Incremental clustering based on Wasserstein distance between histogram models

Xiaotong Qian, Guénaël Cabanes, Parisa Rastin, Mohamed Alae Guidani, Ghassen Marrakchi, Marianne Clausel, Nistor Grozavu

Pattern Recognition 162 (2025) 111414

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Incremental clustering based on Wasserstein distance between histogram models☆

Xiaotong Qian [a,*], Guénaël Cabanes [b], Parisa Rastin [b], Mohamed Alae Guidani [c], Ghassen Marrakchi [d], Marianne Clausel [b], Nistor Grozavu [a]

[a] ETIS, UMR 8051, CY Cergy Paris Université, Cergy, 95000, France
[b] LORIA, UMR 7503, Université de lorraine, Vandoeuvre-lès-Nancy, 54500, France
[c] École nationale supérieure des mines de Nancy, Campus Artem, Nancy, 54042, France
[d] LIPN, UMR 7030, Université Sorbonne Paris Nord, Villetaneuse, 93430, France

ARTICLE INFO

Keywords:
Unsupervised learning
Static and dynamic clustering
Large datasets
Data streams
Sliding windows
Histogram models
Wasserstein distance

ABSTRACT

In this article, we present an innovative clustering framework designed for large datasets and real-time data streams which uses a sliding window and histogram model to address the challenge of memory congestion while reducing computational complexity and improving cluster quality for both static and dynamic clustering. The framework provides a simple way to characterize the probability distribution of cluster distributions through histogram models, regardless of their distribution type. This advantage allows for efficient use with various conventional clustering algorithms. To facilitate effective clustering across windows, we use a statistical measure that allows the comparison and merging of different clusters based on the calculation of the Wasserstein distance between histograms.

1. Introduction

Machine learning is a prominent field of research that focuses on learning patterns and relationships within datasets to make intelligent predictions or analyzes. It includes two main types of algorithms: supervised learning and unsupervised learning. They are often referred to as classification and clustering [1], respectively. Although both aim to separate and group objects, there is a fundamental difference between them. In classification, the categories are predetermined and objects are assigned to a specific category (called a label). The labels are therefore necessary to train a classification model. In clustering, on the other hand, labels are not needed to train a clustering model; the goal is to group similar objects based on some definition of distance or similarity. Depending on how the data are processed, clustering analysis can be divided into two forms: conventional clustering and data stream clustering.

Conventional clustering algorithms [2] focus more on static datasets that are fixed and remain the same throughout training. With the rapidly increasing volume of data collection, clustering approaches capable of handling large and high-dimensional data have become a popular topic. In [2], the authors present clustering components and detail key concepts related to clustering algorithms, new variants, similarity/dissimilarity measures, optimization challenges, validation, and data types. Although there are subtle differences in categorizing clustering algorithms, several broad categories are popular: hierarchical, partition, graph, density, model and grid-based clustering. Hierarchical-based clustering iteratively divides the dataset into subsets from top to bottom, or merges individual objects from bottom to top, forming a dendrogram by grouping data objects into a tree of clusters; popular examples of this type include BIRCH [3] and Agglomerative clustering [4]. Partition-based clustering aims to separate objects into K groups by optimizing some criterion such as minimizing the total intra-cluster distance; K-means [5] and ORCLUS [6] are typical examples. Graph-based clustering, such as Spectral clustering [7], uses the concept of graphs to represent data points and their relationships and involves the task of dividing nodes into clusters. Density-based clustering, such as DBSCAN [8], relies on notions of density within "neighborhoods" to determine clusters, growing a cluster as long as the density in the neighborhood exceeds some threshold. Model-based clustering assumes models for clusters and tries to best fit the data to the assumed model, for example, Gaussian Mixture [9] assumes that data points are derived from a combination of Gaussian distributions with different parameters.

☆ This work was funded through the ANR project Pro-TEXT (project N° ANR-18-CE23-0024-01). More details are available at: https://pro-text.huma-num.fr/le-projet/.

* Corresponding author.
E-mail address: xiaotong.qian@ensea.fr (X. Qian).

https://doi.org/10.1016/j.patcog.2025.111414
Received 11 August 2023; Received in revised form 6 January 2025; Accepted 25 January 2025
Available online 4 February 2025

Finally, grid-based clustering divides the data space into multiple cells to form a grid structure and obtains clusters based on cells within the same grid; algorithm such as CLIQUE [10] is a representation of this type. These clustering algorithms provide valuable insights and form the basis for data analysis in various fields. In addition, several clustering algorithms could be classified into multiple categories, for instance KASP [11], which is considered as a member of the large-scale clustering algorithms because it uses a sampling approach to deal with very large datasets. It works by applying K-means to identify subsets of the datasets and then performing Spectral clustering on those subsets; or like HDBSCAN [12] which uses a hierarchical approach to identify clusters after determining relationships between objects based on density.

Data stream clustering [13] differs from conventional clustering. It operates on continuous data streams, eliminating the need to wait for the entire dataset to be collected. This real-time nature allows it to dynamically process incoming data points and adapt the clustering models to any changes in the data distribution over time. Many data stream clustering approaches can be considered extensions or variations of conventional clustering, resulting in different categories within the field. The first proposed data stream clustering STREAM [14] is of the partition-based type. This algorithm reads a data stream over fixed size batches or windows, applies KMedian clustering to each batch, and merges the obtained centers into K medians to find final clusters. A few years later, the authors proposed an improved version, STREAMLSEARCH [15], by adding local search techniques. Other algorithms such as StreamK-means [15], and SequentialK-means [16] are also representations of the partition-based type. CluStream [17], a well-known hierarchical and partition-based approach, extends the clustering feature tree (CF-tree) structure of BIRCH. By using an online-offline algorithm with K-means, it effectively summarizes micro-clusters and preserves them at specific points in time within a pyramidal time frame. In addition, density-based approaches have been proposed, such as DenStream [18], which is based on the DBSCAN algorithm with the addition of a CF-tree structure, DBStream [19], which uses a shared density graph to connect micro-clusters, eliminating gaps in dense areas during online clustering refinement. These types of algorithms have the ability to detect clusters of any shape without prior knowledge of the number of clusters, and are able to deal with outliers. Apart from these, there are other types of data stream clustering, including grid-based approach such as DGClust [20], as well as model-based algorithm SWEM [21].

A wide variety of clustering algorithms exist for both static and dynamic datasets. However, no clustering algorithm is universally perfect; each type of clustering algorithm has its advantages and disadvantages. For example, partition-based algorithms have limitations: they require the number of clusters to be specified in advance and struggle with outliers, but they work efficiently and can handle large datasets with relatively low time complexity. Meanwhile, density-based clustering algorithms work on any kind of cluster distribution, automatically detect the number of clusters, and handle outliers very well, but are slower and more difficult to parameterize. Consequently, clustering algorithms can be applied in different domains and the applicability of different clustering approaches depends strongly on the context of the specific problem to solve. The application or choice of different algorithms is therefore highly dependent on the specific requirements or needs of the task. In this article, we present a fast, innovative clustering framework that is able to adapt most clustering approaches to both static and dynamic datasets. The proposed framework operates on sliding windows and uses histogram models to characterize clustering distributions. Histogram models provide simple representations of clusters without requiring prior knowledge of their distribution. To compare and merge different clusters in different windows, we take advantage of the calculation of the Wasserstein distance between histograms [22], facilitating effective cluster analysis and synthesis. A major advantage of the proposed framework is that it allows fast application of most clustering algorithms, minimizing computational cost and memory consumption without using resource-intensive techniques such as distributed/parallel processing or GPU computing. The work presented in this paper fits into the context of frugal clustering [23], which is particularly relevant in resource-constrained environments where traditional clustering algorithms may not be feasible or efficient.

The paper is organized as follows. In Section 2, we provide an overview of the theoretical background, including the construction of the histogram models, the computation of the similarities between distributions using Wasserstein distance. In Section 3, we delve into the details of the approach, outlining the process of modeling clusters as histograms and the subsequent merging of different windows. In Section 4, we conduct experiments to compare the static and dynamic approaches to conventional clustering and data stream clustering on both artificial and real datasets. Finally, in Section 5, we summarize the results and discuss potential perspectives for future research.

2. Background

In this section, we present the theoretical background on which our approach is based, aiming at efficiently comparing cluster distributions with minimal computational cost to determine if clusters should remain separate or be merged. For this purpose, representing the distribution of each cluster as a set of histograms, and then compare these distributions using Wasserstein distance adapted to this representation can be an efficient solution. Finally, a specific two-sample test is computed from this measure to assess the significance of the distance obtained.

2.1. Data distributions using histograms

Estimating probabilistic data distributions [24] is a fundamental part of data science and machine learning, especially in unsupervised learning. Unraveling the distribution of data allows the discovery of hidden patterns in datasets and is a powerful tool for analyzing large amounts of data in an unsupervised manner. In clustering applications, clusters are often assumed to be normally distributed (i.e., have a multivariate Gaussian distribution) or sometimes have a more complex distribution, such as gamma distributions. All subsequent analysis is then based on this assumption. These parametric approaches can provide good results, but they have limited applications when it comes to representing clusters of arbitrary distributions, which is an important requirement for many real-world scenarios. One possible solution is to model the distribution as a mixture of simple functions (usually Gaussian). However, Gaussian mixture models can be computationally expensive and slow to train, especially when the number of mixture components is large. They are also sensitive to initialization, and choosing the number of mixture components in the model is often challenging. Another solution, which we focus on in this paper, is to model the cluster distributions using a set of histograms computed from the empirical distribution [25] of the data. Although this representation can lead to a loss of fine variations of the underlying distribution depending on the number of bins, it has the advantage of being independent of any assumptions and is fast and easy to compute. It also greatly simplifies the computation of the Wasserstein distance [26].

The term histogram was first proposed by [27], who described a histogram as a series of rectangles of equal width whose height could represent the number of values falling within the interval formed by their two edges. Using histograms to represent data could be a concise and flexible case of symbolic or summarized data analysis when faced with large amounts of data. In this case, the weight of each rectangle of the histogram is no longer the number of observations, but the probability or proportion of values over a set of intervals, formally defined that a histogram is a model for representing the empirical distribution of a continuous variable Y divided into a set of contiguous I_ϕ intervals (bins) with associated π_ϕ weights. A histogram H is thus

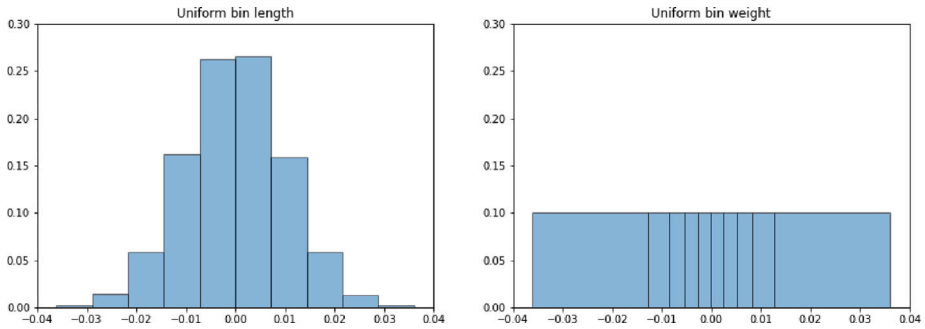

Fig. 1. Two types of histogram representations which display a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ where μ is 0 and σ is 0.01.

represented by a set of Φ ordered pairs (I_ϕ, π_ϕ), where π_ϕ is a non-negative measure of a probability distribution on the domain of Y such that: (1) $\sum_{\phi=1}^{\Phi} \pi_\phi = 1$ with $\pi_\phi \geq 0$; (2) $I_\phi \cap I_{\phi'} = \emptyset, \phi \neq \phi'$; (3) $\bigcup_{\phi=1,\ldots,\Phi} I_\phi = [Y_{min}, Y_{max}]$.

There are two ways of displaying histograms. The first type allows each bin to have a fixed uniform length, but not a uniform weight, expressed as: $|I_\phi| = \frac{Y_{max}-Y_{min}}{H}$. The second type allows each bin to have a fixed uniform weight, but not a uniform length, expressed as: $|\pi_\phi| = \frac{1}{H}$. Fig. 1 illustrates the difference.

The time complexity of computing a univariate histogram depends on the number of bins Φ in the histogram and the number of data points n being processed which is $O(n * \Phi)$, where Φ is usually much smaller than n and need not be proportional to n. However, computing a multivariate histogram is much more expensive because the number of bins Φ in each dimension should be considered, with a time complexity of $O(n * \Phi^v)$, where v is the dimensionality of the dataset. One way to reduce the computational cost of multivariate histograms is to compute separate independent histogram for each variable, which reduces the complexity to $O(n * v * \Phi)$, but correlation information is lost in this approach. If some attributes are correlated, we cannot treat their histogram representation independently. For these reasons, in this paper we focus on independent histograms and deal with the loss of correlation information by using random projections which will be presented in Section 3.1.

2.2. Distribution comparison

To increase storage efficiency and achieve more consistent clustering results, it is imperative to compare the distributions of different clusters once they have been identified. Then, clusters with similar distributions should be merged to achieve better clustering results. However, comparing different distributions can be challenging and requires a suitable metric to assess the similarity between two distributions μ and v on the same sample space $\mathcal{X}$. A widely used measure is the Kullback–Leibler divergence (KL divergence expressed in Eq. (1)) [28], which computes the expectation of the logarithmic difference between the probability distributions μ and v:

$$KL(\mu, v) = \int_{\mathcal{X}} \mu(x) log(\frac{\mu(x)}{v(x)}) dx. \tag{1}$$

Since the KL divergence is not symmetric, the Jensen–Shannon divergence (JS divergence expressed in Eq. (2)) [29] has been proposed to solve the symmetry problem, known as the total divergence to the average, the square root of the JS divergence is a metric often referred to as the JS distance.

$$JS(\mu, v) = \frac{KL(\mu, \frac{\mu+v}{2}) + KL(v, \frac{\mu+v}{2})}{2} \tag{2}$$

The KL and JS divergences suffer from several drawbacks, as non robustness to outliers. In addition, KL and JS are not distances and they do not satisfy triangle inequality. For all these reasons, another metric, the so called Wasserstein distance has gained in popularity. Wasserstein distance takes into account the cost of moving the mass from one distribution to another, which helps to mitigate the impact of outliers. The Wasserstein distance is also a continuous function, meaning that small changes in the input distributions result in small changes in the distance measurement. This is not the case with KL divergence, which can exhibit discontinuities and is therefore less suitable for some applications. In addition, Wasserstein distance satisfies the properties of a metric, such as symmetry, triangle inequality, and non-negativity. This property is not satisfied by JS and KL distances, making Wasserstein distance more suitable for use in optimization and clustering algorithms. Finally, Wasserstein distance has a natural interpretation as the minimum cost of transforming one distribution to another, which makes it easier to understand and explain in comparison to other distance measures.

Originally introduced in the context of optimal transport theory [30], this metric quantifies the minimum cost required to transform one probability distribution into another, where cost is defined as the amount of "work" required to move a given amount of mass from one point to another. The traditional approach to estimating the Wasserstein distance is outlined in [31] with a computational complexity of $O(n^3 log(n))$ using the EMD definition of this distance.

$$W_{emd}(\mu, v) = \min_{\gamma \in \mathcal{P}} \int \|x - y\| \gamma(x, y) dx dy = \mathop{\mathbf{E}}_{(x,y)\sim\gamma} [\|x - y\|] \tag{3}$$

where $\mathcal{P}$ is the set of all joint distributions $\Pi(\mu, v)$ whose marginals are μ and v, $\|x-y\|$ gives the cost of transporting a unit of mass from point x to point y. A transport plan to move μ to v can be described by a function $\gamma(x, y)$ which gives the amount of mass to be moved from x to y.

The Sinkhorn distance [32] is a faster alternative to Wasserstein distance that uses an entropy regularization term into the optimal transport problem, making the problem more computationally efficient to solve with computational complexity up to $O(n^2)$. Given a cost matrix M, the cost of mapping μ to v using the joint probability $\mathcal{P}$ can be qualified as $\langle \mathcal{P}, M \rangle$, and the Sinkhorn distance directly regularizes the original transport problem with a regularization parameter λ and the entropy $h(\mathcal{P}) = -\sum_{i,j=1} \mathcal{P}_{i,j} log \mathcal{P}_{i,j}$ defined as follow:

$$\lambda > 0, W^{\lambda}_{sinkorn}(\mu, v) := \langle \mathcal{P}^{\lambda}, M \rangle, \ \mathcal{P}^{\lambda} = \text{argmin} \langle \mathcal{P}, M \rangle - \frac{1}{\lambda} h(\mathcal{P}) \tag{4}$$

Computation of the Wasserstein distance (including Sinkhorn distance) can become exceedingly complex for large datasets, making it necessary to explore alternative methods. One such approach, which is specifically designed for histograms, can be found in [22]. This approach introduces an elegant formulation of the Wasserstein distance between two histograms H^i and H^j expressed as:

$$W^2_{hist}(H^i, H^j) = \sum_{\phi=1}^{\Phi} \pi_\phi d^2(I^i_\phi, I^j_\phi) = \sum_{\phi=1}^{\Phi} \pi_\phi \left[(c^i_\phi - c^j_\phi)^2 + \frac{1}{3}(r^i_\phi - r^j_\phi)^2 \right], \tag{5}$$

which simply compute the sum of the differences between each pair of centers C and radius R of two histograms, where π_ϕ is the probability weight associated to each bin, and I^i_ϕ represents an interval $([\underline{y^i_\phi}, \overline{y^i_\phi}])$

of histogram H^i, I^j_ϕ represents an interval $([\underline{y^j_\phi}, \overline{y^j_\phi}])$ of histogram H^j, with:

$$c^i_\phi = \frac{\underline{y^i_\phi} + \overline{y^i_\phi}}{2}; \quad c^j_\phi = \frac{\underline{y^j_\phi} + \overline{y^j_\phi}}{2}; \quad r^i_\phi = \frac{\underline{y^i_\phi} - \overline{y^i_\phi}}{2}; \quad r^j_\phi = \frac{\underline{y^j_\phi} - \overline{y^j_\phi}}{2}.$$

They also propose a way to compute the Wasserstein distance between multivariate histograms. Suppose there are p histogram variables for observation i and j ($H^i_1 \cdots H^i_p$ and $H^j_1 \cdots H^j_p$), under the hypothesis that the variables are independent, the process is expressed like:

$$W^2_{hist}[(H^i_1 \cdots H^i_p), (H^j_1 \ldots H^j_p)] = \sum_{q=1}^{p} W^2_{hist}(H^i_q, H^j_q) \tag{6}$$

The proposed approach described in this paper is to compute histograms that model the data distribution before computing the distance between the centers and the radius of each interval. The time complexity of computing the Wasserstein distance between histograms is $O(\Phi^v)$ for multivariate histograms and $O(\Phi * v)$ for independent histograms. Overall, since Φ is usually chosen as a constant positive parameter, the total time complexity of this approach for a set of v independent histograms can be efficient compared to the complexity of multivariate histograms. Moreover, the time complexity of an algorithm also depends on its implementation and hardware. For example, parallelization can greatly reduce the complexity of independent histograms.

It is possible to compute the barycenter of a set of histograms by defining a distance measure between histograms. This barycenter is essentially a single histogram that minimizes the squared Wasserstein distance between itself and each member of the set. In other words, it is the optimal representative histogram that captures the essential features of the entire set. In the case of histograms with uniform bins weight, each one is described by the centers C and radius R of its bins, this involves computing new centers C^b and radius R^b while keeping the weight of each bin constant. The Eq. (7), mentioned in [22], shows that this simply requires computing the mean center and radius of each bin, where N_h is the number of histograms in the set, in our case N_h is set to 2:

$$C^b = N_h^{-1} \sum_{j=1}^{N_h} C^j \text{ and } R^b = N_h^{-1} \sum_{j=1}^{N_h} R^j. \tag{7}$$

2.3. Wasserstein two-samples testing

Performing non parametric two sample testing allows to detect, given samples $X_1, \ldots, X_n \sim \mu$ and $Y_1, \ldots, Y_m \sim \nu$ from the two unknown distributions μ and ν, if they are significantly different. One classical test is the Kolmogorov–Smirnov test (see [33]), or K-S test, which is commonly used to compare an empirical distribution to a theoretical distribution or to compare two empirical distributions. In the case of deciding whether two one-dimensional probability distributions (μ, ν) differ from each other by computing Eq. (8):

$$D_{\mu,\nu} = \sup_x |F_\mu(x) - F_\nu(x)| \tag{8}$$

and test the null hypothesis $(H_0) : \mu = \nu$ versus $(H_1) : \mu \neq \nu$. One then rejects the null hypothesis (H_0) and accepts (H_1) with significance level α if

$$D_{\mu,\upsilon} > C(\alpha)\sqrt{\frac{n+m}{n \cdot m}}, \text{ where } C(\alpha) = \sqrt{-ln(\frac{\alpha}{2}) \times \frac{1}{2}}.$$

An alternative distribution free statistical test based on the Wasserstein distance has been proposed in [34]. This test is based on the following preliminary result on the asymptotic empirical Wasserstein distance between two samples. Denote F_n, G_m, the two empirical cumulative distribution functions (CDF) associated to the samples $X_1, \ldots, X_n \sim \mu$ and $Y_1, \ldots, Y_m \sim \nu$. Under the null hypothesis $H_0 : \mu = \nu$, one has

$$T_{m,n} := \frac{n \cdot m}{n+m} \int_0^1 \left(G_m \left(F_n^{-1}(t) - t \right)^2 \right) dt \rightarrow_w Z$$

$$:= \int_0^1 \mathbf{B}(t)^2 \, dt \text{ as } n, m \rightarrow \infty \tag{9}$$

where $B(t)$ represents the Brownian bridge [35]. The Brownian bridge is defined as $B(t) = W_{pr}(t) - \frac{t}{T} W_{pr}(T)$, where $t \in [0, T]$ (in the assumption $T = 1$) and W_{pr} is a standard Wiener process [36]. It describes a random walk from 0 to T starting at 0 and ending at 0, such as $W_{pr}(0) = W_{pr}(T) = 0$. We emphasize that on can simulate the Brownian Bridge and then compute empirically the quantiles of the random variable Z. Using the test statistic $T_{m,n}$, we then rejects H_0 and accepts H_1 if $T_{m,n} > z_\alpha$ where z_α is the α-quantile of distribution Z. In short, instead of testing the vanishing of Wasserstein distance between G_m and F_n directly, this test computes the Wasserstein distance between $G_m(F_n^{-1})$ and a uniform distribution $U_{[0,1]}$ on $[0, 1]$ which makes it distribution free.

3. Proposed approach

The proposed approach uses a non-overlapping sliding window model to run a clustering algorithm on batches of data samples that are briefly held in memory. Then allowing any type of clustering algorithms to be applied across sliding windows. This flexibility is a key strength of our approach, allowing the selection of the most appropriate clustering method for the data at hand. The ability to remove noise using algorithms such as those in the DBSCAN family is particularly valuable in this context, as histograms can be sensitive to extreme values and outliers. However, our approach has shown excellent overall performance in tests with other types of clustering algorithms (see Section 4). This allows incremental clustering of large datasets and handling of data streams. The distribution of each cluster in a window is modeled as a set of unidimensional histograms computed in a random projection space, capturing correlation information between variables without the need to compute a costly multivariate histogram.

As shown in Fig. 2, input data points are continuously collected within a sliding window. A conventional clustering algorithm is then applied to identify clusters within the current window. Random projections of these data points are then computed (see Section 3.1), and the distributions of the detected clusters is modeled using a histogram plot applied on each of the projection axes (see Section 3.2). To compare the newly computed clusters with those from previous windows, we use the Wasserstein two-sample test based on the Wasserstein distance, a statistical test that aims to identify clusters with similar distributions (see Section 3.3). These similar clusters are merged and replaced by their barycentric histograms using a modified calculation form (see Section 3.4). Once this process is complete, the histogram models are stored and the data samples are discarded to leave space for a new window of data samples in memory. The main steps of the approach are also summarized in the Algorithm 2 (see Section 3.5).

3.1. Multivariate histogram extension

Multivariate histogram models to represent the data distribution can be very computationally costly when the number of dimensions is high. Instead of using a multivariate histogram, the distribution can be modeled as a set of univariate histograms at very low computational cost, but all correlation information is lost. One solution inspired by Sliced Wasserstein distance proposed in [37] is to compute random projections of the data onto new axes drawn uniformly on the unit sphere. If the number of new axes is large enough, the resulting model of the distribution is very close to the multivariate model in terms of computing the Wasserstein distance, while reducing the computational complexity very significantly. The axes of the projections of the data points are obtained via the dot product between the sample vectors X and M_p, the intermediate matrix M_p is expressed in Eq. (10) as follow:

$$M_p = \frac{M^2_\Psi}{[\sqrt{\sum_{i=1}^{v} (\psi_i^1)^2}, \ldots, \sqrt{\sum_{i=1}^{v} (\psi_i^p)^2}]}, \qquad X^p = X \cdot M_p \tag{10}$$

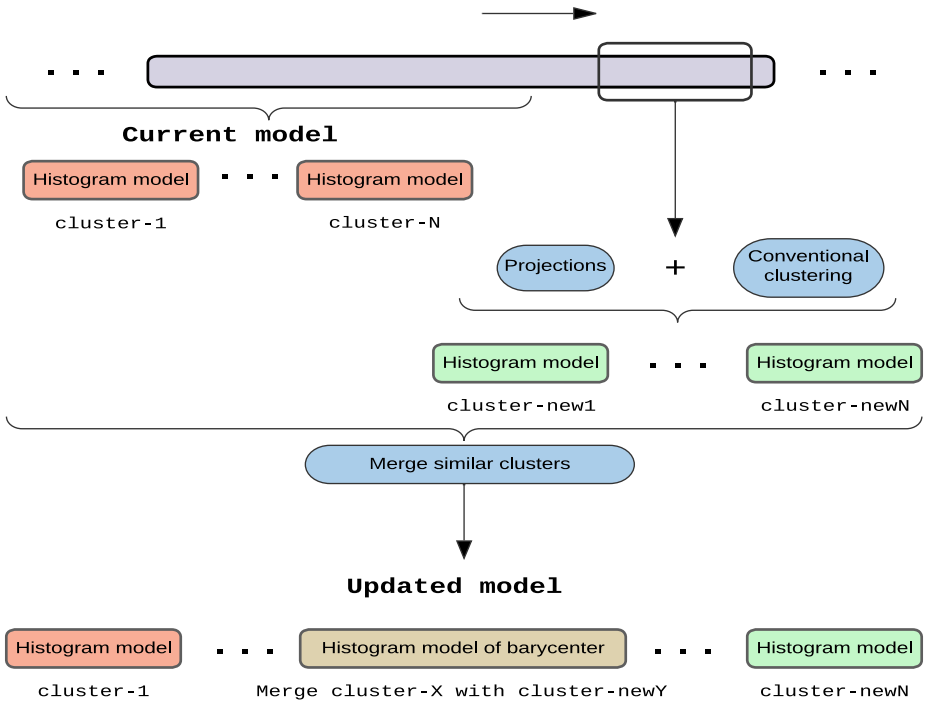

Fig. 2. Proposed approach.

where M_Ψ is a matrix which follows a standard normal distribution with shape (v, p), v is the number of variables, and p is the number of projections, and $\psi_i^j \in M_\Psi$.

3.2. Histograms computation

As mentioned in Section 2.1, we favor a fixed weight representation for the histograms, which greatly simplifies the computation of the Wasserstein distance introduced in Eq. (5), Eq. (6). In the proposed approach, each cluster is represented by a set of independent histograms. For this purpose, the samples of a cluster X_c^p obtained in the previous step must be divided into Φ intervals by computing the 1 to Φ-th quantile of X_c^p along each dimension, keeping the same weight π_ϕ for each interval, which is equal to $\frac{1}{\Phi}$. This process is repeated for all clusters in the current window.

3.3. Cluster similarity test

With respect to Eq. (9), the detailed procedures for defining the similarity between two cluster distributions in the form of histograms are as follows:

1. Determine the Brownian bridge threshold δ corresponding to the dimension p using linear regression.
2. Obtain the CDF F_n and G_m for each pairwise histogram distribution μ and v of two clusters.
3. Compute the composition function of G_m and the inverse function F_n^{-1} of F_n: $G_m(F_n^{-1})$.
4. Compute the sum of WD between each $G_m(F_n^{-1})$ and $U_{[0,1]}$ by using Eq. (6), then multiply with $\frac{n*m}{n+m}$.
5. Compare result of step 4 with the threshold δ.
6. If the result $< \delta$, it means two clusters are similar and must be merged.

To define the threshold δ, we used simulations of Brownian bridges for multivariate observations. Since the threshold δ corresponds to a Wasserstein distance between a random Brownian bridge and the uniform walk $B_t = 0$ with a probability of occurrence below 5%. The algorithm which computes the threshold δ correspond of p variables could be developed as described in Algorithm 1.

Experiments show that the threshold δ has a linear relationship with the number of dimensions p. Therefore, instead of repeating the process of Algorithm 1, a simple linear regression model trained by δ_1 and δ_2 could be applied to any dimension p, allowing a fast computation of δ_p. Incidentally, the values of M and N do not matter much for the definition of the threshold; to maintain consistency, it is enough to keep them constant for the computation of δ_1 and δ_2. Precisely, we set them both to 10000.

Algorithm 1: Computation of threshold δ corresponding to p dimensions

Input: p, M, N
Output: δ_p
for *i in 1,..,p* **do**
    /* Create M synthetic Brownian Bridges, each one includes N 1d points, i.e. a randomly normal increase/decrease at each time step */
    B[i] ← Create_Brownian_Bridge(M, N);
    MeanB[i] ← mean(B[i]^2, axis=1) ; /* Mean of B[i]^2 by columns */
Sum_MeanB ← Sum_MeanB(MeanB, axis=0) ; /* Sum of MeanB by rows */
/* Extract (95%×M)-th element of sorted Sum_MeanB as threshold */
δ_p ← sort(Sum_MeanB)[M*0.95];

3.4. Merging process

Once similar clusters found, merging them by computing their barycenter in order to reduce the complexity of time and space. Since some clusters may have similar distributions but large differences in

Table 1
Datasets summary.

Real datasets	Nb samples	Nb dimensions	Nb clusters
Outdoor	3501	21	40
Gassenor	13 377	128	6
IGBN	24 150	33	6
IABN	52 848	33	6
Rialto	82 250	27	10
IIAIN	452 044	33	6
IIRIN	452 044	33	6
Covtype	549 829	10	7

Artificial datasets	Nb samples	Nb dimensions	Nb clusters
Comet	3,20(*10^4)	50,100	15,50
Meteorite	3,20(*10^4)	50,100	15,50
Square	3,20(*10^4)	50,100	15,50
Gaussian	3,20(*10^4)	50,100	15,50
Moon	3,20(*10^4)	50,100	15,50
Circle	3,20(*10^4)	50,100	15,50
MixSmall	1,2,3 (*10^4)	50,50,50	24,36,48
MixLarge	10,20,50 (*10^4)	100,100,100	54,66,90

the number of data, the barycenter distributions must be computed as a weighted sum of the mean centers and radius by modifying Eq. (7):

$$C^b = \frac{\sum_{j=1}^{N_h} n_j * C^j}{\sum_{j=1}^{N_h} n_j} \; ; \; R^b = \frac{\sum_{j=1}^{N_h} n_j * R^j}{\sum_{j=1}^{N_h} n_j} \tag{11}$$

Finally, the process of cluster detection and subsequent merging within a current window is achieved. By repeating this process for each window, it is possible to efficiently handle large datasets and data streams.

3.5. Detailed algorithm

The proposed framework can be summarized as follows: First, a clustering algorithm $Clus(\theta, X[nb\_w])$ with parameters θ is applied to data points $X[nb\_w]$ temporarily stored in memory within the nb_w-th time window, and the resulting clusters are stored as $X_c[nb\_w]$. The sliding window ensures incremental data collection both for large datasets and data streams. Random projections are then performed to map the data into a p-dimensional space, stored as $X_c^p[nb\_w]$. Next, the clusters are modeled as empirical distributions in the form of a set of 1-D histograms, notated as $H\_X_c^p[nb\_w]$, applied on each random projection axis. The set of 1-D histograms on random projections approximates the full multidimensional histogram with reduced memory and computational cost. These histograms allow the representation of clusters in arbitrary shapes, which is critical for applying the framework to various clustering algorithms. Data points in the current window are then discarded to save memory. Finally, the computed cluster distributions update the overall data stream model Θ by adding new distributions and merging similar ones, as discussed in Sections 3.3 and 3.4 using the Wasserstein distance metric. The detailed process is shown in the algorithm 2.

Algorithm 2: Proposed approach

Input: Number of projections p, number of bins Φ, data points arriving by windows $\mathbf{X} = \{X[1], X[2]...\}$, conventional clustering algorithm with θ parameters $Clus(\theta)$, similarity threshold δ_p computed by Algorithm 1, stored clusters in current model Θ

Output: Updated clusters Θ

```
nb_w ← Index of current window ;
while length(X[nb_w]) != 0 do
    X_c[nb_w]) ← Clus(θ, X[nb_w]) ;  /* Detect clusters in the nb_w-th window */
    X_c^p[nb_w] ← p Random Projections of X_c[nb_w] by Eq. (10) ;
    H_X_c^p[nb_w] ← Convert all the clusters X_c^p[nb_w] to histogram models as described in Section. 3.2;
    for each cluster i in Θ do
        for each cluster j in H_X_c^p[nb_w] do
            if i and j are not significantly different with respect to Eq. (9) then
                Update i which is stored in Θ by the barycenter of i and j using Eq. (11)
            else
                Θ ← Θ.add(j) ;  /* Add j to the entire set Θ */
    nb_w ← nb_w + 1;
```

4. Experimental protocol and results

4.1. Datasets

To evaluate the proposed approach, we performed a series of experiments on different datasets, including real datasets and artificial datasets, with a variety of number of samples, dimensions and clusters (). Detection of clusters in arbitrary shapes plays an important role in the proposed approach, besides real popular datasets used in recent research, artificial datasets with different cluster distributions are necessarily generated. Right after the creation and collection of real and artificial datasets, two ways of dataset preprocessing are followed. This allows clustering in both static and dynamic contexts, demonstrating the flexibility of the proposed approach.

4.1.1. Artificial datasets

Generate several artificial databases in different distributions to simulate real scenarios, such as comet, meteorite, square, moon, circular, and Gaussian, by adding orientation to some clusters to replicate the real case. For the square distribution, the first dimension follows a univariate Gaussian distribution, then the remaining dimensions follow a uniform distribution. Then comes the comet distribution, where the first dimension follows a gamma distribution, then the remaining dimensions follow a univariate distribution. Similar to the comet distribution, the meteorite distribution follows a gamma distribution for all dimensions. Another type of distribution is the circle distribution, which is based on cosine and sine elements, and the moon distribution could be treated as a semicircle distribution. Then there is the Gaussian distribution, best known as the multinomial Gaussian distribution. An example of the visualization of all artificial distributions could be seen in Fig. 3

4.1.2. Real datasets

The real datasets used in the experimental sections have already mentioned in including **Outdoor** which consists of 4000 images of 40 different objects taken with a smartphone camera in a garden environment. The images were taken under different lighting conditions and from different distances and positions; **Gassor** [38] contains 13910 recordings from 16 chemical sensors measuring six pure gases (ammonia, acetaldehyde, acetone, ethylene, ethanol and toluene) in a gas delivery platform at the University of California, San Diego; **Rialto** [39] contains 82,250 examples of ten colorful buildings near the Rialto Bridge in Venice; **Covtype** [40] contains 581012 instances with 54 attributes related to forest cover type obtained from the US Forest Service Region 2 Resource Information System. And datasets from [41] that involve the identification of disease-carrying insects using optical sensors. The following abbreviations are used to name them based on different impact of temperature: **IABN**, **IGBN**, **IIAIN**, **IIRIN**.

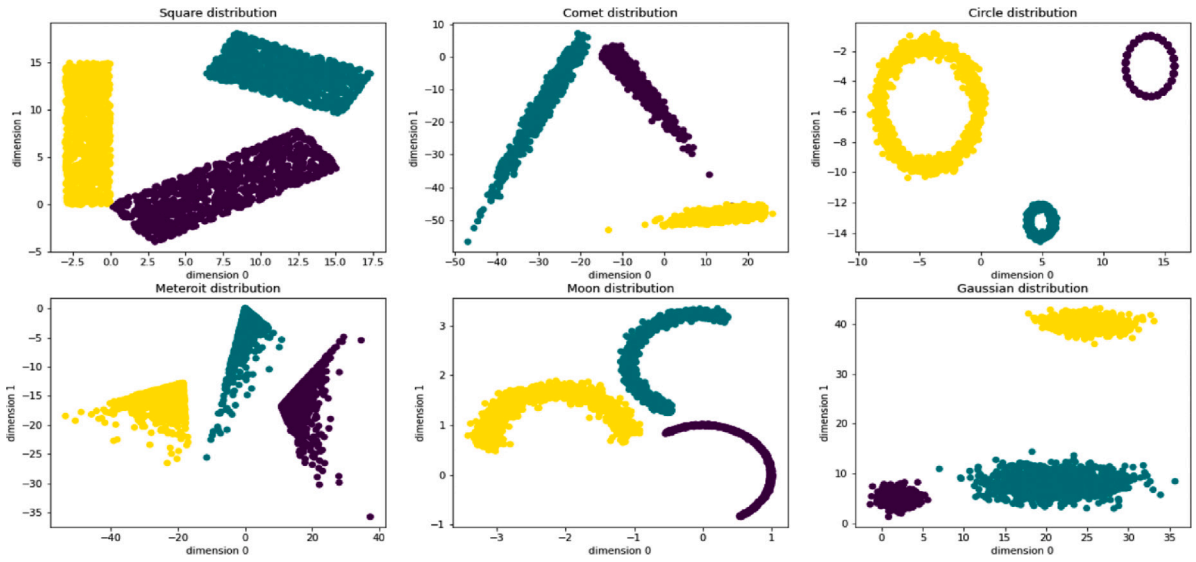

Fig. 3. Artificial datasets of different distributions.

4.1.3. Datasets preprocessing

Since the proposed framework supports both static (batch) and dynamic (data stream) clustering, the experiments are performed on artificial and real datasets suitable for both types of clustering. Therefore, the setup steps are adapted to these two types of scenarios:

- Static Clustering: For the static experiments, the results are compared with conventional clustering applied to the whole datasets. The order of the data is randomly shuffled, then the windows are defined according to a predefined number of data instances per window, usually expressed as a proportion of the number of instances.
- Dynamic Clustering: For the dynamic experiments, the results were compared with popular data stream clustering algorithms. To simulate a dynamic process in the artificial datasets, each data instance was randomly assigned a timestamp. Real datasets were kept with their original timestamps, if available, or were timestamped according to their original order. Windows were defined according to a chosen time interval, and the number of data instances in different windows can vary according to the timestamps of the instances.

In both cases, the goal is to apply conventional clustering on non-overlapping windows to detect clusters before merging similar clusters using the proposed framework.

4.2. Clustering evaluation strategy

The Adjusted Rand Index (ARI), Normalized Mutual Informations (NMI) and Purity are becoming increasingly popular for assessing the quality of clustering, as they are regularly used in a large number of studies. Each of them is an external index based on a priori knowledge of the data structure. Internal indices typically check the compactness and separability of clusters and work well to evaluate clusters that are close to a Gaussian distribution. Due to the diversity of cluster distributions in the experimental datasets, we decided to restrict the evaluation to external indices. For both ARI and NMI, we used the functions implemented in the scikit-learn [42] Python package, which compute the ARI score, bounded between −0.5 and 1, and the NMI score in the range [0,1]. Both indexes are based on the computation of a contingency table (see Table 2).

- **Adjusted Rand Index (ARI)** [43]: By considering all sample pair assignments in both the expected and actual clusterings, as well as counting sample pair assignments in the predicted or actual clusterings the ARI formula is as follow:

$$ARI = \frac{\sum_{i,j}\binom{n_{i,j}}{2} - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{N}{2}}}{\frac{1}{2}\left(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right) - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{N}{2}}}$$

- **Purity** [44]: Each cluster is given a description based on the class that makes up the majority of the cluster. Purity is then computed as the fraction of all data points divided by the number of successfully matched class and cluster labels. The formula is summarized as follows, where n is the number of samples and K is the number of clusters.

$$Purity = \frac{\sum_{k=1}^{K} max(\text{number of majority cluster in k})}{n}$$

- **Normalized Mutual Information (NMI)** [45]: Widely used and increasingly popular metric for evaluating community detection methods, it measures the similarity between two partitions by calculating their mutual information, normalized by the entropies of the partitions. Unlike overlap metrics, NMI does not require that the two partitions have the same number of groups. This makes it particularly useful for assessing the agreement between two independent labeling strategies on the same dataset, even when the true underlying labels are unknown.

$$NMI = \frac{-2\sum_{i,j} n_{i,j}\log\left(\frac{N\cdot n_{i,j}}{b_j\cdot a_i}\right)}{\sum_j b_j \log\left(\frac{b_j}{N}\right) + \sum_i a_i \log\left(\frac{a_i}{N}\right)}$$

As discussed in [46], it is sometimes valuable to consider the "trade-off" between computation time and clustering quality as an additional analysis of clustering performance. This consideration is important because the proposed approach may not always be the best when evaluated solely by the traditional metrics mentioned above. In some cases, its clustering quality may be slightly lower than others. However, if this difference in quality is limited while the approach significantly reduces the computation time, the practical value of the proposed framework is worth considering. To facilitate this discussion, we use a simple index:

$$\text{Trade-off} = \frac{\text{Clustering quality}}{\text{Normalized time}}.$$

Table 2
Each element $n_{i,j}$ of this table correspond to the number of data points in common between cluster $C_p(i)$ and cluster $C_r(j)$.

Real/Predicted clusters	$C_r(1)$	...	$C_r(j)$	...	$C_r(N_r)$	$a_i = \sum_j n_{i,j}$
$C_p(1)$	$n_{1,1}$		$n_{1,j}$		n_{1,N_r}	a_1
...			...			...
$C_p(i)$	$n_{i,1}$	...	$n_{i,j}$	...	n_{i,N_r}	a_i
...			...			...
$C_p(N_p)$	$n_{N_p,1}$		$n_{N_p,j}$		n_{N_p,N_r}	a_{N_p}
$b_j = \sum_i n_{i,j}$	b_1		b_j		b_{N_r}	$N = \sum_{i,j} n_{i,j}$

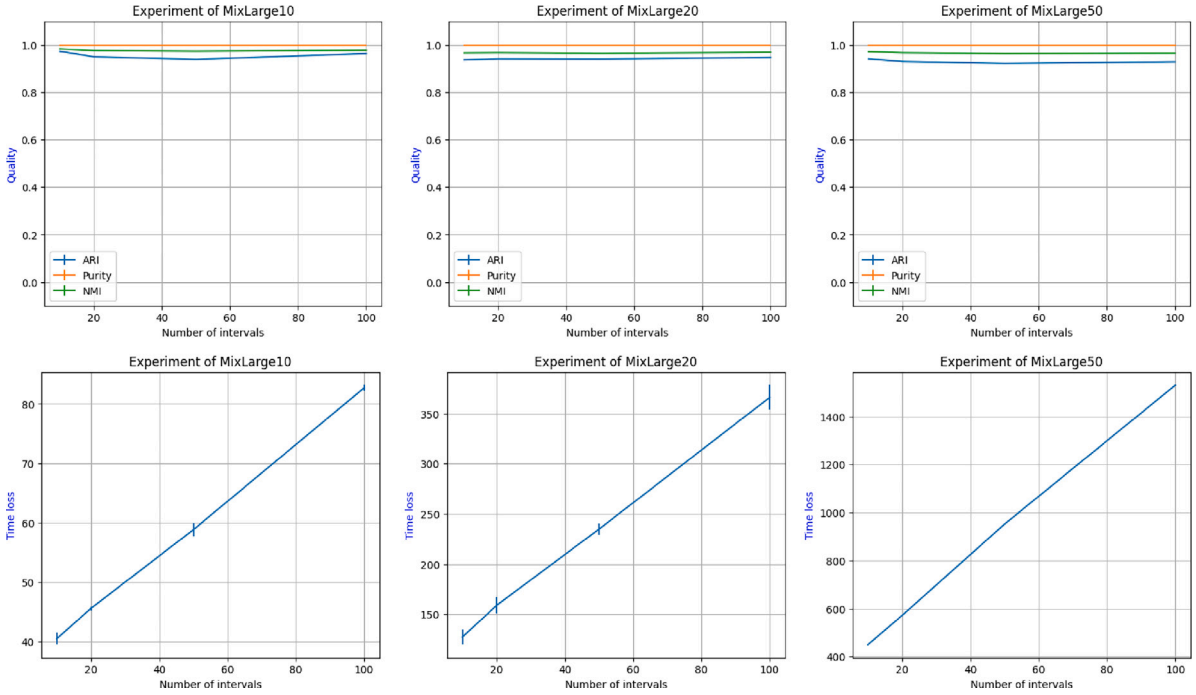

Fig. 4. Results with increasing the number of bins — Artificial datasets.

Here, "normalized time" refers to the computation time normalized by the total time of all algorithms for each dataset. This normalization is useful for visualization purposes. Unlike standard evaluation metrics, this "trade-off" metric provides a quick comparison that highlights the balance between time consumption and clustering quality. While the exact scores/values are not the primary focus of the experiments, the resulting visualizations of this metric help to discuss the importance and efficiency of the proposed approach.

4.3. Parameter analysis

An analysis of the impact of different parameters of the proposed approach on its quality has been carried out. In particular, the number of bins of the histogram model, the number of random projections, and the number of windows (also related to the number of samples in a window), which could change the quality and complexity of the clustering, all these parameters could have an impact on the result of our approach. We ran each test 10 times and report the average performance with the corresponding standard deviation.

4.3.1. Number of bins

As can be seen from the results in Figs. 4–6, the proposed approach performs quite similarly regardless of the number of bins, on both artificial and real datasets. In fact, the quality of the clustering did not change significantly when the number of bins was adjusted. However, it is clear that the computation time is higher when the number of bins is increased. Therefore, in the following experiments, we set the number of bins to 10 in order to be efficient without losing quality.

4.3.2. Number of projections

We found that the quality of clustering does not vary significantly when the number of projections is changed, both on real and artificial datasets, according to the results shown in Figs. 7–9. This seems that reducing the data dimensions still preserves the histogram's ability to efficiently detect clusters with similar distributions. The algorithm runs faster with fewer projections than variables in the dataset, but computation time increases with more projections. Therefore, we set the number of projections to 4 for the following experiments.

4.3.3. Window size

When the datasets are divided into more windows, the proposed cluster similarity test is applied more often on clusters with few samples, potentially leading to more errors, but the process is faster. The aim of testing the effect of changing the number of windows is the same as testing the effect of the number of samples in a window, as well as the effect of the number of times we apply the similarity test. As can be seen in Figs. 10–12, there is a gradual decrease in quality as the number of windows increases, as the number of comparisons increases, although there is also an initial dramatic decrease in computation time followed by convergence after a certain number of windows.

Therefore, to achieve better performance, we set the number of samples or time intervals in each window differently for different datasets. In our experiments, we carefully choose the number of data points (or time intervals) in each batch to ensure the effectiveness of our approach. Striking a balance is essential; using too few data points can result in scattered points, making it difficult to find clusters. Conversely, too many data points would lead to an over-reliance on

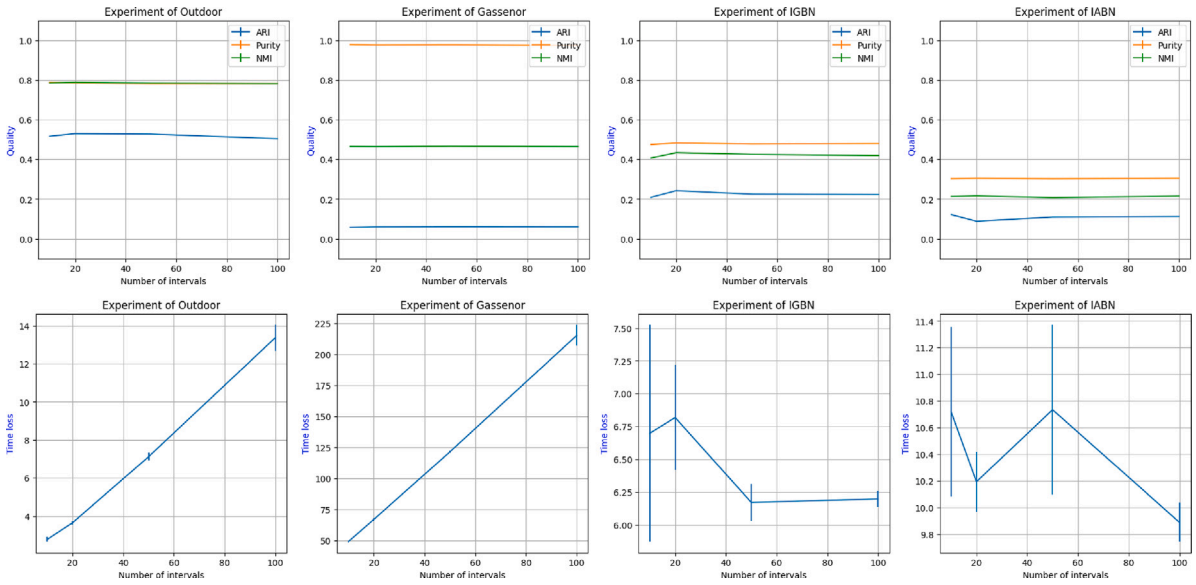

Fig. 5. Results with increasing the number of bins — Real datasets (PART I).

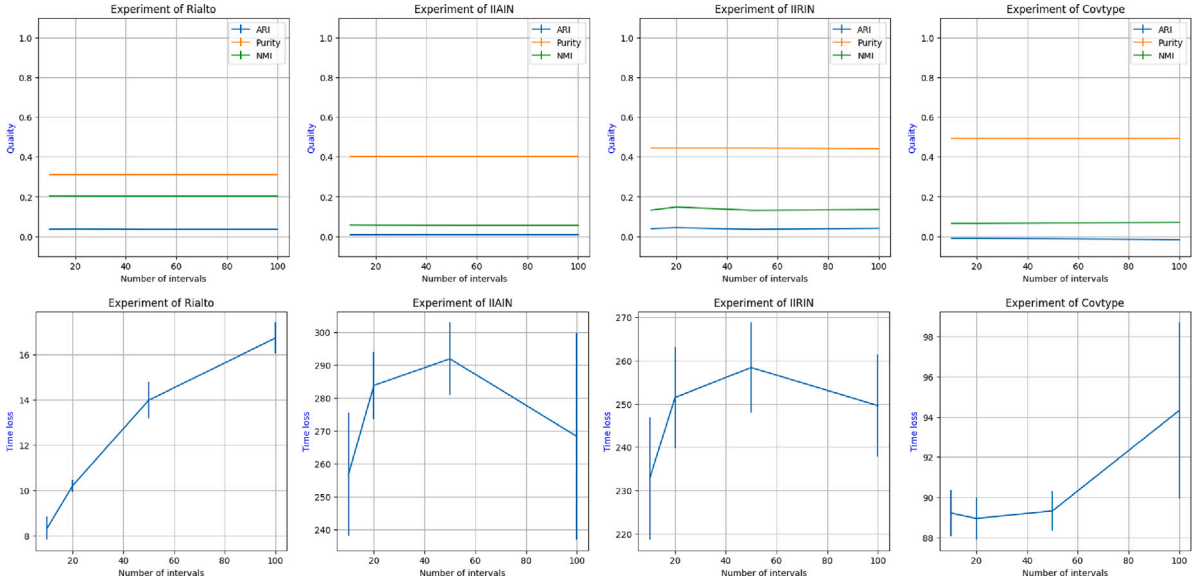

Fig. 6. Results with increasing the number of bins — Real datasets (PART II).

standard clustering within each batch, potentially obscuring the impact of the proposed approach.

To address this, we have developed a systematic configuration for the experiments on both static datasets and dynamic datasets. For smaller datasets, we ensure that the number of data points in each batch is approximately 10% of the total number. For larger datasets, this proportion is reduced to 5%. For artificial datasets, we keep the percentage constant. However, real-world datasets may consist of varying amounts of samples, we allow for slight adjustments to avoid inconsistent number of batches in the analysis. The number of samples in each window for the batch clustering tests, the total duration and the chosen time intervals of the windows for the dynamic clustering tests are both described in .

4.4. Quality of the distribution comparisons

We performed an experiment to confirm that the statistical test we use to identify similarity between identical distributions works correctly, as this is important for the proposed framework. First, we create pairs of clusters drawn from the same distribution. Each pair is constructed with different distribution types and dimensions. We then apply the similarity test to these pairs. As expected, the results show that the percentage of similar clusters discovered for each distribution is about 95%, in accordance with the chosen threshold $\delta = 5\%$ (see Fig. 13).

We also evaluated the robustness of the similarity detection by shifting the position of a distribution by a fraction of its standard deviation σ over a random axis, to check at what level the two clusters are treated as not similar. therefore, we fix a cluster distribution and shift the position of the cluster by $1+\sigma$*std_per. After each shift, we compute the similarity between the two cluster distributions (Fig. 14).

For datasets with different dimensions (see Fig. 15), as long as the shift remains below about 10% of the standard deviation, we observe only a slow decrease in the proportion of pairs detected as similar, from 95% to 80%. If we further increase the shift, we observe, as expected, a sharp decrease in the proportion of pairs detected as similar.

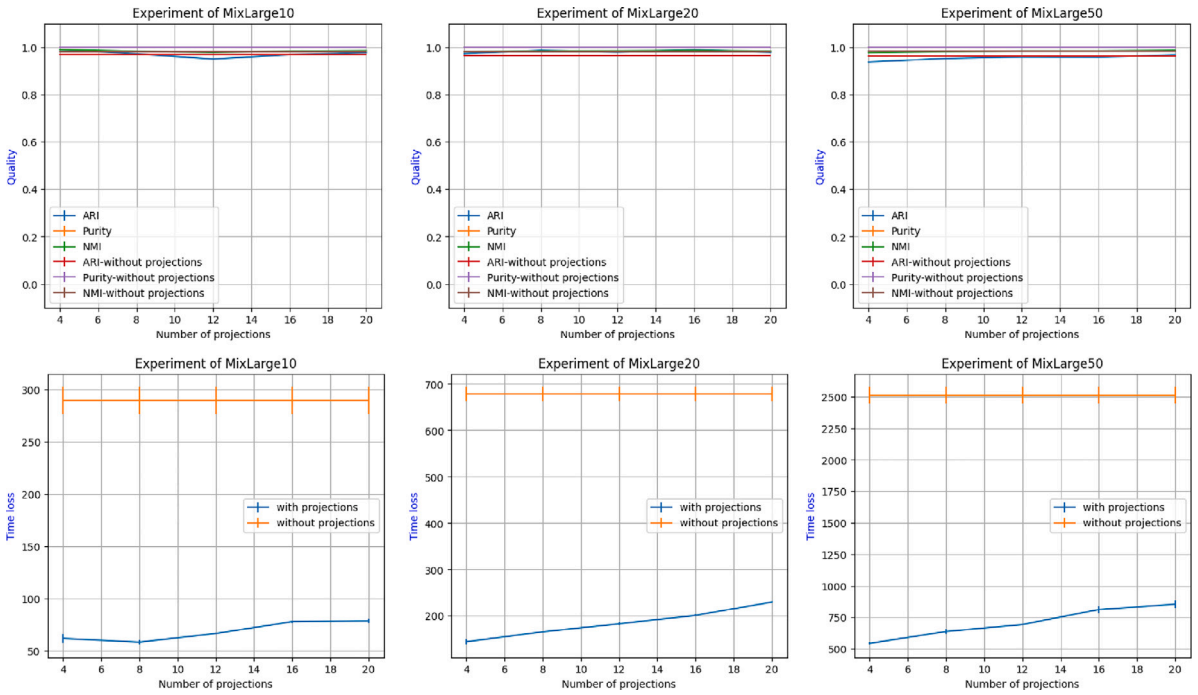

Fig. 7. Results with increasing the number of projections — Artificial datasets.

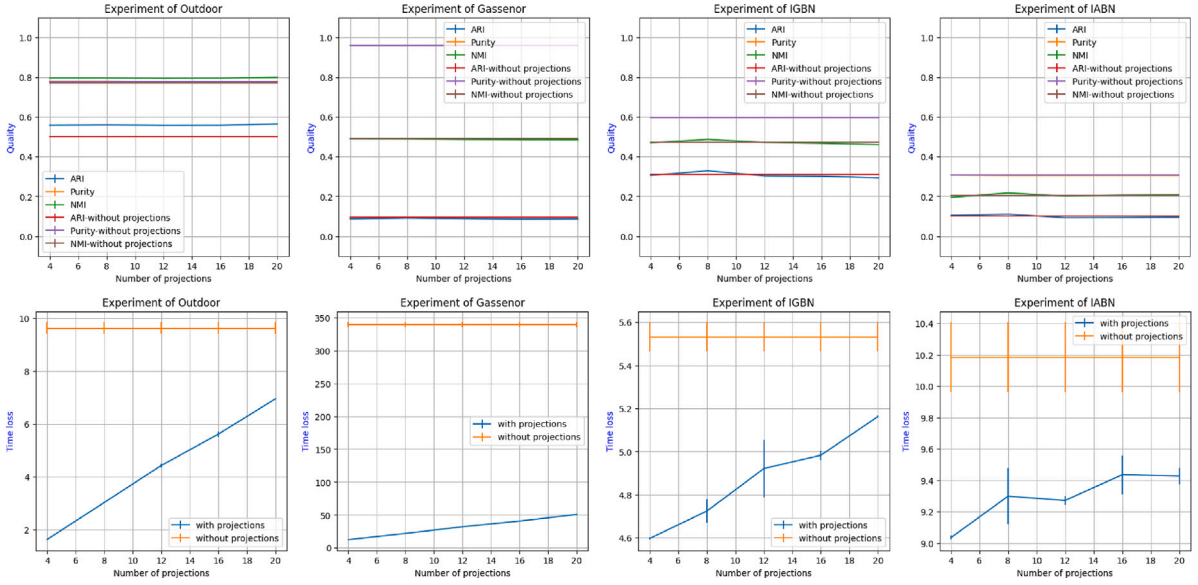

Fig. 8. Results with increasing the number of projections — Real datasets (PART I).

Table 3
Number of samples in the windows for each dataset.

Number of samples in the windows for each dataset					
Datasets	Nb_samples	Datasets	Nb_samples	Datasets	Nb_samples
Comet3	3000	Comet20	10 000	MixSmall1	1000
Meteorite3	3000	Meteorite20	10 000	MixSmall2	2000
Square3	3000	Square20	10 000	MixSmall3	3000
Gaussian3	3000	Gaussian20	10 000	MixLarge10	5000
Moon3	3000	Moon20	10 000	MixLarge20	10 000
Circle3	3000	Circle20	10 000	MixLarge30	20 000
Outdoor	1500	Gassenor	2000	IGBN	3000
IABN	6000	Rialto	8000	IIAIN	8000
IIRIN	40 000	Covtype	60 000		

Total duration and time interval of the windows for each dataset					
Datasets	Total_dur	Time_inter	Datasets	Total_dur	Time_inter
Comet20	200	4	Meteorite20	200	4
Square20	200	4	Gaussian20	200	4
Moon20	200	4	Circle20	200	4
MixLarge10	100	5	MixLarge20	200	6
MixLarge50	500	7	Outdoor	30	6
Gassenor	50	5	IGBN	50	5
IABN	80	4	Rialto	80	4
IIAIN	150	5	IIRIN	150	5
Covtype	150	5			

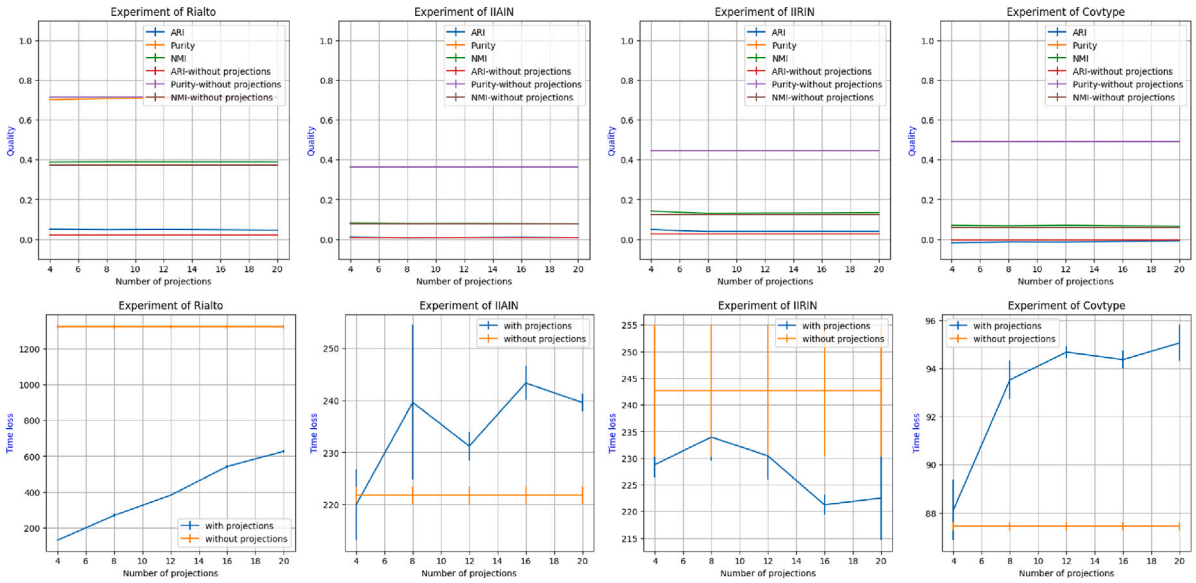

Fig. 9. Results with increasing the number of projections — Real datasets (PART II).

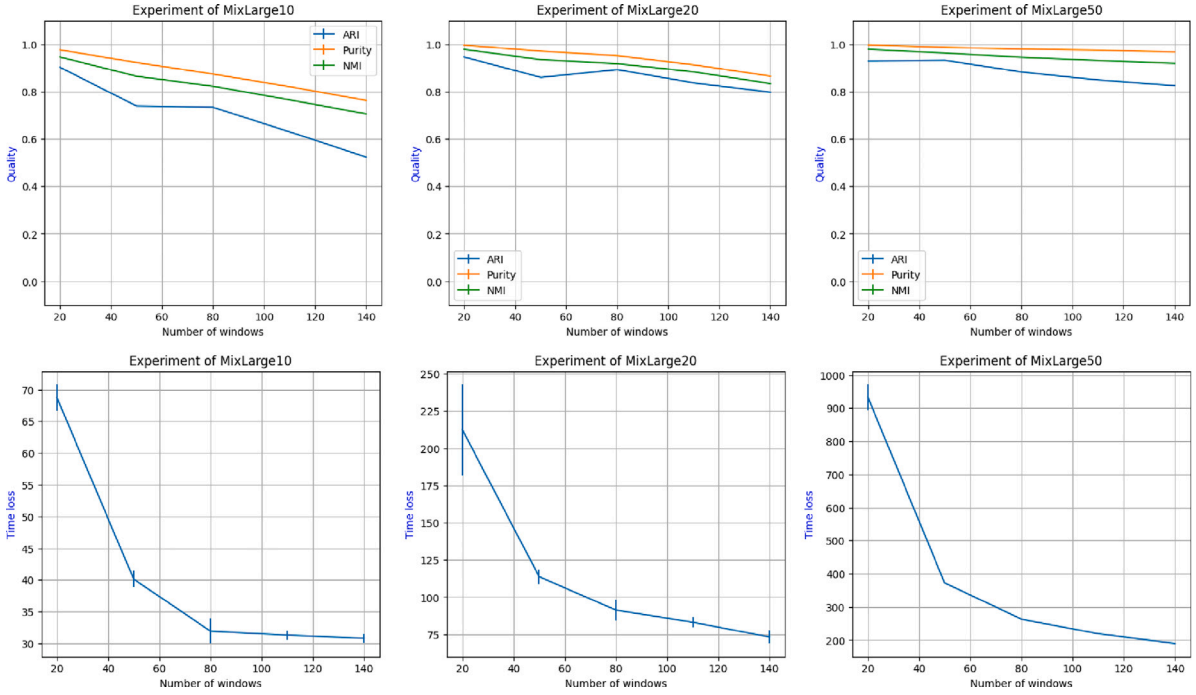

Fig. 10. Results with increasing the number of windows — Artificial datasets.

4.5. Assessment of algorithm quality and complexity

We performed a series of experiments to evaluate the quality and complexity of the resulting clustering in comparison to existing approaches (Fig. 16). The proposed framework can be used with any clustering algorithm, depending on the required cluster properties, by working on subsets (windows) of data and comparing the obtained clusters between windows. For experiments on static datasets, we separated the artificial datasets into small and large categories to apply two sets of appropriate clustering algorithms, while applying all algorithms to the real datasets. We expect the proposed framework to reduce the computational speed and memory requirements of most existing clustering algorithms, while preserving their interesting properties, with minimal loss of quality. For experiments on dynamic datasets, we applied the algorithms on large artificial datasets and all the real datasets. We expect the proposed framework to be competitive with existing stream clustering approaches in terms of quality and complexity, while allowing great flexibility in the choice of the clustering algorithm to be applied. For all experiments, the mean and standard deviation over five replicates are presented as results.

4.5.1. Experiments on static datasets

As mentioned above, our approach relies on sliding windows and the basic idea is to first perform conventional clustering on each window, then represent the resulting clusters in a histogram model and merge the windows to obtain the final clustering results. This part of the test compares conventional clustering algorithms applied to the full dataset with the proposed approach using the same clustering

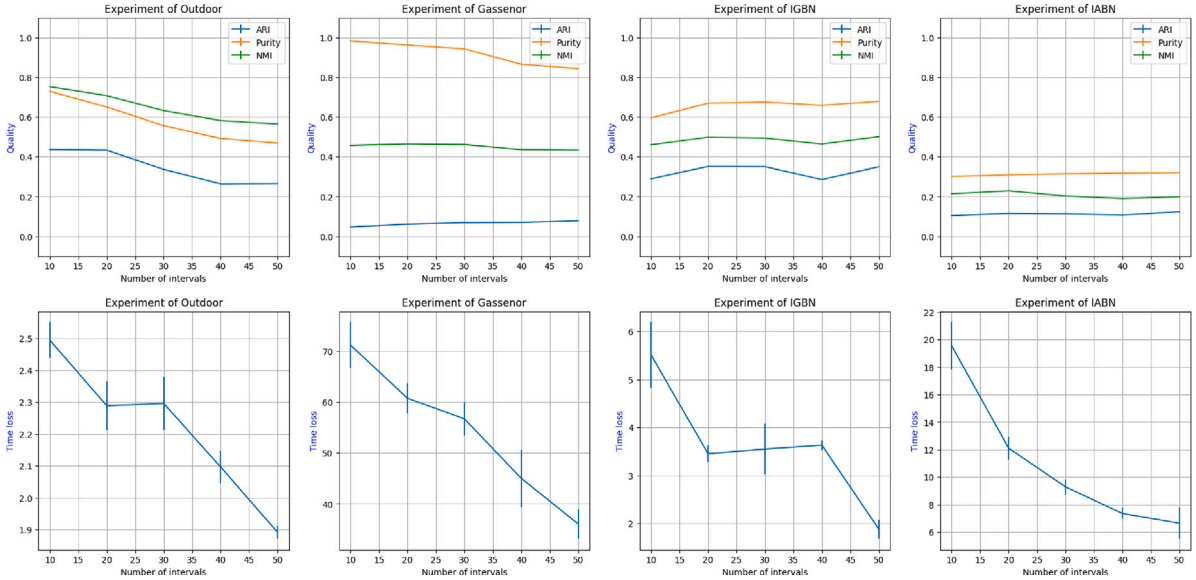

Fig. 11. Results with increasing the number of windows — Real datasets (PART I).

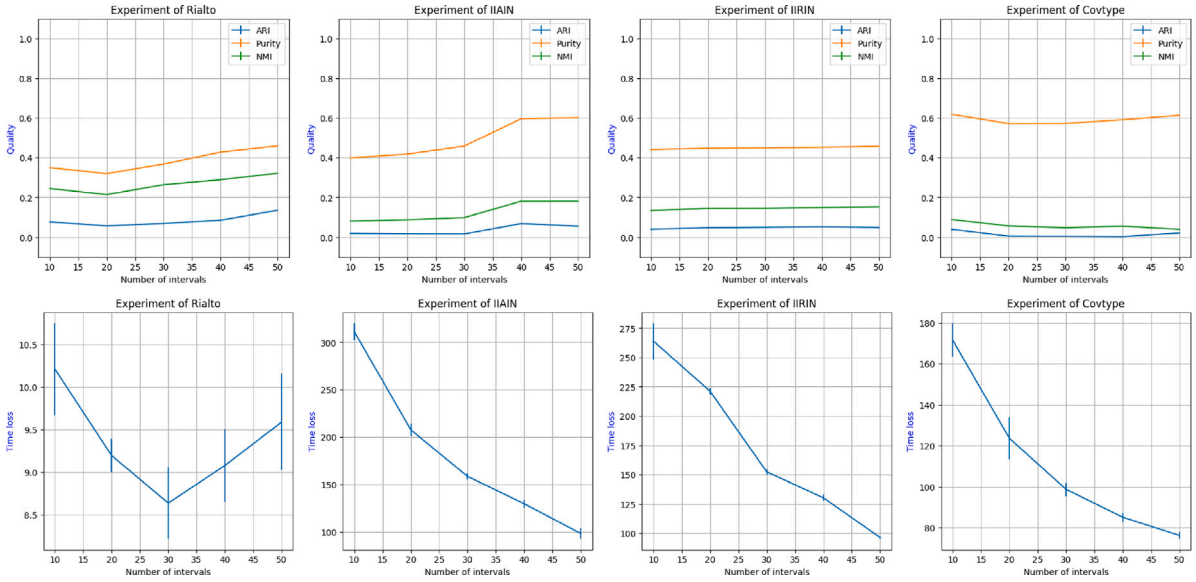

Fig. 12. Results with increasing the number of windows — Real datasets (PART II).

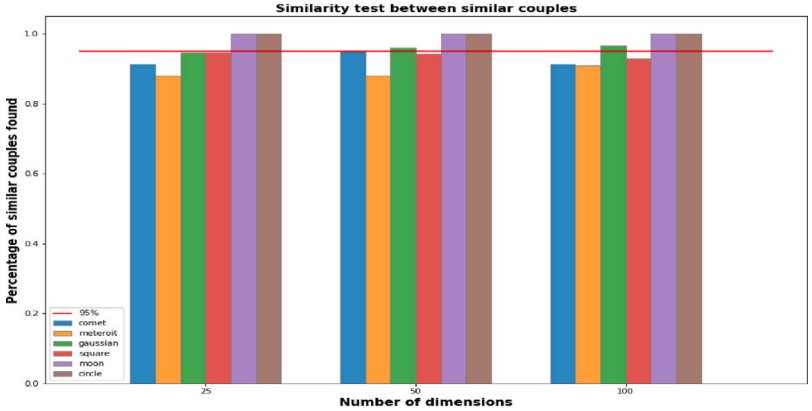

Fig. 13. Similarity tests between similar distribution. The red line represent 95% of correct detection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

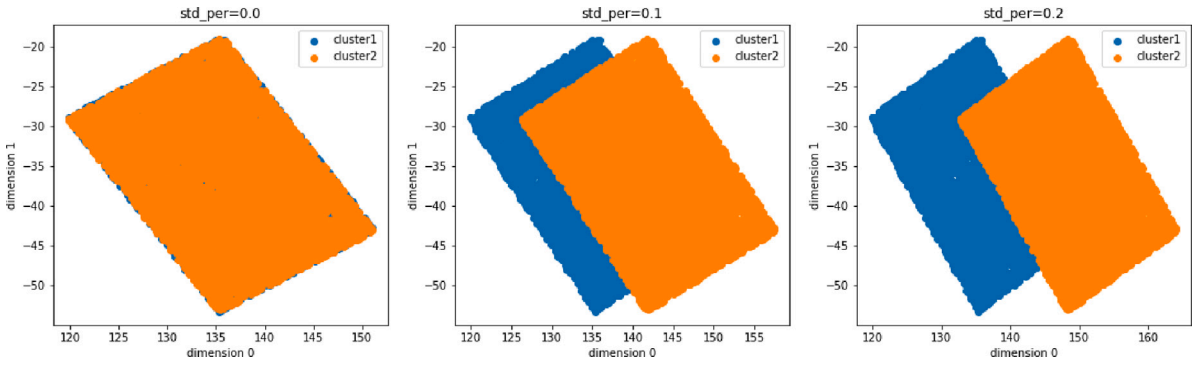

Fig. 14. Example of a similarity test between two similar clusters by changing the standard deviation percentage (std_per) to change the position of the second cluster.

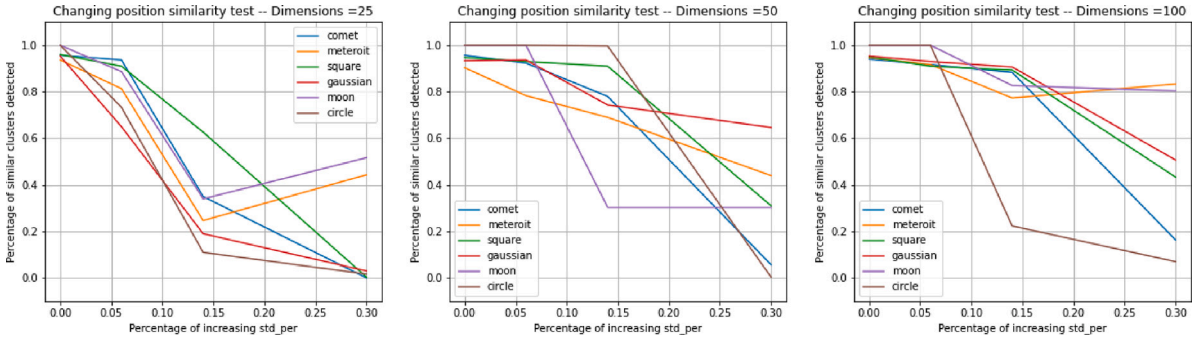

Fig. 15. Similarity test on distinct distributions of different dimension numbers by increasing std_per.

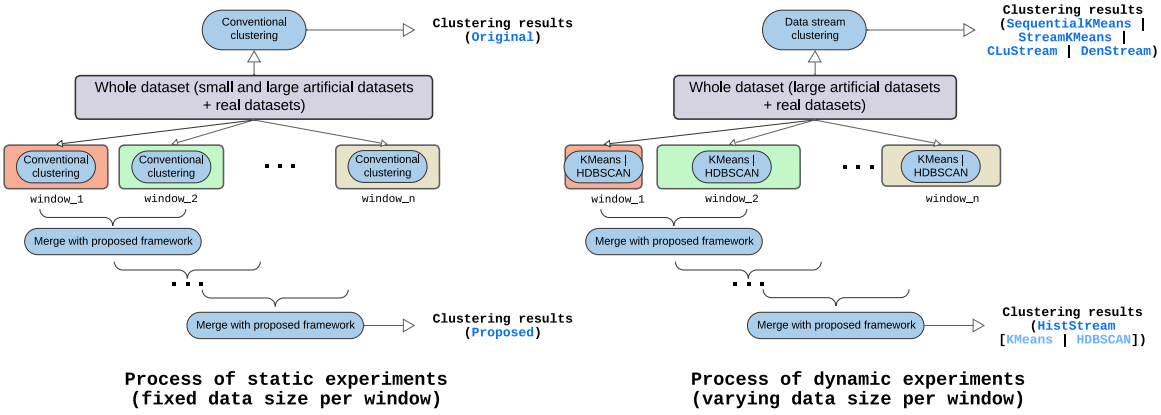

Fig. 16. Overview of the experimental protocol for static and dynamic datasets.

algorithms applied to sliding windows (illustrated in Fig. 16). The following clustering algorithms were selected for comparison in this section: BIRCH, Agglomerative Clustering, OPTICS, Spectral Clustering, K-means, Gaussian Mixture, HDBSCAN, and KASP. Except for spectral clustering and KASP, other algorithms were implemented in the scikit-learn package [42]. For algorithms that require a predefined number of clusters, we set it based on true labels. For spectral clustering, we used the STAGPy library [47], a C++ library with a Python wrapper, and set the neighborhood parameter to 10. Due to high complexity, BIRCH, Agglomerative, and Spectral clustering were only tested on small datasets. KASP used scikit-learn's K-means and STAGPy's spectral clustering with subsample size set to $5 * \sqrt{Nb\_samples}$ and neighborhoods set to 10. K-means, Gaussian Mixture, and KASP are optimized for large datasets, while OPTICS has only been tested on small datasets due to slower performance compared to HDBSCAN on larger datasets.

For small datasets, the ARI, Purity, NMI, and computation time results are detailed in Tables 4, 5, 6, and 7. With the means of each index separated between artificial and real datasets, show that for artificial datasets, the proposed approaches perform similarly to BIRCH and Agglomerative Clustering, with only slight differences. Compared to OPTICS and Spectral clustering, the proposed approaches show superior quality. For real datasets, the proposed approaches generally outperform conventional clustering methods.

In addition, the proposed approaches consistently exhibit faster computation time than the original algorithms. The Trade-off analysis between clustering quality and computation time, shown in Figs. 17 and 18, further confirms the efficiency of the proposed framework.

For large datasets, the ARI, Purity, NMI and computation time results are presented in Tables 8, 9, 10, and 11. From the results of artificial datasets, the proposed approaches are comparable to K-means and GaussianMixture and superior to HDBSCAN. The difference with KASP is more noticeable for the ARI index, but the proposed framework is superior for NMI and Purity. It also always outperforms the conventional algorithms on real datasets.

Table 4
ARI results of static experiments — Small datasets.

Datasets	BIRCH				Agglomerative				OPTICS				SpectralClustering			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet3	0.930	<1e−4	0.959	<1e−4	0.932	<1e−4	0.958	<1e−4	0.337	<1e−4	0.745	<1e−4	0.305	<1e−4	0.182	<1e−4
Meteroit3	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.999	<1e−4	1.000	<1e−4	0.836	<1e−4	0.598	<1e−4	0.431	<1e−4
Circle3	0.981	<1e−4	0.963	<1e−4	0.981	<1e−4	0.966	<1e−4	0.025	<1e−4	0.945	<1e−4	0.186	<1e−4	0.294	<1e−4
Gaussian3	1.000	<1e−4	0.896	<1e−4	1.000	<1e−4	0.709	<1e−4	1.000	<1e−4	0.999	<1e−4	0.098	<1e−4	0.319	<1e−4
Square3	1.000	<1e−4	0.978	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.993	<1e−4	0.326	<1e−4	0.431	<1e−4
Moon3	1.000	<1e−4	0.903	<1e−4	1.000	<1e−4	0.904	<1e−4	0.865	<1e−4	0.978	<1e−4	0.264	<1e−4	0.168	<1e−4
MixSmall1	0.982	<1e−4	0.955	<1e−4	0.979	<1e−4	0.956	<1e−4	0.696	<1e−4	0.953	<1e−4	0.298	<1e−4	0.356	<1e−4
MixSmall2	0.927	<1e−4	0.879	<1e−4	0.931	<1e−4	0.881	<1e−4	0.988	<1e−4	0.955	<1e−4	0.407	<1e−4	0.412	<1e−4
MixSmall3	0.994	<1e−4	0.984	<1e−4	0.994	<1e−4	0.984	<1e−4	0.996	<1e−4	0.984	<1e−4	0.208	<1e−4	0.443	<1e−4
Average mean(Artificial)	0.979	<1e−4	0.946	<1e−4	0.980	<1e−4	0.929	<1e−4	0.767	<1e−4	0.932	<1e−4	0.299	<1e−4	0.337	<1e−4
Outdoor	0.003	<1e−4	0.425	<1e−4	0.404	<1e−4	0.445	<1e−4	0.248	<1e−4	0.265	<1e−4	0.404	<1e−4	0.354	<1e−4
Gassenor	0.127	<1e−4	0.275	<1e−4	0.127	<1e−4	0.284	<1e−4	0.013	<1e−4	0.022	<1e−4	0.240	<1e−4	0.312	<1e−4
IGBN	0.009	<1e−4	0.286	<1e−4	0.065	<1e−4	0.227	<1e−4	0.006	<1e−4	0.013	<1e−4	0.137	<1e−4	0.201	<1e−4
IABN	0.111	<1e−4	0.100	<1e−4	0.110	<1e−4	0.062	<1e−4	0.003	<1e−4	0.007	<1e−4	0.122	<1e−4	0.078	<1e−4
Average mean(Real)	0.062	<1e−4	0.272	<1e−4	0.177	<1e−4	0.255	<1e−4	0.068	<1e−4	0.077	<1e−4	0.226	<1e−4	0.236	<1e−4
Total average mean	0.697	<1e−4	0.739	<1e−4	0.733	<1e−4	0.721	<1e−4	0.552	<1e−4	0.669	<1e−4	0.276	<1e−4	0.306	<1e−4

Table 5
Purity results for static experiments — Small datasets.

Datasets	BIRCH				Agglomerative				OPTICS				SpectralClustering			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet3	0.999	<1e−4	1.000	<1e−4	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.676	<1e−4	0.825	<1e−4
Meteroit3	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.757	<1e−4	0.913	<1e−4
Circle3	0.998	<1e−4	0.999	<1e−4	0.998	<1e−4	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	0.556	<1e−4	0.855	<1e−4
Gaussian3	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.449	<1e−4	0.904	<1e−4
Square3	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.573	<1e−4	0.838	<1e−4
Moon3	1.000	<1e−4	0.998	<1e−4	1.000	<1e−4	0.998	<1e−4	1.000	<1e−4	1.000	<1e−4	0.621	<1e−4	0.798	<1e−4
MixSmall1	0.999	<1e−4	0.999	<1e−4	0.999	<1e−4	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	0.669	<1e−4	0.896	<1e−4
MixSmall2	0.998	<1e−4	0.999	<1e−4	0.998	<1e−4	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	0.686	<1e−4	0.895	<1e−4
MixSmall3	0.999	<1e−4	1.000	<1e−4	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.608	<1e−4	0.915	<1e−4
Average mean(Artificial)	0.999	<1e−4	0.999	<1e−4	0.999	<1e−4	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	0.622	<1e−4	0.871	<1e−4
Outdoor	0.057	<1e−4	0.617	<1e−4	0.564	<1e−4	0.656	<1e−4	0.955	<1e−4	0.931	<1e−4	0.559	<1e−4	0.629	<1e−4
Gassenor	0.425	<1e−4	0.680	<1e−4	0.425	<1e−4	0.680	<1e−4	0.993	<1e−4	0.987	<1e−4	0.500	<1e−4	0.714	<1e−4
IGBN	0.235	<1e−4	0.557	<1e−4	0.320	<1e−4	0.632	<1e−4	0.659	<1e−4	0.697	<1e−4	0.375	<1e−4	0.648	<1e−4
IABN	0.308	<1e−4	0.374	<1e−4	0.349	<1e−4	0.414	<1e−4	0.670	<1e−4	0.647	<1e−4	0.325	<1e−4	0.428	<1e−4
Average mean(Real)	0.256	<1e−4	0.557	<1e−4	0.414	<1e−4	0.596	<1e−4	0.819	<1e−4	0.816	<1e−4	0.440	<1e−4	0.605	<1e−4
Total average mean	0.771	<1e−4	0.863	<1e−4	0.819	<1e−4	0.875	<1e−4	0.944	<1e−4	0.943	<1e−4	0.566	<1e−4	0.789	<1e−4

Table 6
NMI results for static experiments — Small datasets.

Datasets	BIRCH				Agglomerative				OPTICS				SpectralClustering			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet3	0.973	<1e−4	0.965	<1e−4	0.974	<1e−4	0.964	<1e−4	0.765	<1e−4	0.803	<1e−4	0.612	<1e−4	0.601	<1e−4
Meteroit3	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.997	<1e−4	1.000	<1e−4	0.948	<1e−4	0.756	<1e−4	0.708	<1e−4
Circle3	0.986	<1e−4	0.958	<1e−4	0.986	<1e−4	0.963	<1e−4	0.386	<1e−4	0.815	<1e−4	0.529	<1e−4	0.661	<1e−4
Gaussian3	1.000	<1e−4	0.955	<1e−4	1.000	<1e−4	0.911	<1e−4	1.000	<1e−4	0.999	<1e−4	0.431	<1e−4	0.702	<1e−4
Square3	1.000	<1e−4	0.990	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.989	<1e−4	0.687	<1e−4	0.728	<1e−4
Moon3	1.000	<1e−4	0.923	<1e−4	1.000	<1e−4	0.924	<1e−4	0.707	<1e−4	0.938	<1e−4	0.600	<1e−4	0.587	<1e−4
MixSmall1	0.991	<1e−4	0.973	<1e−4	0.990	<1e−4	0.973	<1e−4	0.813	<1e−4	0.933	<1e−4	0.726	<1e−4	0.751	<1e−4
MixSmall2	0.982	<1e−4	0.955	<1e−4	0.982	<1e−4	0.957	<1e−4	0.959	<1e−4	0.945	<1e−4	0.788	<1e−4	0.780	<1e−4
MixSmall3	0.996	<1e−4	0.987	<1e−4	0.996	<1e−4	0.987	<1e−4	0.981	<1e−4	0.961	<1e−4	0.678	<1e−4	0.802	<1e−4
Average mean(Artificial)	0.992	<1e−4	0.967	<1e−4	0.992	<1e−4	0.964	<1e−4	0.846	<1e−4	0.926	<1e−4	0.645	<1e−4	0.702	<1e−4
Outdoor	0.068	<1e−4	0.731	<1e−4	0.715	<1e−4	0.737	<1e−4	0.779	<1e−4	0.774	<1e−4	0.694	<1e−4	0.699	<1e−4
Gassenor	0.275	<1e−4	0.471	<1e−4	0.275	<1e−4	0.475	<1e−4	0.419	<1e−4	0.438	<1e−4	0.424	<1e−4	0.510	<1e−4
IGBN	0.087	<1e−4	0.431	<1e−4	0.124	<1e−4	0.412	<1e−4	0.257	<1e−4	0.297	<1e−4	0.197	<1e−4	0.409	<1e−4
IABN	0.138	<1e−4	0.216	<1e−4	0.137	<1e−4	0.191	<1e−4	0.245	<1e−4	0.242	<1e−4	0.173	<1e−4	0.216	<1e−4
Average mean(Real)	0.142	<1e−4	0.462	<1e−4	0.313	<1e−4	0.454	<1e−4	0.425	<1e−4	0.438	<1e−4	0.372	<1e−4	0.459	<1e−4
Total average mean	0.730	<1e−4	0.812	<1e−4	0.783	<1e−4	0.807	<1e−4	0.716	<1e−4	0.776	<1e−4	0.561	<1e−4	0.627	<1e−4

In order to analyze the potential limitations of the proposed approach in more detail, it is important to set out the prerequisites for a high quality result. First, since the framework decomposes the dataset into smaller subsets, it is necessary that the clustering algorithm applied to each subset maintains good performance (in terms of computation time and result quality) compared to the same algorithm applied to the entire dataset. Second, since the distribution of the data in each cluster is estimated according to a set of univariate distributions based on random projections, this estimation must be reliable, which may not be the case for some specific non-convex distributions.

The first case is illustrated in our experimental result with the slight decrease in quality observed when using KASP in the proposed framework. KASP operates on a subsample of the original dataset distribution generated using K-means prototypes, followed by applying spectral clustering to this subsample to obtain preliminary partitions that guide the final clustering process. The size of the subsample has a significant impact on the clustering quality, and depends on the number of data points, leading to different performances on the whole dataset compared to its decomposition into smaller subsets. Indeed, clustering quality in KASP shows considerable variability as the number of samples changes, and the relationship between performance and

Table 7
Computation time for static experiments — Small datasets.

Datasets	BIRCH				Agglomerative				OPTICS				SpectralClustering			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet3	46.399	5.525	7.841	0.243	127.408	58.311	5.026	0.387	493.379	29.105	163.468	4.287	12.357	1.412	8.769	1.257
Meteroit3	67.044	7.569	8.112	0.412	98.868	3.322	4.885	0.161	918.683	38.070	150.940	2.901	13.534	2.371	8.857	1.275
Circle3	58.836	14.692	7.689	0.115	385.324	380.004	4.635	0.046	549.950	93.681	200.485	9.385	11.447	1.409	6.992	0.041
Gaussian3	50.604	6.429	7.934	0.260	149.074	54.408	4.911	0.121	907.935	7.840	156.065	15.702	8.010	0.036	6.942	0.016
Square3	48.930	13.221	10.005	0.339	269.327	26.808	5.556	0.319	517.414	57.988	183.234	5.660	8.706	0.077	6.885	0.056
Moon3	50.688	8.058	8.957	0.777	116.288	12.716	5.423	0.418	616.353	81.158	177.262	2.302	9.945	0.044	7.074	0.038
MixSmall1	10.356	1.137	2.038	0.042	11.855	0.824	1.266	0.021	227.003	9.311	31.719	0.944	2.887	0.004	2.572	0.011
MixSmall2	54.298	11.138	6.528	0.140	46.100	1.544	4.892	0.646	544.662	77.890	139.102	4.347	9.112	0.070	5.603	0.046
MixSmall3	90.402	9.046	11.934	0.188	173.702	75.256	8.875	0.303	762.134	181.823	291.226	7.313	13.905	0.043	8.832	0.034
Average mean(Artificial)	53.062	8.535	7.893	0.280	153.105	68.133	5.052	0.269	615.279	64.096	165.945	5.871	9.989	0.607	6.947	0.308
Outdoor	0.062	<1e−4	1.026	0.030	0.312	<1e−4	0.821	0.009	4.584	1.552	10.977	0.176	0.972	0.015	1.085	0.006
Gassenor	22.957	2.199	3.475	0.046	19.527	1.609	2.107	0.046	335.529	21.115	225.419	8.425	3.443	0.045	3.243	0.158
IGBN	0.743	0.006	0.685	0.037	34.408	3.658	3.480	0.150	422.940	131.530	161.536	2.347	9.125	0.925	8.629	1.812
IABN	0.921	0.001	1.327	0.007	48 190.256	3384.368	15.942	0.300	974.319	0.373	687.564	21.508	21.413	2.743	17.525	3.143
Average mean(Real)	6.171	0.551	1.628	0.030	12 061.126	847.409	5.588	0.126	434.343	38.642	271.374	8.114	8.738	0.932	7.620	1.280
Total average mean	38.634	6.079	5.965	0.203	3817.111	307.910	5.217	0.225	559.607	56.264	198.384	6.561	9.604	0.707	7.154	0.607

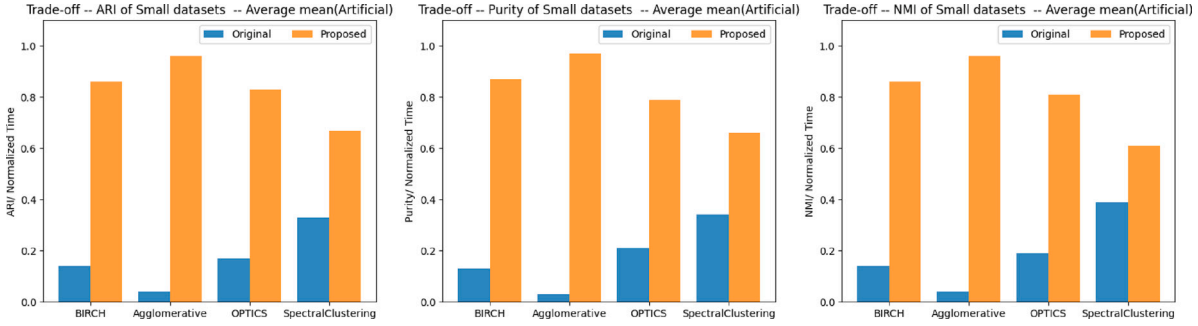

Fig. 17. Trade-off – Mean values of ARI (left), Purity (center) and NMI (right) for static experiments — Small datasets (Artificial).

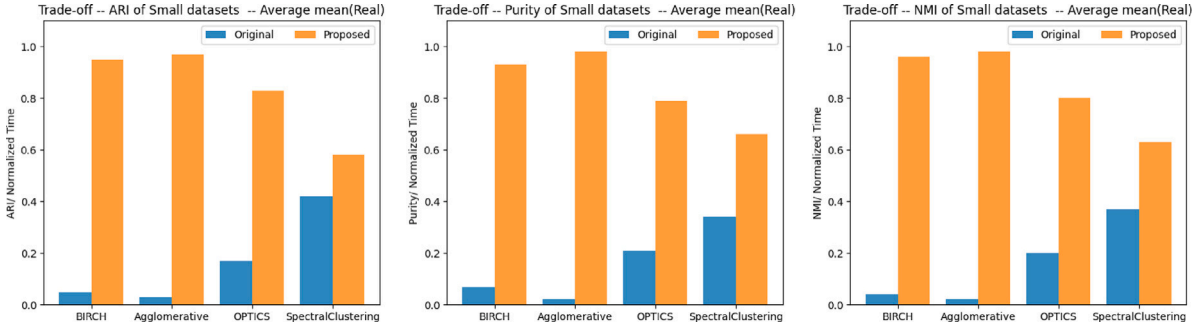

Fig. 18. Trade-off – Mean values of ARI (left), Purity (center) and NMI (right) for static experiments — Small datasets (Real).

dataset size is non-linear. This contrasts with other algorithms, such as K-means, where performance is more stable. As a result, the proposed framework may not fully benefit from the decomposition when using KASP or similar algorithms, as their instability on smaller subsets may lead to less accurate clustering despite a better computation time.

The second case is visible in the slightly suboptimal results obtained on the Circle20 dataset. Its artificial clusters have the shape of circles, a highly non-convex shape with a hole in the center. The cluster distributions estimated from univariate distribution models based on random projections do not correctly capture the real distribution. In particular, the central hole can never be modeled from a univariate projection. This leads to a loss of structural information present in the original distribution. Since the clustering process is based on these projections, any bias introduced by the projection can negatively affect the efficiency of the final clustering process. This is obviously an extreme case, as most distributions can be reliably modeled using random projections.

The trade-off results in Figs. 19 and 20 confirm the efficiency of the proposed framework (except, again, for KASP according to the ARI index, while the trade-off is better according to the Purity and NMI). Overall, the proposed framework significantly accelerates the clustering process for both real and artificial datasets without compromising quality, proving to be effective across different clustering algorithms and offering great flexibility on massive datasets.

4.5.2. *Experiments on dynamic datasets*

To evaluate our approach on dynamic datasets, we compared it with traditional methods such as CluStream and DenStream (discussed in Section 1) and two methods from the Python package River [48]: SequentialK-means (a mini-batch K-means variant with a parameter *halflife* that shifts cluster centers toward more recent observations) and StreamK-means (a STREAMLSEARCH variant that uses incremental K-means for efficiency). CluStream and DenStream were implemented based on their respective papers. For our approaches, we chose HDBSCAN and K-means as conventional clustering in each window, and named it HistStream(HDBSCAN) and HistStream(K-means)

Table 8
ARI results of static experiments — Large datasets.

Datasets	KMeans				GaussianMixture				HDBSCAN				KASP			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	1.000	<1e−4	0.931	<1e−4	0.979	0.015	0.932	0.014	0.300	<1e−4	0.936	<1e−4	0.541	0.071	0.206	0.011
Meteroit20	1.000	<1e−4	0.987	<1e−4	0.998	0.002	0.963	0.017	0.741	<1e−4	0.948	<1e−4	0.477	0.119	0.324	0.011
Circle20	1.000	<1e−4	0.765	0.001	0.911	<1e−4	0.779	0.012	0.140	<1e−4	0.906	<1e−4	0.510	0.126	0.176	0.012
Gaussian20	1.000	<1e−4	0.974	<1e−4	1.000	<1e−4	0.978	0.004	0.909	<1e−4	0.982	<1e−4	0.625	0.042	0.277	0.017
Square20	1.000	<1e−4	0.984	<1e−4	1.000	<1e−4	0.945	0.011	0.634	<1e−4	0.963	<1e−4	0.631	0.059	0.348	0.052
Moon20	1.000	<1e−4	0.975	<1e−4	1.000	<1e−4	0.972	0.005	0.269	<1e−4	0.964	<1e−4	0.304	0.038	0.314	0.013
MixLarge10	1.000	<1e−4	0.976	0.001	0.993	0.005	0.932	0.013	0.466	<1e−4	0.975	<1e−4	0.548	0.087	0.317	0.017
MixLarge20	0.936	0.002	0.883	0.005	0.940	0.003	0.864	0.010	0.187	<1e−4	0.979	<1e−4	0.535	0.031	0.468	0.010
MixLarge50	0.948	<1e−4	0.855	0.004	0.926	0.009	0.792	0.004	0.177	<1e−4	0.948	<1e−4	0.420	0.086	0.336	0.013
Average mean(Artificial)	0.987	0.000	0.926	0.001	0.972	0.004	0.906	0.010	0.425	<1e−4	0.956	<1e−4	0.510	0.073	0.307	0.017
Outdoor	0.411	0.008	0.454	0.012	0.432	<1e−4	0.454	0.009	<1e−4	<1e−4	0.563	<1e−4	0.300	0.019	0.360	0.016
Gassenor	0.138	<1e−4	0.278	0.008	0.123	<1e−4	0.306	0.011	0.383	<1e−4	0.044	<1e−4	0.174	0.013	0.282	0.011
IGBN	0.080	<1e−4	0.217	0.009	0.166	<1e−4	0.249	0.007	<1e−4	<1e−4	0.306	<1e−4	0.086	0.012	0.200	0.011
IABN	0.100	0.018	0.078	0.001	0.262	<1e−4	0.113	0.014	<1e−4	<1e−4	0.116	<1e−4	0.091	0.003	0.090	0.012
Rialto	0.068	<1e−4	0.063	0.002	0.058	<1e−4	0.048	0.001	0.056	<1e−4	0.081	<1e−4	0.055	0.001	0.059	0.002
IIAIN	0.048	<1e−4	0.021	0.001	0.299	<1e−4	0.032	0.002	<1e−4	<1e−4	0.021	<1e−4	0.064	0.003	0.019	0.001
IIRIN	0.049	<1e−4	0.037	0.001	0.178	<1e−4	0.057	0.001	<1e−4	<1e−4	0.073	<1e−4	0.063	0.001	0.035	0.001
Covtype	0.014	<1e−4	0.015	<1e−4	0.020	<1e−4	0.015	<1e−4	<1e−4	<1e−4	0.023	<1e−4	0.016	0.004	0.010	0.001
Average mean(Real)	0.113	0.003	0.145	0.004	0.192	<1e−4	0.159	0.006	0.055	<1e−4	0.153	<1e−4	0.106	0.007	0.132	0.007
Total average mean	0.576	0.002	0.558	0.003	0.605	0.002	0.555	0.008	0.251	<1e−4	0.578	<1e−4	0.320	0.042	0.225	0.012

Table 9
Purity results of static experiments — Large datasets.

Datasets	KMeans				GaussianMixture				HDBSCAN				KASP			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.760	0.040	0.873	0.006
Meteroit20	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.618	0.060	0.748	0.004
Circle20	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.749	0.044	0.844	0.005
Gaussian20	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.705	0.029	0.761	0.010
Square20	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.706	0.028	0.788	0.010
Moon20	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.432	0.036	0.650	0.010
MixLarge10	1.000	<1e−4	1.000	<1e−4	0.999	0.001	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	0.705	0.052	0.775	0.014
MixLarge20	1.000	<1e−4	0.999	<1e−4	0.999	0.001	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	0.613	0.023	0.760	0.010
MixLarge50	1.000	<1e−4	1.000	<1e−4	0.999	<1e−4	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.682	0.043	0.705	0.011
Average mean(Artificial)	1.000	<1e−4	1.000	<1e−4	1.000	0.000	1.000	<1e−4	1.000	<1e−4	1.000	<1e−4	0.663	0.039	0.767	0.009
Outdoor	0.573	0.001	0.655	0.004	0.589	<1e−4	0.672	0.007	0.053	<1e−4	0.877	<1e−4	0.432	0.011	0.539	0.018
Gassenor	0.430	<1e−4	0.668	0.001	0.394	<1e−4	0.727	0.003	0.835	<1e−4	0.987	<1e−4	0.458	0.002	0.688	0.002
IGBN	0.326	<1e−4	0.611	0.005	0.409	<1e−4	0.688	0.009	0.284	<1e−4	0.554	<1e−4	0.332	0.016	0.611	0.026
IABN	0.335	0.016	0.458	0.001	0.455	<1e−4	0.544	0.008	0.230	<1e−4	0.300	<1e−4	0.331	0.002	0.442	0.003
Rialto	0.224	<1e−4	0.331	<1e−4	0.205	<1e−4	0.295	0.002	0.824	<1e−4	0.335	<1e−4	0.217	0.004	0.318	0.004
IIAIN	0.382	<1e−4	0.439	0.001	0.660	<1e−4	0.563	0.013	0.298	<1e−4	0.402	<1e−4	0.423	0.005	0.445	0.003
IIRIN	0.382	<1e−4	0.499	0.002	0.530	<1e−4	0.619	0.009	0.298	<1e−4	0.524	<1e−4	0.420	0.004	0.518	0.002
Covtype	0.484	<1e−4	0.631	<1e−4	0.480	<1e−4	0.617	<1e−4	0.474	<1e−4	0.555	<1e−4	0.489	0.004	0.535	0.002
Average mean(Real)	0.392	0.002	0.536	0.002	0.465	<1e−4	0.591	0.006	0.412	<1e−4	0.567	<1e−4	0.388	0.006	0.512	0.007
Total average mean	0.714	0.001	0.782	0.001	0.748	0.000	0.807	0.003	0.723	<1e−4	0.796	<1e−4	0.534	0.024	0.647	0.008

Table 10
NMI results of static experiments — Large datasets.

Datasets	KMeans				GaussianMixture				HDBSCAN				KASP			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	1.000	<1e−4	0.975	<1e−4	0.995	0.003	0.977	0.004	0.730	<1e−4	0.979	<1e−4	0.847	0.016	0.723	0.005
Meteroit20	1.000	<1e−4	0.992	<1e−4	0.999	0.001	0.981	0.005	0.863	<1e−4	0.981	<1e−4	0.806	0.044	0.711	0.003
Circle20	1.000	<1e−4	0.947	<1e−4	0.987	<1e−4	0.941	0.005	0.673	<1e−4	0.939	<1e−4	0.825	0.029	0.696	0.004
Gaussian20	1.000	<1e−4	0.989	<1e−4	1.000	<1e−4	0.989	0.002	0.885	<1e−4	0.991	<1e−4	0.864	0.011	0.714	0.006
Square20	1.000	<1e−4	0.990	0.001	1.000	<1e−4	0.981	0.004	0.767	<1e−4	0.987	<1e−4	0.863	0.016	0.715	0.006
Moon20	1.000	<1e−4	0.990	<1e−4	1.000	<1e−4	0.986	0.001	0.732	<1e−4	0.986	<1e−4	0.744	0.026	0.683	0.007
MixLarge10	1.000	<1e−4	0.984	0.001	0.997	0.002	0.967	0.005	0.808	<1e−4	0.984	<1e−4	0.834	0.035	0.711	0.008
MixLarge20	0.988	<1e−4	0.965	0.001	0.989	0.001	0.960	0.003	0.716	<1e−4	0.984	<1e−4	0.837	0.014	0.751	0.006
MixLarge50	0.994	<1e−4	0.964	0.001	0.989	0.002	0.944	0.002	0.718	<1e−4	0.978	<1e−4	0.851	0.022	0.714	0.003
Average mean(Artificial)	0.998	<1e−4	0.977	0.000	0.995	0.001	0.970	0.003	0.766	<1e−4	0.979	<1e−4	0.830	0.024	0.713	0.005
Outdoor	0.708	0.002	0.733	0.005	0.736	<1e−4	0.748	0.002	<1e−4	<1e−4	0.813	<1e−4	0.626	0.006	0.674	0.007
Gassenor	0.284	<1e−4	0.484	0.009	0.239	<1e−4	0.520	0.007	0.649	<1e−4	0.459	<1e−4	0.309	0.005	0.489	0.005
IGBN	0.127	0.001	0.397	0.007	0.275	<1e−4	0.460	0.004	<1e−4	<1e−4	0.457	<1e−4	0.135	0.008	0.395	0.015
IABN	0.132	0.008	0.217	<1e−4	0.335	<1e−4	0.287	0.004	<1e−4	<1e−4	0.216	<1e−4	0.124	0.002	0.212	0.002
Rialto	0.155	<1e−4	0.205	0.001	0.133	<1e−4	0.185	0.001	0.406	<1e−4	0.241	<1e−4	0.155	0.005	0.203	0.002
IIAIN	0.084	<1e−4	0.101	0.001	0.345	<1e−4	0.173	0.004	<1e−4	<1e−4	0.073	<1e−4	0.097	0.004	0.094	0.002
IIRIN	0.084	<1e−4	0.140	0.002	0.247	<1e−4	0.212	0.005	<1e−4	<1e−4	0.173	<1e−4	0.092	0.006	0.141	0.002
Covtype	0.063	<1e−4	0.141	<1e−4	0.033	<1e−4	0.134	<1e−4	<1e−4	<1e−4	0.100	<1e−4	0.060	0.002	0.069	0.001
Average mean(Real)	0.205	0.001	0.302	0.003	0.293	<1e−4	0.340	0.003	0.132	<1e−4	0.317	<1e−4	0.200	0.005	0.285	0.005
Total average mean	0.625	0.001	0.660	0.002	0.665	0.001	0.673	0.003	0.467	<1e−4	0.667	<1e−4	0.533	0.015	0.511	0.005

Table 11
Computation time of static experiments — Large datasets.

Datasets	KMeans				GaussianMixture				HDBSCAN				KASP			
	Original		Proposed		Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	23.734	2.791	19.337	1.458	222.994	78.158	65.818	7.758	4213.044	171.537	214.139	13.723	44.895	4.278	35.147	0.149
Meteroit20	24.148	1.002	18.958	0.441	132.950	34.525	85.222	13.100	4406.303	56.387	222.485	14.603	59.371	2.900	34.722	0.230
Circle20	16.433	1.387	20.670	0.554	113.119	2.273	104.328	13.320	3383.867	187.844	289.187	1.569	45.265	1.233	43.100	3.870
Gaussian20	23.563	0.489	14.661	0.249	115.247	1.359	58.952	5.740	3979.659	184.060	144.848	1.269	54.484	1.281	41.287	1.863
Square20	34.939	1.601	13.419	0.436	115.244	1.931	57.755	7.963	5379.374	519.370	153.297	2.101	47.494	0.984	39.315	2.496
Moon20	27.831	4.163	12.889	0.304	114.948	1.686	44.908	1.449	3994.732	237.748	142.009	2.100	41.096	0.809	32.762	0.108
MixLarge10	18.324	3.399	14.327	0.363	72.196	5.959	42.049	6.500	1205.131	74.827	49.496	0.889	18.909	0.328	23.478	0.196
MixLarge20	50.899	1.612	26.494	2.309	391.995	121.818	130.698	16.686	5014.705	408.986	191.287	4.018	56.267	3.025	39.175	0.879
MixLarge50	95.487	27.448	79.692	5.588	1076.031	390.764	812.565	88.755	18195.626	4904.393	787.725	55.214	245.408	19.075	112.379	1.000
Average mean(Artificial)	35.040	4.877	24.494	1.300	261.636	70.941	155.811	17.919	5530.271	749.461	243.830	10.610	68.132	3.768	44.596	1.199
Outdoor	0.302	0.007	1.035	0.049	0.766	0.013	1.928	0.174	0.266	0.013	2.995	0.087	0.340	0.007	0.714	0.015
Gassenor	0.406	0.066	0.557	0.013	19.528	1.907	4.792	0.605	16.792	0.332	71.868	1.691	2.168	0.276	2.001	0.040
IGBN	0.507	0.017	1.192	0.078	5.265	2.509	2.424	0.422	12.455	2.896	5.787	0.116	2.554	0.102	2.565	0.054
IABN	0.989	0.181	1.690	0.031	15.504	1.708	11.095	0.324	35.226	2.451	17.093	0.020	7.354	0.315	5.927	0.264
Rialto	2.737	0.374	3.271	0.034	14.258	1.003	14.315	1.294	12.315	0.132	10.889	0.287	11.539	0.900	12.881	0.284
IIAIN	11.121	0.706	15.691	1.426	262.002	2.598	220.794	25.076	2503.787	263.043	367.586	38.623	168.647	7.423	72.061	3.659
IIRIN	30.035	2.565	15.250	1.923	292.251	9.682	192.213	15.150	2871.159	13.816	353.679	11.625	175.862	9.507	66.223	1.020
Covtype	12.711	1.346	6.836	0.261	88.757	1.030	28.667	0.561	3187.865	138.144	102.875	3.611	177.251	14.071	79.992	2.320
Average mean(Real)	7.351	0.658	5.690	0.477	87.291	2.556	59.529	5.451	1079.983	52.603	116.596	7.008	68.214	4.075	30.296	0.957
Total average mean	22.010	2.891	15.645	0.913	179.591	38.760	110.501	12.052	3436.018	421.528	183.956	8.914	68.171	3.913	37.866	1.085

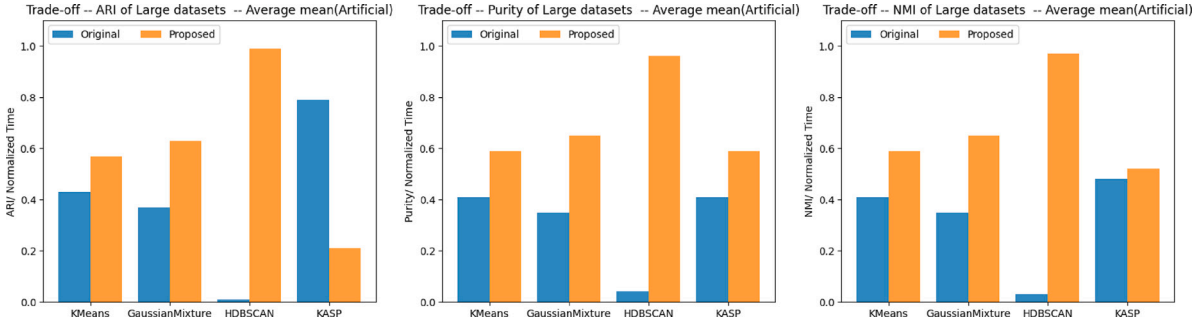

Fig. 19. Trade-off – Mean values of ARI (left), Purity (center) and NMI (right) for static experiments — Large datasets (Artificial).

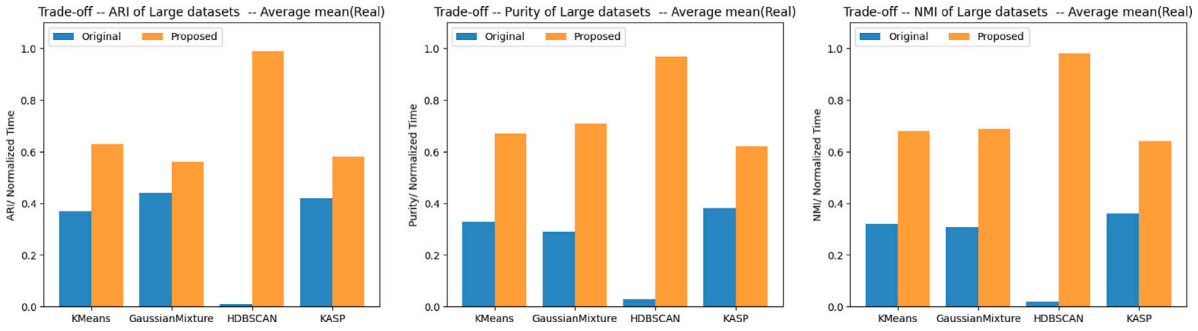

Fig. 20. Trade-off – Mean values of ARI (left), Purity (center) and NMI (right) for static experiments — Large datasets (Real).

for comparison with partition-based methods such as SequentialK-means, StreamK-means, and CluStream, and density-based clustering DenStream, respectively. The parameter settings were the same as in the static experiments, but the window size was defined by fixed time intervals instead of a fixed number of data points, which allows more flexibility in dynamic scenarios.

The results of ARI, NMI, Purity and Computation time are stored in Tables 12, 13, 14 and 15. Similar to the previous experimental section, to gain a more complete understanding of the overall performance differences between the algorithms, we computed the mean value of each index for each approach, separated into artificial datasets and real datasets. The results from both artificial and real datasets provide compelling evidence of the superiority of HistStream(K-means) over other algorithms in terms of clustering quality across multiple scenarios, consistently maintaining competitive performance. Similarly, HistStream(HDBSCAN) consistently achieves better clustering quality than DenStream. It is also faster than DenStream on high-dimensional datasets, which is particularly evident here with the artificial dataset, while DenStream appears to be faster than HistStream(HDBSCAN) on lower-dimensional datasets. Another interesting observation is that the stability of DenStream seems to be worse, especially when dealing with larger datasets, compared to the consistent performance of HistStream(HDBSCAN).

The results presented in terms of trade-off between clustering evaluation index and computation time (seen in Figs. 21 and 22) clearly highlight the better performance of the proposed approach over other algorithms in terms of clustering quality, without any significant compromise in computational efficiency. In particular, HistStream(K-means) exhibits significantly faster running time compared to both SequentialK-means and StreamK-means, while still being slightly faster than CluStream. Moreover, interestingly, the application of the proposed framework with HDBSCAN and K-means reveals noticeable differences. These differences highlight the distinct advantages of each

Table 12
ARI results of dynamic experiments.

Datasets	SequentialKmeans		StreamKmeans		CluStream		HistStream				DenStream	
							Kmeans		HDBSCAN			
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	0.7955	<1e−4	0.6770	0.0005	0.8516	0.0107	0.9713	<1e−4	0.9321	<1e−4	0.9537	<1e−4
Meteroit20	0.8112	<1e−4	0.6125	0.0034	0.8713	0.0021	0.9559	0.0002	0.9330	<1e−4	0.8531	<1e−4
Circle20	0.8849	<1e−4	0.7163	0.0027	0.8174	0.0452	0.8399	<1e−4	0.9508	<1e−4	0.8059	<1e−4
Gaussian20	0.9999	<1e−4	0.9996	<1e−4	0.8892	0.0049	0.9690	0.0008	0.9798	<1e−4	0.7939	<1e−4
Square20	0.9412	<1e−4	0.7284	0.0058	0.9507	0.0001	0.9784	<1e−4	0.9901	<1e−4	0.9452	<1e−4
Moon20	0.8476	<1e−4	0.5151	0.0091	0.8572	0.0080	0.9741	<1e−4	0.9440	<1e−4	0.9907	<1e−4
MixLarge10	0.8324	<1e−4	0.7487	0.0103	0.9236	0.0259	0.9566	0.0062	0.9823	<1e−4	0.8750	<1e−4
MixLarge20	0.9369	<1e−4	0.8850	0.0016	0.8841	0.0215	0.8850	0.0038	0.9621	<1e−4	0.9104	<1e−4
MixLarge50	0.8576	<1e−4	0.8507	0.0022	0.0036	<1e−4	0.8696	0.0025	0.9544	<1e−4	0.8750	<1e−4
Average mean(Artificial)	0.8786	<1e−4	0.7481	0.0040	0.7832	0.0132	0.9333	0.0015	0.9587	<1e−4	0.8892	<1e−4
Outdoor	0.0909	<1e−4	0.0477	0.0027	0.3670	0.0052	0.4500	<1e−4	0.5843	<1e−4	0.1301	<1e−4
Gassenor	0.0385	<1e−4	0.0425	<1e−4	0.0588	0.0002	0.4020	0.0001	0.8362	<1e−4	0.1978	<1e−4
IGBN	0.0101	<1e−4	0.2069	0.0892	0.0406	0.0023	0.2076	<1e−4	0.3301	<1e−4	0.0839	<1e−4
IABN	0.0046	<1e−4	0.0605	0.0074	0.0545	0.0003	0.0698	0.0003	0.2430	<1e−4	0.0714	<1e−4
Rialto	0.0017	<1e−4	0.0002	0.0001	0.0517	0.0036	0.0514	0.0002	0.0802	<1e−4	0.0655	<1e−4
IIAIN	<1e−4	<1e−4	0.0087	0.0043	0.0421	0.0077	0.0202	<1e−4	0.0122	<1e−4	0.0395	<1e−4
IIRIN	0.0016	<1e−4	0.0195	0.0116	0.0331	0.0020	0.0331	0.0007	0.0318	<1e−4	0.0259	<1e−4
Covtype	0.0063	0.0034	0.0015	0.0004	0.0114	0.0009	0.0057	0.0015	0.0092	<1e−4	<1e−4	<1e−4
Average mean(Real)	0.0192	0.0004	0.0484	0.0145	0.0824	0.0028	0.1550	0.0004	0.2659	<1e−4	0.0768	<1e−4
Total average mean	0.4742	0.0002	0.4189	0.0089	0.4534	0.0083	0.5670	0.0010	0.6327	<1e−4	0.5069	<1e−4

Table 13
Purity results of dynamic experiments.

Datasets	SequentialKmeans		StreamKmeans		CluStream		HistStream				DenStream	
							Kmeans		HDBSCAN			
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	0.8709	<1e−4	0.7531	0.0005	0.9627	0.0060	1.0000	<1e−4	1.0000	<1e−4	1.0000	<1e−4
Meteroit20	0.8590	<1e−4	0.6799	0.0069	0.9079	0.0043	1.0000	<1e−4	0.9999	<1e−4	1.0000	<1e−4
Circle20	0.9549	<1e−4	0.8063	0.0051	0.9206	0.0047	0.9998	<1e−4	1.0000	<1e−4	0.9998	<1e−4
Gaussian20	1.0000	<1e−4	0.9997	<1e−4	0.9179	<1e−4	1.0000	<1e−4	1.0000	<1e−4	0.8565	<1e−4
Square20	0.9301	<1e−4	0.7653	0.0141	0.9535	<1e−4	0.9999	<1e−4	1.0000	<1e−4	0.9994	<1e−4
Moon20	0.8918	<1e−4	0.6072	0.0094	0.9181	0.0100	1.0000	<1e−4	0.9998	<1e−4	0.9950	<1e−4
MixLarge10	0.8978	<1e−4	0.8330	0.0089	0.9597	0.0088	0.9996	<1e−4	1.0000	<1e−4	0.9996	<1e−4
MixLarge20	0.9347	<1e−4	0.8736	0.0018	0.9437	0.0113	0.9998	<1e−4	0.9998	<1e−4	0.9992	<1e−4
MixLarge50	0.9403	<1e−4	0.9206	0.0024	0.1055	<1e−4	0.9998	<1e−4	0.9998	<1e−4	0.9996	<1e−4
Average mean(Artificial)	0.9199	<1e−4	0.8043	0.0055	0.8433	0.0050	0.9999	<1e−4	0.9999	<1e−4	0.9832	<1e−4
Outdoor	0.2031	<1e−4	0.1724	0.0015	0.5004	0.0133	0.6786	<1e−4	0.8928	<1e−4	0.1969	<1e−4
Gassenor	0.3093	<1e−4	0.3181	<1e−4	0.3882	0.0019	0.6922	0.0010	0.8050	<1e−4	0.4967	<1e−4
IGBN	0.2138	<1e−4	0.3853	0.0738	0.2680	0.0105	0.5453	0.0001	0.5743	<1e−4	0.3789	<1e−4
IABN	0.2020	<1e−4	0.2660	0.0122	0.2950	0.0003	0.4523	0.0009	0.5175	<1e−4	0.2772	<1e−4
Rialto	0.1225	<1e−4	0.1086	0.0010	0.1953	0.0038	0.3224	0.0001	0.3132	<1e−4	0.2115	<1e−4
IIAIN	0.2980	<1e−4	0.3227	0.0075	0.3848	0.0122	0.4284	<1e−4	0.3674	<1e−4	0.4315	<1e−4
IIRIN	0.3156	<1e−4	0.3539	0.0233	0.3685	0.0011	0.5127	0.0023	0.4426	<1e−4	0.3764	<1e−4
Covtype	0.4762	0.0017	0.4742	0.0003	0.4846	0.0001	0.5318	0.0005	0.5312	<1e−4	0.4740	<1e−4
Average mean(Real)	0.2676	0.0002	0.3002	0.0150	0.3606	0.0054	0.5205	0.0006	0.5555	<1e−4	0.3554	<1e−4
Total average mean	0.6129	<1e−4	0.5671	0.0099	0.6161	0.0052	0.7743	0.0003	0.7908	<1e−4	0.6878	<1e−4

conventional clustering method, such as the efficiency of K-means and the better clustering quality of HDBSCAN on arbitrary clusters. This observation shows the exceptional flexibility of the proposed framework in adapting to different tasks by judiciously selecting different conventional clustering algorithms.

5. Conclusion

A significant advancement in clustering, especially for large datasets, is the modeling of cluster distributions using innovative representations. In this research, we present a novel approach designed for both static and dynamic clustering, based on a histogram representation using Wasserstein distance to compare distributions. The proposed framework can be used with any clustering algorithm, depending on the required cluster properties, by working on subsets of data and comparing the obtained clusters between subsets. The main goal is to reduce computation time and improve clustering quality. Specifically, experiments on real and artificial datasets show that the proposed framework can increase the computational speed and memory requirements of most traditional clustering algorithms with minimal loss of quality. For dynamic datasets, the proposed framework is competitive with existing approaches in terms of quality and complexity, while allowing great flexibility in the application of clustering algorithm.

The proposed framework is relatively easy to parameterize due to its robustness with respect to most parameters. The only parameter that needs to be adjusted is the "window size", as larger windows often lead to better quality at the cost of slower computation. However, defining an optimal size is not straightforward. It is also worth noting that in some cases, the proposed approach may be slower than conventional algorithms without an improvement in quality, and is therefore not optimal for such situations. This is especially the case when compared to clustering algorithms known for their low complexity (notably K-means and related algorithms) applied to relatively small datasets and/or with well-defined clusters, or algorithms that show a significant drop in performance on smaller datasets, such as KASP, which suffer from the decomposition of the original dataset into a collection of subsets. Nevertheless, the good experimental performance of the proposed

Table 14
NMI results of dynamic experiments.

Datasets	SequentialKmeans		StreamKmeans		CluStream		HistStream				DenStream	
							Kmeans		HDBSCAN			
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	0.9391	<1e−4	0.8062	0.0002	0.9499	0.0040	0.9848	<1e−4	0.9734	<1e−4	0.9843	<1e−4
Meteroit20	0.9180	<1e−4	0.7239	0.0018	0.9558	0.0014	0.9782	0.0001	0.9750	<1e−4	0.9119	<1e−4
Circle20	0.9664	<1e−4	0.8156	0.0025	0.9302	0.0082	0.9517	<1e−4	0.9164	<1e−4	0.9303	<1e−4
Gaussian20	0.9999	<1e−4	0.9994	<1e−4	0.9627	0.0011	0.9861	0.0005	0.9900	<1e−4	0.9010	<1e−4
Square20	0.9746	<1e−4	0.7986	0.0034	0.9753	0.0003	0.9848	0.0001	0.9909	<1e−4	0.9656	<1e−4
Moon20	0.9296	<1e−4	0.6858	0.0071	0.9557	0.0024	0.9854	<1e−4	0.9774	<1e−4	0.9950	<1e−4
MixLarge10	0.9572	<1e−4	0.8468	0.0058	0.9694	0.0068	0.9802	0.0015	0.9890	<1e−4	0.9579	<1e−4
MixLarge20	0.9662	<1e−4	0.8986	0.0014	0.9662	0.0072	0.9655	0.0013	0.9756	<1e−4	0.9761	<1e−4
MixLarge50	0.9615	<1e−4	0.9302	0.0015	0.0840	<1e−4	0.9652	0.0008	0.9797	<1e−4	0.9579	<1e−4
Average mean(Artificial)	0.9569	<1e−4	0.8339	0.0026	0.8610	0.0035	0.9758	0.0005	0.9742	<1e−4	0.9533	<1e−4
Outdoor	0.3402	<1e−4	0.2249	0.0027	0.7041	0.0019	0.7346	<1e−4	0.8258	<1e−4	0.5115	<1e−4
Gassenor	0.0920	<1e−4	0.1149	<1e−4	0.1958	0.0010	0.5079	0.0005	0.6968	<1e−4	0.3671	<1e−4
IGBN	0.0232	<1e−4	0.3047	0.1116	0.0705	0.0079	0.4189	<1e−4	0.5319	<1e−4	0.1705	<1e−4
IABN	0.0073	<1e−4	0.1614	0.0106	0.0875	0.0007	0.2152	0.0011	0.3623	<1e−4	0.1185	<1e−4
Rialto	0.0036	<1e−4	0.0006	0.0001	0.1288	0.0059	0.2059	<1e−4	0.2449	<1e−4	0.1336	<1e−4
IIAIN	0.0002	<1e−4	0.0142	0.0093	0.0836	0.0151	0.1024	<1e−4	0.0723	<1e−4	0.1384	<1e−4
IIRIN	0.0023	<1e−4	0.0240	0.0122	0.0589	0.0013	0.1479	0.0009	0.1440	<1e−4	0.0884	<1e−4
Covtype	0.0044	0.0015	0.0021	0.0002	0.0582	0.0009	0.0824	0.0011	0.0739	<1e−4	<1e−4	<1e−4
Average mean(Real)	0.0592	0.0002	0.1059	0.0183	0.1734	0.0043	0.3019	0.0004	0.3690	<1e−4	0.1910	<1e−4
Total average mean	0.5345	<1e−4	0.4913	0.0100	0.5374	0.0039	0.6587	0.0005	0.6894	<1e−4	0.5946	<1e−4

Table 15
Computation time of dynamic experiments.

Datasets	SequentialKmeans		StreamKmeans		CluStream		HistStream				DenStream	
							Kmeans		HDBSCAN			
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	601.1789	0.8627	4173.0255	5.8701	111.1533	10.8181	20.0096	0.0576	136.5564	1.9086	103.7651	2.6580
Meteroit20	598.7418	0.3835	4048.3590	0.5028	99.2521	4.6470	20.7868	0.1174	127.1455	1.1586	9854.3365	87.1560
Circle20	612.8237	3.0160	4321.1806	116.6523	100.8072	5.9999	29.4217	0.5364	494.6203	7.7403	141.2185	0.5147
Gaussian20	603.9073	0.2887	4199.7799	3.8610	107.0685	9.2846	24.4203	0.4745	76.8281	0.5030	3105.9656	6.6823
Square20	602.7990	0.6457	4173.3107	3.6656	106.2824	11.4305	25.6883	2.3608	82.2653	0.9315	4769.8329	32.2027
Moon20	637.8218	3.3549	4435.9815	18.2940	93.5763	5.8634	25.0204	3.4944	96.8381	1.1616	85.1582	2.9777
MixLarge10	343.2683	0.5161	2501.5632	5.0294	56.7119	4.6763	14.6774	0.6988	52.6882	1.3952	15 544.6931	345.2638
MixLarge20	841.1053	1.8018	7051.6518	51.9163	137.4155	9.9892	33.4324	2.2048	179.6185	11.3344	2597.6667	59.2815
MixLarge50	2867.8369	7.7141	32 481.0794	1056.8821	318.0409	8.9821	117.0566	10.4723	483.7691	17.7896	15 544.6931	345.2638
Average mean(Artificial)	856.6092	2.0648	7487.3257	140.2971	125.5898	7.9657	34.5015	2.2686	192.2588	4.8803	5749.7033	98.0001
Outdoor	1.9826	0.0119	12.0543	0.0459	18.3913	0.9477	1.3999	<1e−4	3.1280	<1e−4	0.3405	0.0117
Gassenor	7.6274	0.2071	19.1491	0.7072	1.0658	0.0581	0.6815	0.2587	4.3237	0.0406	8.5452	0.0990
IGBN	3.7845	0.2189	6.1328	0.1192	2.9484	2.3777	0.6990	0.1686	9.8378	0.0485	2.9768	0.1999
IABN	8.4413	0.2436	14.6793	0.5363	4.2191	0.8793	1.6813	0.1438	14.7541	0.0692	2.6717	0.1316
Rialto	15.5729	0.0744	47.5999	0.2665	12.2390	1.1187	2.8241	0.1433	11.8558	0.0565	8.0523	0.1978
IIAIN	70.0952	3.2963	125.4809	3.5934	30.6166	0.1338	10.3653	<1e−4	312.6661	<1e−4	82.7187	0.2978
IIRIN	68.3161	0.9216	123.9349	0.8913	30.1934	0.0635	24.9010	0.7840	378.9039	12.6035	103.2595	1.5032
Covtype	32.2408	0.4028	86.6200	1.5781	43.3229	1.1053	12.2326	1.1983	142.8380	3.6063	21.7024	0.1732
Average mean(Real)	26.0076	0.6721	54.4564	0.9672	17.8746	0.8355	6.8481	0.3371	109.7884	2.0531	28.7834	0.3268
Total average mean	465.7379	1.4094	3989.5049	74.7301	74.9003	4.6103	21.4881	1.3596	153.4492	3.5498	3057.5057	52.0362

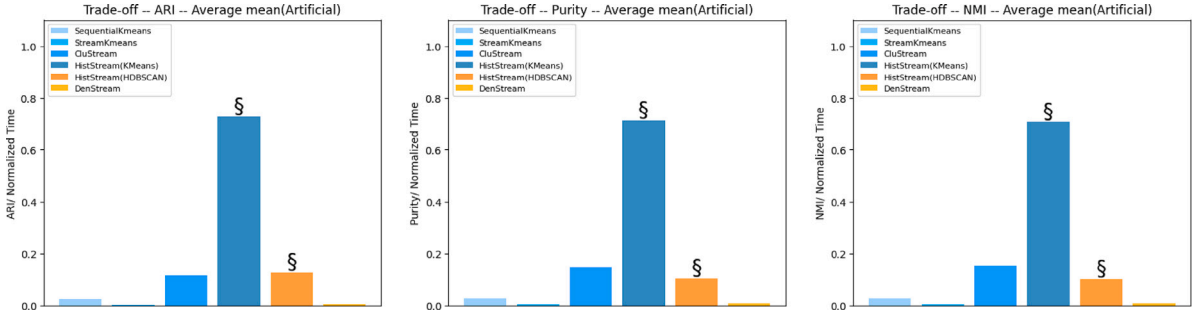

Fig. 21. Trade-off – Mean ARI (left), Purity (center) and NMI (right) values for dynamic experiments (Artificial datasets) – approaches based on K-Means are in blue and on DBSCAN are in orange – "§" denote the proposed approaches in respective categories. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

approach in most cases shows its flexibility and applicability to many clustering algorithms, especially in real-world scenarios.

In this article, we have compared the performance of standard clustering algorithms independently and when integrated with the proposed framework. In order to assess the quality of the proposed approach to data stream clustering, we have selected the standard clustering algorithms to be integrated into the framework so as to ensure comparability with other families of data stream clustering, specifically prototype-based and density-based approaches. Ultimately, the choice of which clustering algorithm to use within the framework should be tailored to each particular application, based on prior knowledge of the dataset or specific expectations about the resulting

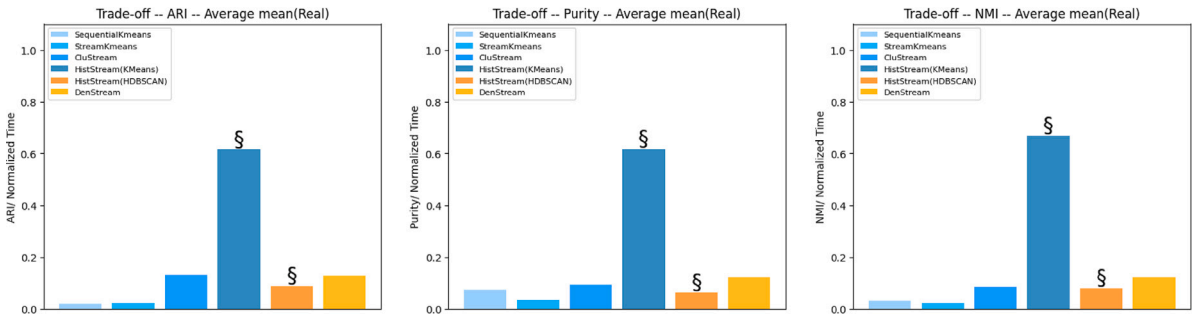

Fig. 22. Trade-off – Mean ARI (left), Purity (center) and NMI (right) values dynamic experiments (Real datasets) – approaches based on K-Means are in blue and on DBSCAN are in orange – "§" denote the proposed approaches in respective categories. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

clustering. While we did not further investigate the question of the optimal choice of clustering algorithm, an issue of obvious importance, it remains a promising area for further research. Indeed, future studies may benefit from advanced techniques such as meta-learning or ensemble clustering, which may help identify the most appropriate standard clustering algorithms for specific scenarios or broader applications, thus improving the overall performance of the framework.

In conclusion, our research highlights a promising new approach that uses histogram modeling with Wasserstein distance for clustering in both static and dynamic scenarios. It provides an efficient option for real-world applications, offering significant advantages in terms of computational time and clustering quality. A critical focus of our future studies will be to automatically determine the optimal window size to achieve the best trade-off between quality and speed, to propose more robust clustering results and to investigate the question of the selection of the clustering algorithm to integrate into the framework.

CRediT authorship contribution statement

Xiaotong Qian: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Guénaël Cabanes:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Parisa Rastin:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Mohamed Alae Guidani:** Resources, Investigation, Formal analysis. **Ghassen Marrakchi:** Validation, Resources, Investigation. **Marianne Clausel:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Nistor Grozavu:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] W.Z. Tareq, M. Davud, Classification and clustering, in: Decision-Making Models, Elsevier, 2024, pp. 351–359.
[2] G.J. Oyewole, G.A. Thopil, Data clustering: Application and trends, Artif. Intell. Rev. 56 (7) (2023) 6439–6475.
[3] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, ACM Sigmod Rec. 25 (2) (1996) 103–114.
[4] F. Murtagh, P. Legendre, Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion? J. Classification 31 (2014) 274–295.
[5] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Statistics) 28 (1) (1979) 100–108.
[6] C.C. Aggarwal, P.S. Yu, Redefining clustering for high-dimensional applications, IEEE Trans. Knowl. Data Eng. 14 (2) (2002) 210–225.
[7] N. Cristianini, J. Shawe-Taylor, J. Kandola, Spectral kernel methods for clustering, Adv. Neural Inf. Process. Syst. 14 (2001).
[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.
[9] C.M. Bishop, N.M. Nasrabadi, Pattern Recognition and Machine Learning, vol. 4, Springer, 2006.
[10] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, 1998, pp. 94–105.
[11] D. Yan, L. Huang, M.I. Jordan, Fast approximate spectral clustering, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 907–916.
[12] C. Malzer, M. Baum, A hybrid approach to hierarchical density-based cluster selection, in: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI, IEEE, 2020, pp. 223–228.
[13] X. Wang, Z. Wang, Z. Wu, S. Zhang, X. Shi, L. Lu, Data stream clustering: An in-depth empirical study, Proc. ACM Manag. Data 1 (2) (2023) 1–26.
[14] G. Sudipto, N. Mishra, R. Motwant, L. O'callaghan, Clustering data streams, in: Proc. 41st FOCS, 2000, pp. 359–366.
[15] L. O'callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani, Streaming-data algorithms for high-quality clustering, in: Proceedings 18th International Conference on Data Engineering, IEEE, 2002, pp. 685–694.
[16] D. Sculley, Web-scale k-means clustering, in: Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 1177–1178.
[17] C.C. Aggarwal, S.Y. Philip, J. Han, J. Wang, A framework for clustering evolving data streams, in: Proceedings 2003 VLDB Conference, Elsevier, 2003, pp. 81–92.
[18] F. Cao, M. Estert, W. Qian, A. Zhou, Density-based clustering over an evolving data stream with noise, in: Proceedings of the 2006 SIAM International Conference on Data Mining, SIAM, 2006, pp. 328–339.
[19] M. Hahsler, M. Bolaños, Clustering data streams based on shared density between micro-clusters, IEEE Trans. Knowl. Data Eng. 28 (6) (2016) 1449–1461.
[20] J. Gama, P.P. Rodrigues, L. Lopes, Clustering distributed sensor data streams using local processing and reduced communication, Intell. Data Anal. 15 (1) (2011) 3–28.
[21] X.H. Dang, V.C. Lee, W.K. Ng, K.L. Ong, Incremental and adaptive clustering stream data over sliding window, in: Database and Expert Systems Applications: 20th International Conference, DEXA 2009, Linz, Austria, August 31–September 4, 2009. Proceedings 20, Springer, 2009, pp. 660–674.
[22] A. Irpino, R. Verde, A new wasserstein based distance for the hierarchical clustering of histogram symbolic data, in: Data Science and Classification, Springer, 2006, pp. 185–192.
[23] M.-F. Balcan, T. Sandholm, E. Vitercik, Learning to optimize computational resources: Frugal training with generalization guarantees, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 3227–3234.

[24] P. Brito, S. Dias, Analysis of Distributional Data, CRC Press, 2022.
[25] A. Irpino, R. Verde, Basic statistics for distributional symbolic variables: a new metric-based approach, Adv. Data Anal. Classif. 9 (2015) 143–175.
[26] C. Villani, C. Villani, The Wasserstein distances, Optim. Transp. Old New (2009) 93–111.
[27] K. Pearson, Contributions to the mathematical theory of evolution, Philos. Trans. R. Soc. Lond. A 185 (1894) 71–110.
[28] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[29] D. Endres, J.E. Schindelin, A new metric for probability distributions, IEEE Trans. Inform. Theory 49 (2003) 1858–1860.
[30] C. Villani, et al., Optimal Transport: Old and New, vol. 338, Springer, 2009.
[31] Y. Rubner, C. Tomasi, L.J. Guibas, The earth mover's distance as a metric for image retrieval, Int. J. Comput. Vis. 40 (2) (2000) 99–121.
[32] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, Adv. Neural Inf. Process. Syst. 26 (2013).
[33] V.W. Berger, Y. Zhou, Kolmogorov–Smirnov test: Overview, in: Wiley statsref: Statistics reference online, Wiley Online Library, 2014.
[34] A. Ramdas, N.G. Trillos, M. Cuturi, On Wasserstein two-sample testing and related families of nonparametric tests, Entropy 19 (2017) 47.
[35] D. Revuz, M. Yor, Continuous Martingales and Brownian Motion, vol. 293, Springer Science & Business Media, 2013.
[36] R. Durrett, Probability: Theory and Examples, vol. 49, Cambridge University Press, 2019.
[37] N. Bonneel, J. Rabin, G. Peyré, H. Pfister, Sliced and radon wasserstein barycenters of measures, J. Math. Imaging Vision 51 (1) (2015) 22–45.
[38] A. Vergara, S. Vembu, T. Ayhan, M.A. Ryan, M.L. Homer, R. Huerta, Chemical gas sensor drift compensation using classifier ensembles, Sensors Actuators B 166 (2012) 320–329.
[39] V. Losing, B. Hammer, H. Wersing, KNN classifier with self adjusting memory for heterogeneous concept drift, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, IEEE, 2016, pp. 291–300.
[40] J.A. Blackard, D.J. Dean, Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables, Comput. Electron. Agric. 24 (3) (1999) 131–151.
[41] V.M.A. Souza, D.M. Reis, A.G. Maletzke, G.E.A.P.A. Batista, Challenges in benchmarking stream learning algorithms with real-world data, Data Min. Knowl. Discov. 34 (2020) 1805–1858, http://dx.doi.org/10.1007/s10618-020-00698-5.
[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[43] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: is a correction for chance necessary? in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 1073–1080.
[44] G.G. Chowdhury, Introduction to Modern Information Retrieval, Facet publishing, 2010.
[45] L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech. Theory Exp. 2005 (09) (2005) P09008.
[46] S. Dhar, M.K. Kundu, Accurate multi-class image segmentation using weak continuity constraints and neutrosophic set, Appl. Soft Comput. 112 (2021) 107759.
[47] A. Morison, M. Ulvrova, S. Labrosse, B4rsh, theofatou, tfrass49, Stag-Python/StagPy: v0.20.1, Zenodo, 2024, http://dx.doi.org/10.5281/zenodo.13684760.
[48] J. Montiel, M. Halford, S.M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H.M. Gomes, J. Read, T. Abdessalem, et al., River: machine learning for streaming data in python, J. Mach. Learn. Res. 22 (110) (2021) 1–8.

Xiaotong Qian received a Master's degree Data Mining and Decision-Making (EID2) at Sorbonne Paris Nord University, France in 2021. She is currently pursuing a Ph.D. degree at CY Cergy Paris University, France. Her research interest include unsupervised learning and data mining.

Guénaël Cabanes is an academic researcher at the University of Lorraine, France, member of the Lorraine Research Laboratory in Computer Science and its Application. His main research interest lies in data mining, particularly in unsupervised learning for dynamic and complex data, with applications in image and text mining and complex systems modeling.

Parisa Rastin received her Ph.D. in Computer Science from the University of Sorbonne Paris Nord. She is currently an associate professor at the University of Lorraine, École nationale supérieure des mines de Nancy. Her main research interests are in the area of unsupervised learning of dynamic and complex data with applications to text mining.

Mohamed Alae Guidani has graduated as an engineer from École nationale supérieure des mines de Nancy in 2018 with a major in Applied Mathematics, Mohamed Alae Guidani currently works in Data and Analytics consulting at EY. He analyzes complex datasets and implements data-driven strategies to drive innovation and optimize business performance.

Ghassen Marrakchi is a Master's Degree Student in Data Mining and Decision-Making (EID2) at Sorbonne Paris Nord University. He received his bachelor's degree in Computer Science from the University of Monastir. As a research enthusiast, he explores different Machine Learning fields, such as unsupervised learning.

Marianne Clausel is a Professor at the University of Lorraine. Her research is in statistical learning and time series analysis. She received her Ph.D. in Applied Mathematics from University of Creteil, France in 2008.

Nistor Grozavu is a full professor of Computer Science at CY Cergy Paris University. He holds a Ph.D. in Unsupervised Machine Learning (2009) from the University Paris 13 and an HdR (2020) from the University Sorbonne Paris Nord. His research interests include unsupervised learning, transfer learning, dimensionality reduction, quantum machine learning.