_n and n.

B. Overlapping between clusters

To formally define a function that measures the overlapping between clusters, first, we revisit some important concepts.

A mapping $\mathcal{O} : FS(X) \times FS(X) \to [0,1]$ is said to be an overlap index [45] if it satisfies the following conditions, for all $A,B,C \in FS(X)$: $(\mathcal{O}1)$ $\mathcal{O}(A,B) = 0$ if and only if, for all $x \in X, A(x) \cdot B(x) = 0$; $(\mathcal{O}2)$ $\mathcal{O}(A,B) = \mathcal{O}(B,A)$; $(\mathcal{O}3)$ If $B \leq C$, meaning that $B(x) \leq C(x)$, for all $x \in X$, then $\mathcal{O}(A,B) \leq \mathcal{O}(A,C)$. An overlap index is normal if it also satisfies the condition: $(\mathcal{O}4)$ If there exists $x{\in}X$ such that $A(x){\cdot}B(x){=}1$, then $\mathcal{O}(A,B){=}1$. As an example, the function $\mathcal{O}_Z{:}FS(U){\times}FS(U) \to [0,1]$ defined, for $A,B{\in}FS(U)$, by:

$$\mathcal{O}_Z(A,B) = \max_{x \in X} \min\{A(x),B(x)\}, \tag{6}$$

is a normal overlap index (Zadeh's consistency index).

We construct overlap indices by aggregating overlap functions (see Table II, Ex. 3.2).

Now, we recall a construction method for overlap indices:

Theorem 4.1: [45] Consider an aggregation function $M : [0,1]^n \to [0,1]$ with the property **(M1)** (see Prop. 4.1) and an overlap function $O : [0,1]^2 \to [0,1]$. Then, the function $\mathcal{O}_M^O : FS(X) \times FS(X) \to [0,1]$, given, for all $A,B \in FS(X)$, by $\mathcal{O}_M^O(A,B) = M(O(A(x_1),B(x_1)),\ldots,O(A(x_n),B(x_n)))$, is an overlap index.

The property **(M1)** coincides with the property **(Gn2)** of n-grouping functions. It follows that:

Corollary 4.1: Let $Gn : [0,1]^n \to [0,1]$ be an n-dimensional grouping function and $O : [0,1]^2 \to [0,1]$ be an overlap function. The function $\mathcal{O}_{Gn}^O : FS(X){\times}FS(X) \to$

TABLE IV: The adopted similarity measures/overlap indices

Indentifier	AF	REF/O	Indentifier	AF	REF/O
Comp1	Fuzzy Disp.	Original	Comp17	GM^d	$ODIV$
Comp2	AM	Sim. Measure	Comp18	GB	$PROD$
Comp3	PS	$PROD$	Comp19	GB	MIN
Comp4	PS	MIN	Comp20	GB	GM
Comp5	PS	GM	Comp21	GB	OB
Comp6	PS	OB	Comp22	GB	$ODIV$
Comp7	PS	$ODIV$	Comp23	$GDIV$	$PROD$
Comp8	MAX	$PROD$	Comp24	$GDIV$	MIN
Comp9	MAX	MIN	Comp25	$GDIV$	GM
Comp10	MAX	GM	Comp26	$GDIV$	OB
Comp11	MAX	OB	Comp27	$GDIV$	$ODIV$
Comp12	MAX	$ODIV$	Comp28	AM	$PROD$
Comp13	GM^d	$PROD$	Comp29	AM	MIN
Comp14	GM^d	MIN	Comp30	AM	GM
Comp15	GM^d	GM	Comp31	AM	OB
Comp16	GM^d	OB	Comp32	AM	$ODIV$

$[0,1]$, given, for all $A, B \in FS(X)$, by $\mathcal{O}^O_{Gn}(A,B) = Gn(O(A(x_1),B(x_1)),\ldots,O(A(x_n),B(x_n)))$, is an overlap index.

By Cor. 4.1, we combine the functions from Tables II and III to construct different overlap indices. If $Gn = MAX$ and $O = MIN$, then we obtain the Zadeh's index (Eq. (6)).

In the following, we introduce our approach for measuring the overlapping between clusters through overlap indexes based on RE aggregation functions:

Definition 4.2: Let $O : [0,1]^2 \rightarrow [0,1]$ be an overlap function and $M : [0,1]^n \rightarrow [0,1]$ be a RE aggregation function, satisfying the constraints required by Theorem 4.1. Then, the function $\mathcal{O}^O_M$ obtained by the construction method in Theorem 4.1 is called a cluster overlap index (COI).

By Cor. 4.1, any n-dimensional grouping that is RE may act as the function M in Def. 4.2, with the advantage that, by Prop. 3.2, they can be obtained by duality from RE n-dimensional overlap functions.

Example 4.3: In this paper, we analyze the overlapping between *FMiCs* using the COI defined by Def. 4.2, adopting as the function M, RE n-dimensional grouping functions G_n (Def. 3.3), as the ones discussed in Ex. 3.5, and the overlap functions of Table II, obtaining the COI $\mathcal{O}^O_{G_n}$. In particular, consider the RE n-dimensional grouping function GB_n (Ex. 3.5 (a)) and the overlap function OB (Table II). Then, in this case, to calculate the overlap index recursively, as each example is processed, besides storing the current value of GB_n and n, it is also necessary to keep the current values of $MIN_n(1-x_1,\ldots,1-x_n)$ and $PROD_n(1-x_1,\ldots,1-x_n)$.

V. Experimental Framework

Our last objective posed in the Introduction is to study, by means of experiments, the behaviour of a set of CMs of the types proposed in the previous section, within the scope of the *d-FuzzStream* clustering algorithm.

Both phases - online and offline – have been evaluated. We have studied and evaluated in detail the online phase since our focus is to get insights into the behaviour of the clustering method during the online process, as we change the CM between clusters. The evaluation of the offline phase includes comparisons with similar approaches. In the following, we establish our experimental framework:

1) Datasets: A summary of each dataset employed in this paper is shown in Table V, showing its synthetic or real origin, with the number of instances, classes and attributes, besides a column **Noise?** indicating whether the dataset contains noisy data and a column **EM** showing the number of instances in each Evaluation Moment (see the Item 2 below). For the analysis of the online phase, two synthetic datasets were considered: Bench1_11k and RBF1_40k. For the offline phase, in addition to the two previous datasets, we also tested the synthetic Gaussian 4C2D800Linear dataset and the real-world datasets Powersupply and NOAA Wheater dataset.

TABLE V: Dataset Description

Dataset	Synthetic/Real	Instances	Attributes	Classes	Noise?	EM
Bench1_11k [46]	Synthetic	11.000	2	2	Y	1.000
RBF1_40k [46]	Synthetic	40.000	2	4	Y	1.000
4C2D800Linear [47]	Synthetic	800	2	2	Y	100
PowerSupply [48]	Real	29.928	2	24	N	1.000
NOAA [49]	Real	18,159	8	2	N	1.000

2) Evaluation Moments: In both phases, online and offline, the validation indexes were applied at specific points during the stream, which we call Evaluation Moments (EM), more precisely after processing a certain number of instances. These moments were defined specifically for each dataset and depend on their size, as can be seen in column **EM** of Table V. To facilitate the clarity and understanding of the analyses, we used, in each of the three groups of experiments with the online phase and the experiments with the offline phase, only some selected EMs and indexes, which are those that support the highlights that are worth pointing out.

3) Validation: For the analyses of the online phase, we used incremental versions of well-known fuzzy clustering validation indexes: Xie Beni (XB) [50] and Partition Coefficient (PC) [34], defined, respectively, as: $XB = \frac{\frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^m\|c_i-x_j\|^2}{min_{i\neq j}\|c_i-c_j\|^2}$ and $PC = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^m$, where c is the number of clusters, n is the number of instances, $\mu_{i,j}$ is the membership degree of instance i in cluster j and c_j is the center of j.

The XB index evaluates the quality of fuzzy partitions considering the concepts of intra-cluster cohesion and inter-cluster separation, ranging in the interval $[0,1]$, and the better partitions are the ones with lower values. The incremental version used here is similar to the one presented in [51]. PC is one of the most traditional fuzzy clustering evaluation indexes, having values in $[1/c,1]$ and the better partitions are the ones with higher values. In the analyses of the offline phase, we used the external validation measure Adjusted Rand Index (ARI) [52] as the average of the results of each EM.

4) Comparison Measures: Experiments with the online phase were run using 32 different CM, which are listed in Table IV. The first one (Comp1) is the dispersion-based similarity proposed in [41] adopted in the original version of *d-FuzzStream*, described in Sect. II-A. It has been included in the experiments for comparison purposes. The second CM (Comp2) is a cluster similarity measure (Sect. IV-A, Def. 4.1) and the other 30 CMs are cluster overlap indices (Sect. IV-B, Def. 4.2). In Table IV, the columns **Identifier** contain an identifier for the CM, columns **AF** and **REF/**O contain the RE aggregation function and REF/overlap function, respectively, used in the definition of the CM.

5) Merging Threshold: The algorithm merging threshold is the value defined to decide, in the online phase, whether two *FMiCs* should be merged. Two *FMiCs* are merged when the value of the applied CM is above the merging threshold. Since different CMs give different results for the same pair of

FMiCs, the online clustering can be sensitive to the definition of such parameters. Thus, the threshold value cannot be a fixed value for all CM. The choice of the appropriate threshold for each CM was made empirically, as a result of the search for a balance between some criteria, such as:

a) Improvement of the clustering validation index: the number of *FMiCs* tends to increase and the number of merges to decrease with the increase of the threshold value. The threshold might be a good choice when it leads to better results than the previous values;

b) Number of merges: if the number of merges is too close to the number of creations of new *FMiCs*, then a merging is done almost every time a new example arrives, which is not compatible with the idea of allowing merges. On the other hand, too few or no merges do not allow the analysis of the merging impact. Values that allow fewer merges than creations of *FMiCs* are preferable;

c) Number of removals: a low number of removals (or even no removals) means that the model takes too much time to "forget" past information and the online clustering might not be able to capture the real structure of the data as new data arrive and the distribution changes. On the other hand, a high number of removals means that *FMiCs* include too few data points. Thresholds that lead to a number of removals of *FMiCs* that is about half of the number of creations are preferable.

6) Algorithms used in the comparison analysis: The comparison analysis for the offline phase was done against classic online clustering algorithms CluStream [38], DBStream [53], DenStream [54], Stream K-Means [55] and Incremental K-Means [56]. The implementation used was obtained from the river library [1]. Besides these traditional algorithms, we also included five recent stream clustering algorithms in the comparison analysis, namely TSSRC [3], EvolveCluster [54], OSRC [55], ACSC[56], and TFS-DBSCAN [32], all mentioned in Section II.

For the fairness of the comparison, we tested a reasonable range of the parameters of each algorithm selecting those that provided the highest average ARI value. In the case of the fuzzy algorithms, the resulting fuzzy clusters were converted to hard clusters based on the highest membership degree. For the algorithms with a k-means base (Incremental K-Means, Stream K-Means, and CluStream), we tuned the clusters' initialization parameters and the inertia towards the new observation. For the CluStream algorithm, we also studied the effect of the maximum number of microclusters. For the DBStream algorithm, we vary the clustering parameters (r, α, and w_{min}), as well as the fading factor λ which controls the importance of historical data. For the DenStream algorithm, we tuned the decay factor λ which controls the importance of historical data and the microclusters maintenance parameters (ϵ, β, and μ). Additionally, we studied the number of samples to initialize the online process. For the TSF-DBSCAN algorithm, we vary the distance thresholds (σ_{min}) and (σ_{max}) and the weight threshold (θ_w) which determine the belonging of a point to a cluster. For the TSSRC algorithm, we tuned the parameters μ and ρ.

[1]https://riverml.xyz

TABLE VI: XB values and Indicators

(1) Bench1_11k						
Comp04	**EMs**	0.25	0.5	0.65	0.8	0.9
	1	0,001440	**0,000002**	**0,000002**	0,000003	0,000224
	2	0,002392	0,000603	0,000549	**0,002223**	0,000528
	3	0,001199	**0,000002**	**0,000002**	**0,000002**	**0,000002**
	4	0,000224	**0,000202**	**0,000202**	**0,000202**	**0,000202**
	5	0,000021	**0,000001**	**0,000001**	**0,000001**	**0,000001**
	6	0,000041	**0,000005**	**0,000005**	**0,000005**	**0,000005**
	7	0,000033	**0,000001**	**0,000001**	**0,000001**	**0,000001**
	8	0,000083	**0,000004**	**0,000004**	**0,000004**	**0,000004**
	9	0,001830	**0,000016**	**0,000016**	**0,000016**	0,000017
	10	**0,000016**	0,000602	0,000478	0,000422	0,000422
	11	0,000396	**0,000005**	0,000012	0,001469	0,000736
	Mean	0,000698	0,000131	**0,000116**	0,000395	0,000195
	Indicators					
	#Creations	8122	8058	7830	7515	7375
	#Absorptions	2878	2942	3170	3485	3625
	#Removals	2727	6274	6422	6293	6837
	#Merges	5346	1734	1358	1172	488
(2) RBF1_40k						
Comp27	**EMs**	0.25	0.5	0.65	0.8	0.9
	1	0,021056	0,021056	0,001557	**0,001517**	0,001520
	2	0,029803	0,029803	**0,000023**	0,000051	0,000044
	3	0,056950	0,056950	**0,000027**	0,000070	0,000049
	4	0,100723	0,100723	0,000102	**0,000095**	**0,000095**
	5	0,124408	0,124408	**0,000023**	0,000026	0,000027
	6	0,112338	0,112338	0,000485	0,000075	**0,000059**
	7	0,230202	0,230202	0,000517	**0,000072**	0,000434
	8	0,331319	0,331319	**0,000031**	0,000187	0,000124
	9	0,204900	0,204900	0,000065	**0,000026**	0,000030
	10	0,223155	0,223155	**0,000060**	0,000121	**0,000060**
	Mean	0,143485	0,143485	0,000289	**0,000224**	0,000244
	Indicators					
	#Creations	33090	33090	32791	32219	32047
	#Absorptions	6910	6910	7209	7781	7953
	#Removals	0	0	23773	31352	31907
	#Merges	33086	33086	8918	767	40

7) Methodology: For each dataset, the analysis of experiments was done separately for the online and offline phases with different purposes. In the online phase the experiments were divided into 3 parts to analyse the impact of the CM on the quality of the micro-clusters: (1) selection of the best threshold for each CM; (2) overall results for all CM; (3) analysis of the most representative behavioural patterns and CM with the best results. In the offline phase, the objective is to evaluate the quality of the offline clustering results and compare them with other traditional and state-of-the-art stream clustering algorithms found in the literature.

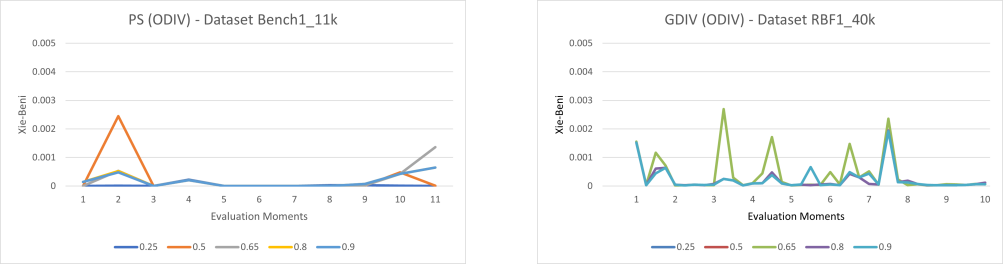
VI. Results and Analysis of the Online Phase

In the online phase, the analyses are discussed according to the three parts mentioned in the previous section for datasets Bench1_11k and RBF1_40k. We remark that, although the values of the validation indexes obtained in the online phase are generally low, the focus of the analysis is the relative value between different merging thresholds and different CM.

Part 1 - Selection of the best threshold for each CM

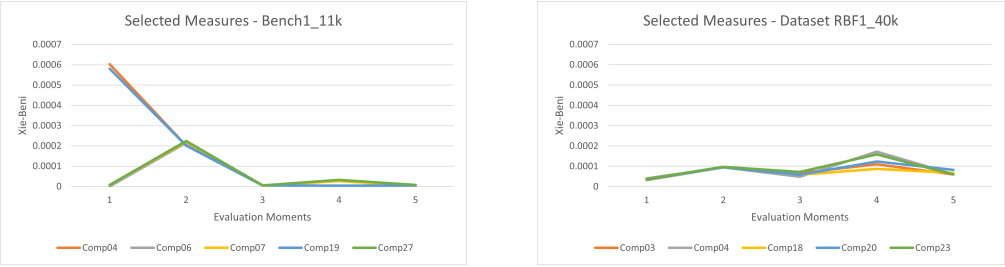
Experiments were run setting different merge threshold values for each CM, with the objective of evaluating the impact of the threshold on the quality of the summary structure. Using the criteria described in the previous section, the most suitable threshold was selected for each of the CMs. The clustering evaluation index used in this part was the XB index.

We present detailed results for one specific CM for each one of the datasets to illustrate this selection process. For dataset Bench1_11k, the chosen CM is the overlap index formed by $PS(Min)$ (Comp04). For dataset RBF1_40k, the chosen CM is the overlap index formed by $GDIV(ODIV)$ (Comp27).



(a) Comp07 - $PS(ODIV)$ (b) Comp27 - $GDIV(ODIV)$

Fig. 2: Xie-Beni values for Bench1_11k (left) and RBF1_40k (right) datasets, except for thresholds 0.25 and 0.5



(a) Best CMs (b) Best CMs

Fig. 3: Xie-Beni values of the best CMs for Bench1_11k (left) and RBF1_40k (right) datasets

In Table VI(1) one can see the XB values for thresholds 0.25, 0.5, 0.65, 0.8, 0.9, and for each EM set at every 1,000 examples of dataset Bench1_11k. The best values for each EM are in bold. We also show the indicators of the number of *FMiCs* that were created/absorbed/removed from the structure and the number of merges that occurred, which are part of the criteria to select the most suitable merge threshold.

The XB values for all thresholds in Table VI(1) are shown in Fig. 2a. Observe that for most thresholds, the index values become higher at EMs 2 and 10, meaning that, in these specific EMs, the clusters become fuzzier. For the threshold 0.25, the results are clearly worse than those of the others. On average, threshold 0.65 presents the best result and 0.5 is the second best. Considering only the index values, 0.65 would be the best choice, but since with threshold 0.5, the number of merges is higher and the number of removals is lower than for 0.65, we decided to select 0.5 as the best threshold for Comp04.

Next, we present the CMs chosen to illustrate the process of selecting the best threshold value for dataset BRF1_40k. Since this dataset has 40,000 examples, the EMs were set at every 4,000 examples, forming 10 EMs. The XB values and indicators for this case are in Table VI(2).

The XB values for all thresholds in Table VI(2) are in Fig. 2b. The best values are for thresholds 0.8 (1srt), 0.9 (2nd), and 0.65 (3rd). Although 0.65 is in the 3rd position considering the validation index, it allows a higher number of merges and a lower number of removals. Thus, we selected 0.65 as the most appropriate threshold for Comp27 in dataset RBF1_40k.

Part 2 - Overall results for all CM

The results presented here aim to provide a global view of the experiments. For each CM and its respective threshold value selected in Part 1, the values of indexes XB and PC are shown in Table VII. For simplicity and clarity, only the mean values of 5 EMs were included, namely the ones at the processing of 2,000, 4,000, 6,000, 8,000, and 11,000 examples for dataset Bench1_11k and of every 8,000 examples for dataset RBF1_40k. The table contains 4 columns for each dataset with the XB and PC validation indexes (**XB** and **PC**) and the rankings of the CMs according to XB and PC indices (**Ranking_XB**, **Ranking_PC**, respectively).

The CMs in Table VII with the best results vary substantially depending on the dataset and the chosen validation index. The rankings for both indexes are different and one explanation for this is that they are based on different components. While XB considers both the membership degrees of the examples in each cluster and the distance between cluster centres, PC is based only on the membership degrees.

The CM of the overlap index type using the maximum (Comp08 to 12), the Geometric Mean (Comp13 to 17) and the Arithmetic Mean (Comp28 to 32), despite appearing among the first positions in the two rankings for both datasets, did not present results of greatest interest for this study, since the merges either did not occur or occurred in a very low number. In these cases, the model is ”forgetting” older data very fast, leading to well-separated clusters that contain only the most recent data, which explains the good index values. However, discarding older data too quickly is not what is expected from DS learning, where the new information should replace the older one gradually, instead of suddenly.

Despite providing different rankings with conflicting cases in some positions, confronting these two indexes, it is possible to extract useful conclusions that help to understand the obtained structure: for both indexes and datasets, most CMs presented better results than the one used in the original version of the algorithm; and considering the top 10 positions (disregarding the 15 CMs that did not allow merges mentioned above), 7 CMs appear in both rankings for Bench1_11k

TABLE VII: Overall results for all CMs in Bench1_11k (left) and RBF1_40k (right) datasets

Bench_11k					RBF1_40k				
Comp	**XB**	**Ranking_XB**	**PC**	**Ranking_PC**	**Comp**	**XB**	**Ranking_XB**	**PC**	**Ranking_PC**
Comp01	0.000376	25	0.003938	27		0,000855	31	0,006417	30
Comp02	0.004659	32	0.001711	32		0,067770	32	0,001243	32
Comp03	0.000253	22	0.004547	26		0,000232	4	0,009072	19
Comp04	0.000131	4	0.005633	20		0,000227	2	0,008977	20
Comp05	0.000344	24	0.002820	29		0,000433	30	0,006553	28
Comp06	0.000031	2	0.004880	22		0,000288	23	0,007674	23
Comp07	0.000029	1	0.004667	24		0,000399	28	0,006618	27
Comp08	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp09	0.000189	19	0.009175	14		0,000238	6	0,008930	21
Comp10	0.000404	26	0.005455	21		0,000307	25	0,006512	29
Comp11	0.000181	17	0.009150	15		0,000242	19	0,010083	15
Comp12	0.000181	18	0.009212	13		0,000242	18	0,009985	17
Comp13	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp14	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp15	0.000179	6	0.009503	1		0,000240	7	0,010243	2
Comp16	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp17	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp18	0.000208	20	0.009451	12		0,000234	5	0,009841	18
Comp19	0.000179	5	0.006909	17		0,000273	22	0,010676	1
Comp20	0.000252	21	0.007827	16		0,000228	3	0,010233	13
Comp21	0.000743	30	0.004695	23		0,000244	20	0,010124	14
Comp22	0.000586	28	0.005649	19		0,000249	21	0,010047	16
Comp23	0.000415	27	0.006289	18		0,000222	1	0,008642	22
Comp24	0.000847	31	0.001845	31		0,000348	26	0,007549	24
Comp25	0.000605	29	0.002113	30		0,000368	27	0,006870	26
Comp26	0.000312	23	0.004578	25		0,000292	24	0,007204	25
Comp27	0.000083	3	0.003377	28		0,000414	29	0,006334	31
Comp28	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp29	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp30	0.000179	6	0.009503	1		0,000240	8	0,010243	3
Comp31	0.000179	8	0.009503	3		0,000240	8	0,010243	3
Comp32	0.000179	8	0.009503	3		0,000240	8	0,010243	3

TABLE VIII: Best CMs according to Xie-Beni values for Bench1_11k (left) and RBF1_40k (right) datasets

Bench1_11K						**RBF1_40k**					
EM	Comp04	Comp06	Comp07	Comp19	Comp27	EM	Comp03	Comp04	Comp18	Comp20	Comp23
2	0.000603	**0.000007**	**0.000007**	0.000581	0.000008	8	0.000037	**0.00004**	0.000032	0.000034	0.000038
4	**0.000202**	0.000215	0.000221	**0.000202**	0.000224	16	0.000096	**0.000095**	**0.000095**	**0.000095**	0.000097
6	**0.000005**	**0.000005**	**0.000005**	**0.000005**	**0.000005**	24	0.000067	**0.000049**	0.000059	0.000059	0.000072
8	**0.000004**	**0.000004**	0.000026	**0.000004**	0.000032	32	**0.00011**	0.000172	0.000087	0.000124	0.000159
11	0.000005	**0.000004**	0.000005	0.000005	0.000007	40	**0.00006**	0.000061	0.000067	0.000082	0.000060
Mean	0.000131	0.000031	**0.000029**	0.000179	0.000083	Mean	0.000232	**0.000227**	0.000234	0.000228	0.000222
Creations	8058	7995	8198	7653	8169	Creations	32144	32497	31971	32126	32017
Absorptions	2942	3005	2802	3347	2831	Absorptions	7856	7503	8029	7874	7794
Removals	6274	6129	5518	4829	5107	Removals	31439	27398	31116	30294	31361
Merges	1734	1816	2630	2774	3012	Merges	605	4999	755	1732	745

(Comp04, 06, 07, 18, 19, 20, 26) and 9 CMs for dataset RBF1_40k (Comp03, 04, 06, 18, 19, 20, 21, 22, 23).

Part 3 - Analysis of the most representative behavioural patterns and CM with the best results

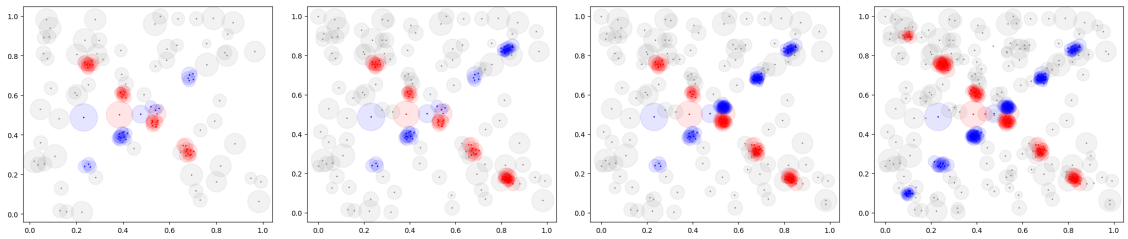
The results obtained allow the identification of three patterns of behaviour of the CM. The first pattern is observed with overlap indexes using the Maximum (Comp08 to 12), the Geometric Mean (Comp13 to 17) and the Arithmetic Mean (Comp28 to 32) that, as already pointed out in Part 2, generate very low comparison values, which lead to a very low number of merges as well. The second pattern is presented by the similarity measures based on Arithmetic Mean (Comp02) that, as opposed to the previous case, generate high comparison values, making the number of merges also high while the number of removals is low. The third pattern of behaviour is presented by the remaining 3 groups of CMs, formed by Probabilistic Sum (Comp03 to 07), GB (Comp18 to 22) and GDIV (Comp23 to 27), that share a similar behaviour, with the decrease of the number of merges and increase of the number of removals as the threshold increases. CMs with such pattern lead to different balances among merges/creations/removals with the definition of different thresholds.

Table VIII(left)/Fig. 3a and Table VIII(right)/Fig. 3b show the CMs in the 5 top positions (excluding Comp08 to Comp17, Comp28 to Comp32) in the XB rankings of Tables VII(left) and VII(right), respectively. The best XB values for each one of the 5 EM and for the mean are in bold in both tables.

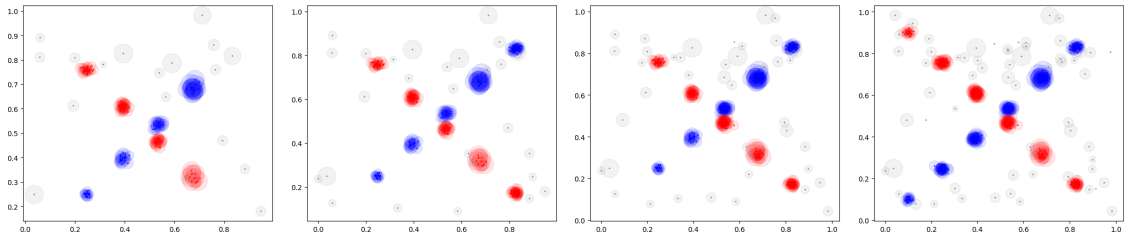
We observe that these best CMs are not necessarily the ones that should be chosen for any data stream. We emphasize that the selection of a specific CM must take into account, besides the values of the index, the balance between the numbers of merges/removals. In cases where the CM leads to a much greater number of removals than merges, the older examples are discarded faster. On the other hand, when the number of merges is higher, the older examples are kept in the structure for a longer period of time.

To complement the numeric analysis of the CMs that presented the best results according to XB, we use a visual analysis, confronting the XB values with the plot of the structure obtained by the algorithm for some selected CMs. Consider the CMs Comp24 and Comp07 for dataset Bench1_11k, which are in the 31st and 1st position, respectively, in the ranking of XB in Table VII(left). Figs. 4a, 4b, 4c and 4d show the plots of the *FMiCs* during the execution of the algorithm using Comp24 and Bench1_11k for 4 EMs, namely after 4,000, 6,000, 8,000 and 11,000 examples, respectively. Figs. 4e, 4f, 4g and 4h show the plots of the *FMiCs* during the execution using Comp07 and Bench1_11k for the same EMs.

In these figures, the circles represent the fuzzy ratio of the *FMiCs*, and the point of the same colour at the centre is the prototype. The colours (red/blue) identify the labels of the examples and grey represents the outliers, which are noise data that do not belong to any class. This set of figures illustrates that using Comp24, the *FMiCs* are more spread out and overlapped than using Comp07, where the *FMiCs* are better separated and with less overlapping. Besides that, with Comp24, several outliers (points in grey) were absorbed by the *FMiCs* representing the two classes (points in red and blue),

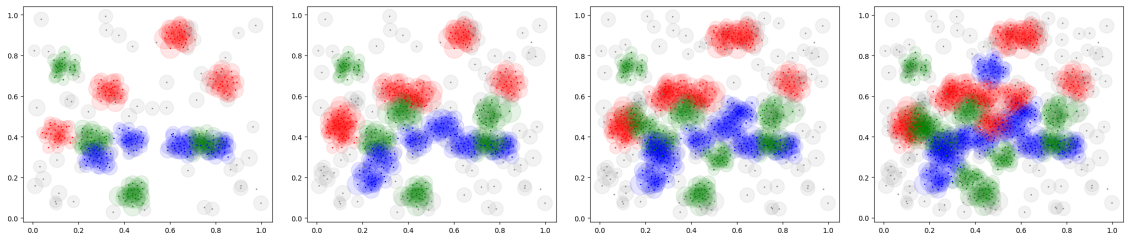


(a) EM4 - 4,000 data points (b) EM6 - 6,000 data points (c) EM8 - 8,000 data points (d) EM11 - 11,000 data pts

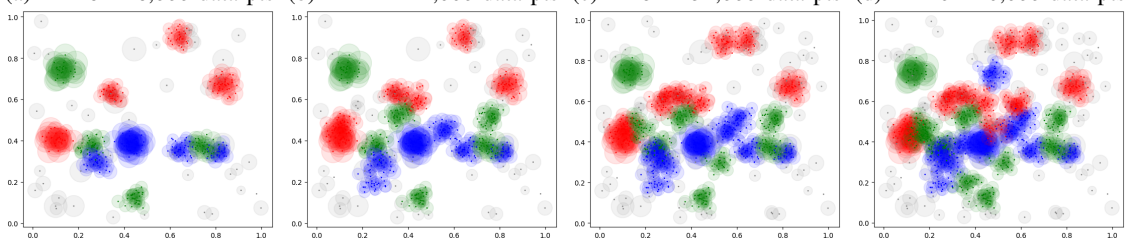


(e) EM4 - 4,000 data points (f) EM6 - 6,000 data points (g) EM8 - 8,000 data pts (h) EM11 - 11,000 data pts

Fig. 4: FMiCs plots of Dataset Bench1_11k using Comp24(top) and Comp07(bottom)



(a) EM16 - 16,000 data pts (b) EM24 - 24,000 data pts (c) EM32 - 32,000 data pts (d) EM40 - 40,000 data pts



(e) EM16 - 16,000 data pts (f) EM24 - 24,000 data pts (g) EM32 - 32,000 data pts (h) EM40 - 40,000 data pts

Fig. 5: FMiCs plots of Dataset RBF1_40k using Comp01 (top) and Comp23 (bottom)

while with Comp07 they remain separated. The visual analysis corroborates the results obtained with the XB index, for which Comp07 has better values than Comp24.

Regarding dataset RBF1_40k, consider the CMs Comp23 and Comp01, which are in the 1st and 31st position, respectively, in the XB ranking of Table VII(right). Fig. 5(top) shows the plots of the *FMiCs* during the execution of the algorithm using Comp01 and RBF1_40k at 4 EMs: EM16, after 16,000 points (Fig. 5a), EM24, after 24,000 points (Fig. 5b), EM32, after 32,000 points (Fig. 5c), EM40, after 40,000 points (Fig. 5d). For the same EMs and dataset, Figures 5e, 5f, 5g and 5h show the plots of the *FMiCs* during the execution of the algorithm using Comp23. In these figures, the circles have the same meaning as in the previous ones, except that now there are 3 classes, coloured in red, blue, and green.

The visual analysis with respect to outliers and *FMiCs* shows that Comp01 tends to keep the *FMiCs* corresponding to noise longer and makes the clusters that represent real classes more blurred. However, using Comp23, it can be seen that there is a smaller number of *FMiCs* representing outliers and that those corresponding to real classes are more defined, i.e. they have higher membership degrees.

VII. Results and Analysis of the Offline Phase

The results obtained in the offline phase, in which we compare our proposal with other benchmark algorithms, are in Table IX. For each algorithm and dataset, we show the ARI index values obtained with the best combination of parameters found in preliminary experiments. The row identified by *d-FuzzStream* shows the results of the original version of the algorithm – the one that uses the density-based CM Comp01. The last row, identified by *RE-FuzzStream* shows the index values for the *FuzzStream* with the CMs that obtain the best result for each dataset. For *d-FuzzStream* and *RE-FuzzStream*, the values shown are the mean of the ARI values for each of the EMs, which is different for each dataset (see Table V).

The last two rows of the table show that *RE-FuzzStream* outperforms, in all cases, for at least one of the CMs, the

TABLE IX: Offline Clustering Quality. Average ARI Index for all the Evaluation Moments

	Benchmark1_11000	RBF1_40k	4C2D800Linear	PowerSupply	NOAA
CluStream	0.2362	0.1174	0.4241	0.0220	0.0281
DBStream	0.7538	0.7997	0.9037	**0.1160**	0.0804
DenStream	0.7673	0.7136	0.9605	0.1127	0.0260
Stream K-Means	0.0073	0.2333	0.9638	0.0383	0.0007
Inc K-Means	0.6898	0.5916	0.9666	0.0594	0.0000
TSSRC	0.4950	0.5325	0.4253	0.0075	0.1366
TSF_DBSCAN	**0.8549**	0.7629	0.9461	0.0881	0.0261
EvolveCluster	0.0902	0.0212	0.7079	0.0157	0.0549
ACSC	0.7148	0.7275	0.9669	0,1127	0.0194
OSRC	0.5948	0.5045	0.4384	0.0068	**0.1624**
d-FuzzStream	0.7099	0.7988	0.9510	0.0659	0.0203
RE-FuzzStream	Comp22 0.7163	Comp21 **0.8132**	Comp19 **0.9703**	Comp24 0.0783	Compo02 0.0273

original *d-FuzzStream* algorithm. In terms of performance with respect to the other algorithms, *RE-FuzzStream* performs very well on datasets with noise, being the best in the cases of RBF1_40k and Gaussian2C4D800Linear, and the 4th in the case of Bench1_11k. The performance on real datasets is somewhat lower, ranking 5th for Powersupply and 6th for NOAA. Only DBStream performs better in both cases, being the 1st and the 3rd, respectively. DenStream, TSF-DBSCAN and ACSC outperform *RE-FuzzStream* in the Powersupply dataset, but their results are worse for the NOAA case, the former moving from 2nd to 8th position, the second from 4th to 7th, while the latter from 2th to 10th. CluStream, TSSRC and OSRC perform better than *RE-FuzzStream* for the NOAA dataset, ranking 4rd, 2nd and 1st, respectively, but worse for Powersupply, ranking 9th, 11th and 12th, respectively.

It is important to note that the results obtained in the experiments do not allow us to evaluate an eventual relationship between the CMs with best results in the online phase and those with best results in the offline phase since the validation indices used in each phase are different. We can state, however, that the CMs excluded from the rankings in the online phase, because they do not allow merges, are not those that produce the best result in the offline phase for each dataset, corroborating our assumption that the merge is beneficial for the final result. Nevertheless, the experimentation showed that our proposal provides balanced results in the different datasets, showing better performance in the cases with noise compared to the other alternatives.

VIII. Conclusion

In this paper, we have introduced the concept of RE aggregation functions, which are operators that may not be associative and can incrementally aggregate information. Then, we have applied them to incrementally calculate the similarity or overlap between clusters to decide on when to merge pairs of clusters in the context of stream clustering. Such CM can take the form of either similarity measures or overlap indices. We have used the *d-FuzzStream* clustering algorithm to evaluate the effects of the proposed CMs analysing both its online and offline phases. A set of 32 CMs have been evaluated in the experiments.

Regarding the online phase, different values of thresholds have been tested to define which one works better for each CM. The CMs have been ranked according to two clustering validation indexes and the most important characteristics of the ones with better results were analyzed. The results for this phase have shown that most CMs present better behaviour than the one used in the original version of the algorithm. The experiments in the offline phase have shown that the revised *d-FuzzStream* algorithm, in which we apply some of the proposed CMs, achieves better or similar performance when compared to other traditional and state-of-the-art algorithms, especially in noisy datasets.

The main contribution of this paper, in addition to the experimental results and their analyses, is to offer a solid theoretical framework, by means of RE aggregation functions, to support information fusion processes in which the data cannot be stored in the system. This can aid the aggregation process in different types of applications, such as incremental-concurrent fusion checking in context consistency [57], incremental data fusion for smart societies [58] or extracted from several external sources for online data integration [59]. In the context of DS clustering, they allow the choice of different constructed similarity measures or overlap indices, among a wide variety of options with different characteristics. Such choices can meet different user goals, which might consider the suitability of the CMs to specific data sets and the speed of response to changes in the data stream.

These benefits of the proposed algorithm, however, require higher memory usage and computing power. This is due to the need to store the partial values for the recursion and its computation each time it evaluates whether a merge is necessary or not. Parameter selection is also more complex, as it adds to the requirements of the basic algorithm the need to select and tune the comparison function to be used.

We let for future work the study of properties of different groups of similarity measures and/or overlap indices, with the application to datasets with specific characteristics, such as incomplete data and real datasets with higher dimensions.

References

[1] J. Gama, *Knowledge Discovery from Data Streams*. Chap.&Hall, 2010.

[2] S. U. Din, J. Shao, J. Kumar, C. B. Mawuli, S. M. H. Mahmud, W. Zhang, and Q. Yang, "Data stream classification with novel class detection: a review, comparison and challenges," *Knowl. Inf. Syst.*, vol. 63, pp. 2231–2276, 2021.

[3] J. Chen, Z. Wang, S. Yang, and H. Mao, "Two-stage sparse representation clustering for dynamic data streams," *IEEE Trans. Cybern.*, vol. 53, no. 10, pp. 6408–6420, 2023.

[4] J. Sui, Z. Liu, L. Liu, A. Jung, and X. Li, "Dynamic sparse subspace clustering for evolving high-dimensional data streams," *IEEE Trans. Cybern.*, vol. 52, no. 6, pp. 4173–4186, 2022.

[5] Y. Zhang, X. Li, L. Wang, S. Fan, L. Zhu, and S. Jiang, "An auto-correlation incremental fuzzy clustering framework based on dynamic conditional scoring model," *Inf. Sci.*, vol. 648, p. 119567, 2023.

[6] R. Vaarandi and A. Guerra-Manzanares, "Stream clustering guided supervised learning for classifying nids alerts," *Future Gener. Comput. Syst.*, vol. 155, pp. 231–244, 2024.

[7] T. Anwar, S. Nepal, C. Paris, J. Yang, J. Wu, and Q. Z. Sheng, "Tracking the evolution of clusters in social media streams," *IEEE Trans. Big Data*, vol. 9, no. 2, pp. 701–715, 2023.

[8] W. Zhu, R. Xiao, R. Huang, P. Gong, S. Zhang, and X. Du, "Efficient gaussian kernel microcluster real-time clustering method for industrial internet of things (iiot) streams," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21 323–21 337, 2022.

[9] A. Abdullatif, F. Masulli, and S. Rovetta, "Clustering of nonstationary data streams: A survey of fuzzy partitional methods," *WIREs Data Min. Knowl. Discov.*, no. 8, pp. 1–18, 2018.

[10] L. A. Zadeh, "Fuzzy sets," *Inf. Cont.*, vol. 8, no. 3, pp. 338–353, 1965.

[11] L. Schick, P. Lopes, and H. Camargo, "*d-FuzzStream*: A dispersion-based fuzzy data stream clustering," in *IEEE Int. Conf. Fuzzy Syst.*, RJ, 2018, pp. 135–142.

[12] C. Alsina, M. J. Frank, and B. Schweizer, *Associative Functions: Triangular Norms and Copulas*. Singapore: WSP, 2006.

[13] H. Bustince, J. Fernandez, R. Mesiar, J. Montero, and R. Orduna, "Overlap functions," *Nonlinear Anal. Theory Methods Appl.*, vol. 72, no. 3-4, pp. 1488–1499, 2010.

[14] T. C. Asmus, J. A. Sanz, G. P. Dimuro, B. Bedregal, J. Fernandez, and H. Bustince, "N-dimensional admissibly ordered interval-valued overlap functions and its influence in interval-valued fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 4, pp. 1060–1072, 2022.

[15] D. Gómez, J. T. Rodríguez, J. Montero, H. Bustince, and E. Barrenechea, "n-dimensional overlap functions," *Fuzzy Sets Syst.*, vol. 287, pp. 57–75, 2016.

[16] J. Montero, R. G. del Campo, L. Garmendia, D. Gómez, and J. T. Rodríguez, "Computable aggregations," *Inf. Sci.*, vol. 460-461, pp. 439–449, 2018.

[17] L. Magdalena, L. Garmendia, D. Gómez, and J. Montero, "Hierarchical computable aggregations," in *IEEE Int. Conf. Fuzzy Syst.*, 2022.

[18] J. Baz, I. Díaz, L. Garmendia, D. Gómez, L. Magdalena, and S. Montes, "Computable aggregations of random variables," *Inf. Sci.*, vol. 654, p. 119842, 2024.

[19] L. Magdalena, D. Gómez, L. Garmendia, and J. Montero, "Population monotonicity of non-deterministic computable aggregations," in *IEEE Int. Conf. Fuzzy Syst.*, 2021.

[20] ——, "Analysing monotonicity in non-deterministic computable aggregations: The probabilistic case," *Inf. Sci.*, vol. 583, pp. 288–305, 2022.

[21] G. Mayor and T. Calvo, "On extended aggregation functions," in *The 7th Int. Fuzzy Syst. Assoc. world Congr.*, Prague, 1997, p. 281–285.

[22] T. Calvo, A. Kolesárová, M. Komorníková, and R. Mesiar, *Aggregation Operators: Properties, Classes and Construction Methods*. Heidelberg: Physica-Verlag HD, 2002, pp. 3–104.

[23] M. Grabisch, J. Marichal, R. Mesiar, and E. Pap, *Aggregation Functions*. Cambridge: Cambridge University Press, 2009.

[24] K. Rojas, D. Gómez, J. Montero, and J. T. Rodríguez, "Strictly stable families of aggregation operators," *Fuzzy Sets Syst.*, vol. 228, pp. 44–63, 2013.

[25] P. Olaso, K. Rojas, D. Gómez, and J. Montero, "A generalization of stability for families of aggregation operators," *Fuzzy Sets Syst.*, vol. 378, pp. 68–78, 2020.

[26] G. Beliakov, S. James, A. Kolesárová, and R. Mesiar, "Cardinality-limiting extended pre-aggregation functions," *Inf. Fusion*, vol. 76, pp. 66–74, 2021.

[27] R. Mesiar, A. Kolesárová, D. Gómez, and J. Montero, "Set-based extended aggregation functions," *Int. J. Intell. Syst.*, vol. 34, no. 9, pp. 2039–2054, 2019.

[28] A. Zubaroglu and V. Atalay, "Data stream clustering: A review," *Artif. Intell. Rev.*, vol. 54, pp. 1201–1236, 2021.

[29] C. Nordahl, V. Boeva, H. Grahn, and M. Persson Netz, "EvolveCluster: an evolutionary clustering algorithm for streaming data," *Evol. Syst.*, vol. 13, no. 4, pp. 603–623, Aug. 2022.

[30] J. Chen, S. Yang, C. Fahy, Z. Wang, Y. Guo, and Y. Chen, "Online sparse representation clustering for evolving data streams," *EEE Trans. Neural Netw. Learn. Syst.*, pp. 1–15, 2023.

[31] C. Fahy, S. Yang, and M. Gongora, "Ant Colony Stream Clustering: A Fast Density Clustering Algorithm for Dynamic Data Streams," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2215–2228, Jun. 2019.

[32] V. Huynh and D. Phung, "Streaming clustering with bayesian nonparametric models," *Neurocomputing*, vol. 258, pp. 52–62, 2017.

[33] Y. Yang, B. Chen, and H. Liu, "Memorized variational continual learning for dirichlet process mixtures," *IEEE Access*, vol. 7, pp. 150 851–150 862, 2019.

[34] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer US, 1981.

[35] A. Bechini, F. Marcelloni, and A. Renda, "TSF-DBSCAN: A Novel Fuzzy Density-Based Approach for Clustering Unbounded Data Streams," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 3, pp. 623–637, 2022.

[36] P. Hore, L. O. Hall, and D. B. Goldgof, "Single pass fuzzy c-means," in *IEEE Int. Conf. Fuzzy Syst.*, 2007.

[37] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM Sigmod Rec.*, vol. 25, no. 2, pp. 103–114, 1996.

[38] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *The 29th Int. Conf. Very Large Data Bases*, 2003, pp. 81–92.

[39] P. A. Lopes and H. A. Camargo, "Fuzzstream: Fuzzy data stream clustering based on the online-offline framework," in *IEEE Int. Conf. Fuzzy Syst.*, 2017.

[40] J. Yu and H. Huang, "A new weighting fuzzy c-means algorithm," in *IEEE Int. Conf. Fuzzy Syst.*, vol. 2, 2003, pp. 896–901.

[41] X. Xiong, K. L. Chan, and K. L. Tan, "Similarity-driven cluster merging method for unsupervised fuzzy clustering," in *The 20th Conf. Uncertain. Artif. Intell.*, 2004, pp. 611–618.

[42] L. Schick and P. L. H. Camargo, "Merging clusters in summary structures for data stream mining based on fuzzy similarity measures," in *The 11th Conf. Eur. Soc. Fuzzy Log. Technol.*, 2019, pp. 812–819.

[43] L. De Miguel, R. Santiago, C. Wagner, J. M. Garibaldi, Z. Takáč, A. F. R. L. de Hierro, and H. Bustince, "Extension of restricted equivalence functions and similarity measures for type-2 fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 9, pp. 4005–4016, 2022.

[44] H. Bustince, E. Barrenechea, and M. Pagola, "Restricted equivalence functions," *Fuzzy Sets Syst.*, vol. 157, no. 17, pp. 2333 – 2346, 2006.

[45] S. Garcia-Jimenez, H. Bustince, E. Hüllermeier, R. Mesiar, N. R. Pal, and A. Pradera, "Overlap indices: Construction of and application to interpolative fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 4, pp. 1259–1273, 2015.

[46] Computational Intelligence Group, "Data stream repository," 2017.

[47] D. G. Márquez, A. Otero, P. Félix, and C. A. García, "A novel and simple strategy for evolving prototype based clustering," *Pattern Recognit.*, vol. 82, pp. 16–30, 2018.

[48] X. Zhu. (2010) Stream data mining repository. [Online]. Available: http://www.cse.fau.edu/xqzhu/stream.html

[49] U. Oceanic and A. A. (NOAA). (2012) Fed. climate complex global surface summary of day data - v. 7 - USAF Datsav3 Station #725540. [Online]. Available: ftp://ftp.ncdc.noaa.gov/pub/data/gsod

[50] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 841–847, 1991.

[51] M. Moshtaghi, J. C. Bezdek, S. M. Erfani, C. Leckie, and J. Bailey, "Online cluster validity indices for performance monitoring of streaming data clustering," *Int. J. Intell. Syst.*, vol. 34, pp. 541–563, 2019.

[52] L. Hubert and P. Arabie, "Comparing partitions," *J. Clas.*, vol. 2, pp. 193–218, 1985.

[53] 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, 2016.

[54] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *SIAM Int. Conf. Data Min.*, 2006, pp. 328–339.

[55] D. Sculley, "Web-scale k-means clustering," in *The 19th Int. Conf. WWW*, 2010, p. 1177–1178.

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

[57] L. Zhang, H. Wang, C. Chen, C. Xu, and P. Yu, "Incremental-concurrent fusion checking for efficient context consistency," *J. Syst. Softw.*, vol. 207, p. 111852, 2024.

[58] G. Bahle, A. Poxrucker, G. Kampis, and P. Lukowicz, "Incremental classifier fusion for smart societies," in *Int. Conf. Electron. Gov. Open Soc. - Chall. in Eurasia*. ACM, 2016, p. 35–40.

[59] C. S. Hara, C. D. de Aguiar Ciferri, and R. R. Ciferri, *Incremental Data Fusion Based on Provenance Information*. Berlin: Springer, 2013, pp. 339–365.