mathematics

Article

A Clustering System for Dynamic Data Streams Based on Metaheuristic Optimisation

Jia Ming Yeoh [1], Fabio Caraffini [1,*], Elmina Homapour [1], Valentino Santucci [2] and Alfredo Milani [3]

1 Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK; jiamingyeoh@gmail.com (J.M.Y.); elmina.homapour@dmu.ac.uk (E.H.)
2 Department of Humanities and Social Sciences, University for Foreigners of Perugia, piazza G. Spitella 3, 06123 Perugia, Italy; valentino.santucci@unistrapg.it
3 Department of Mathematics and Computer Science, University of Perugia, via Vanvitelli 1, 06123 Perugia, Italy; alfredo.milani@unipg.it
* Correspondence: fabio.caraffini@dmu.ac.uk

Received: 31 October 2019; Accepted: 10 December 2019; Published: 12 December 2019

Abstract: This article presents the Optimised Stream clustering algorithm (OpStream), a novel approach to cluster dynamic data streams. The proposed system displays desirable features, such as a low number of parameters and good scalability capabilities to both high-dimensional data and numbers of clusters in the dataset, and it is based on a hybrid structure using deterministic clustering methods and stochastic optimisation approaches to optimally centre the clusters. Similar to other state-of-the-art methods available in the literature, it uses "microclusters" and other established techniques, such as density based clustering. Unlike other methods, it makes use of metaheuristic optimisation to maximise performances during the initialisation phase, which precedes the classic online phase. Experimental results show that OpStream outperforms the state-of-the-art methods in several cases, and it is always competitive against other comparison algorithms regardless of the chosen optimisation method. Three variants of OpStream, each coming with a different optimisation algorithm, are presented in this study. A thorough sensitive analysis is performed by using the best variant to point out OpStream's robustness to noise and resiliency to parameter changes.

Keywords: dynamic stream clustering; online clustering; metaheuristics; optimisation; population based algorithms; density based clustering; k-means centroid; concept drift; concept evolution

1. Introduction

Clustering is the process of grouping homogeneous objects based on the correlation among similar attributes. This is useful in several common applications that require the discovery of hidden patterns among the collective data to assist decision making, e.g., bank transaction fraud detection [1], market trend prediction [2,3] and a network intrusion detection system [4]. Most traditional clustering algorithms developed rely on multiple iterations of evaluation on a fixed set of data to generate the clusters. However, in practical applications, these detection systems are operating daily, whereby millions of input data points are continuously streamed indefinitely, hence imposing speed and memory constraints. In such dynamic data stream environments, keeping track of every historical data would be highly memory expensive and, even if possible, would not solve the problem of analysing big data within the real-time requirements. Hence, a method of analysing and storing the essential information of the historical data in a single pass is mandatory for clustering data streams.

In addition, the dynamic data clustering algorithm needs to address two special characteristics that often occur in data streams, which are known as "*concept drift*" and "*concept evolution*" [5]. Concept drift refers to the change of underlying concepts in the stream as time progresses, i.e., the change in the relationship between the attributes of the object within the individual clusters. For example, customer behaviour in purchasing trending products always changes in between seasonal sales. Meanwhile, concept evolution occurs when a new class definition has evolved in the data streams, i.e., the number of clusters has changed due to the creation of new clusters or the deprecation of old clusters. This phenomenon often occurs in the detection system whereby an anomaly has emerged in the data traffic. An ideal data stream clustering algorithm should address these two main considerations to detect and adapt effectively to changes in the dynamic data environment.

Based on recent literature, metaheuristics for black-box optimisation have been greatly adopted in traditional static data clustering [6]. These algorithms have a general purpose application domain and often display self-adaptive capabilities, thus being able to tackle the problem at hand, regardless of its nature and formulation, and return near-optimal solutions. For clustering purposes, the so-called "population based" metaheuristic algorithms have been discovered to be able to achieve better global optimisation results than their "single solution" counterparts [7]. Amongst the most commonly used optimisation paradigms of this kind, it is worth mentioning the established Differential Evolution (DE) framework [8–10], as well as more recent nature inspired algorithms from the Swarm Intelligence (SI) field, such as the Whale Optimisation Algorithm (WOA) [11] and the Bat-inspired algorithm in [12], here referred to as BAT. Although the literature is replete with examples of data clustering strategies based on DE, WOA and BAT for the static domain, as, e.g., those presented in [13–16], little has been done for the dynamic environment due to the difficulties in handling data streams. The current state of dynamic clustering is therefore unsatisfactory as it mainly relies on algorithms based on techniques such as density microclustering and density grid based clustering, which require the tuning of several parameters to work effectively [17].

This paper presents a methodology for integrating metaheuristic optimisation into data stream clustering, thus maximising the performance of the classification process. The proposed model does not require specifically tailored optimisation algorithms to function, but it is a rather general framework to use when highly dynamic streams of data have to be clustered. Unlike similar methods, we do not optimise the parameters of a clustering algorithm, but use metaheuristic optimisation in its initialisation phase, in which the first clusters are created, by finding the optimal position of their centroids. This is a key step as the grouped points are subsequently processed with the method in [18] to form compact, but informative, microclusters. Hence, by creating the optimal initial environment for the clustering method, we make sure that the dynamic nature of the problem will not deteriorate its performances. It must be noted that microclusters are lighter representations of the original scenario, which are stored to preserve the "memory" of the past classifications. These play a major role since they aid subsequent clustering processes when new data streams are received. Thus, a non-optimal microclusters store in memory can have catastrophic consequences in terms of classification results. In this light, our original use of the metaheuristic algorithm finds its purpose, and results confirm the validity of our idea. The proposed clustering scheme efficiently tracks changes and spots patterns accordingly.

The remainder of this paper has the following structure:

- Section 2 discusses the recent literature and briefly explains the logic behind the leading data stream clustering algorithms;
- Section 3 establishes the motivations and objectives of this research and presents the used metaheuristic optimisation methods, the employed performance metrics and the considered datasets for producing numerical results;
- Section 4 gives a detailed description of each step involved in the proposed clustering system, clarifies its working mechanism and shows methodologies for its implementation;
- Section 5 describes the performance metrics used to evaluate the system and provides experimental details to reproduce the presented results;
- Section 6 presents and comments on the produced results, including a comparison among different variants of the proposed system, over several evaluation metrics;

- Section 7 outlines a thorough analysis of the impact of the parameter setting for the optimisation algorithm on the overall performance of the clustering system;
- Section 8 summarises the conclusions of this research.

2. Background

There are two fundamentals aspects to take into consideration in data stream clustering, namely concept drift and concept evolution. The first aspect refers to the phenomenon when the data in the stream undergo changes in the statistical properties of the clusters with respect to the time [19,20] while the second to the event when there is an unseen novel cluster appearing in the stream [5,21].

Time window models are deployed to handle concept drift in data streams. These are usually embedded into clustering algorithms to control the quantity of historical information used in analysing dynamic patterns. Currently, there are four predominant window models in the literature [22]:

- the "damped time window" model, where historical data weights are dynamically adjusted by fixing a rate of decay according to the number of observations assigned to it [23];
- the "sliding time window" model, where only the most recent past data observations are considered with a simple First-In-First-Out (FIFO) mechanism. as in [24];
- the "landmark time window" model, where the data stream is analysed in batches by accumulating data in a fixed width buffer before being processed;
- the "tilted time window" model, where the granularity level of weights gradually decreases as data points get older.

As for concept evolution, most of the existing data stream clustering algorithms are designed following a two phase approach, i.e., consisting of an online clustering process followed by an offline one, which was first proposed in [25]. In this work, the concept of microclusters was also defined to design the so-called CluStream algorithm. This method forms microclusters having statistical features representing the data stream online. Similar microclusters are then merged into macro-clusters, keeping only information related to the centre of the densest region. This is performed offline, upon user request, as it comes with information losses since merged clusters can no longer be split again to obtain the original ones.

In terms of online microclustering, most algorithms in the literature are distance based [17,22,26], whereby new observations are either merged with existing microclusters or form new microclusters based on a distance threshold. The earliest form of distance based clustering strategy was the process of extracting information about a cluster into the form of a Clustering Feature (CF) vector. Each CF usually consists of three main components: (1) a linear combination of the data points referred to as Linear Sum vector $\overrightarrow{LS}$; (2) a vector $\overrightarrow{SS}$ whose components are the Squared Sums of the corresponding data points' components; (3) the number N of points in a cluster.

As an instance, the popular CluStream algorithm in [25] makes use of CF and the tilted time window model. During the initialisation phase, data points are accumulated to a certain amount before being converted into some microclusters. On the arrival of new streams, new data are merged with the closest microclusters if their distance from the centre of the data point to the centre of the microclusters is within a given radius (i.e., the ϵ-neighbourhood method). If there is no suitable microclusters within this range, a new microclusters is formed. When requested, the CluStream uses the k-means algorithm [27] to generate macroclusters from microclusters in its offline phase. It also implements an ageing mechanism based on timestamps to remove outdated clusters from its online components.

Another state-of-the-art algorithm, i.e. DenStream, was proposed in [18] as an extension of CluStream using the damped time window and a novel clustering strategy named "time-faded CF". DenStream separates the microclusters into two categories: the potential core microclusters (referred to as p-microclusters) and the outlier microclusters (referred to as o-microclusters). Each entry of the CF is subject to a decay function that gradually reduces the weight of each microcluster at a regular evaluation interval period. When the weight falls below a threshold value, the affected p-microclusters are degraded to the o-microclusters, and they are removed from the o-microclusters if the weights deteriorate further. On the other hand,o-microclusters that have their weights improved are promoted to p-microclusters. This concept allows new and old clusters to form online gradually, so addressing the concept evolution issue. In the offline phase, only the p-microclusters are used for generating the final clusters. Similar p-microclusters are merged employing a density based approach based on the ϵ-neighbourhood method. Unlike other commonly used methods, in this case, clusters can assume an arbitrary shape, and no a priori information is needed to fix the number of clusters.

An alternative approach was given in [28], where the proposed STREAM algorithm did not store CF vectors, but directly computed centroids on-the-fly. This was done by solving the "k-means clustering" problem to identify the centroids of K clusters. The problem was structured in a form whereby the distance from data points to the closest cluster had associated costs. Using this framework, the clustering task was defined as a minimisation problem to find the number and position of centroids that yielded the lowest costs. To process an indefinite length of streaming data, the landmark time window was used to divide the streams into n batches of data, and the K-means problem solving was performed on each chunk. Although the solution was plausible, the algorithm was evaluated to be time consuming and memory expensive in processing streaming data.

The OLINDDA method proposed in [29] extends the previously described centroid approach by integrating the ϵ-neighbourhood concept. This was used to detect drifting and new clusters in the data stream, with the assumption that drift changes occurred within the existing cluster region, whilst new clusters formed outside the existing cluster region. The downside of the centroid approach was that the number of K centroids needed to be known a priori, which is problematic in a dynamic data environment.

There is one shortcoming for the two phase approach, i.e., the ability to track changes in the behaviour of the clusters is linearly proportional to the frequency of requests for the offline component [30]. In other words, the higher the sensitivity to changes, the higher the computational cost. To mitigate these issues, an alternative approach has been explored by researchers to merge these two phases into a single online phase. FlockStream [31] deploys data points into a virtual mapping of a two-dimensional grid, where each point is represented as an agent. Each agent navigates around the virtual space according to a model mimicking the behaviour of flocking birds, as done in the most popular SI algorithms, e.g., those in [32–34]. The agent behaviour was designed in a way such that similar (according to a given metric) birds would move in the same direction as the closest neighbours, forming different groups of the flock. These groups can be seen as clusters, thus eliminating the need for a subsequent offline phase.

MDSC [35] is another single phase method exploiting the SI paradigm inspired by the density based approached introduced in DenStream. In this method, the Ant Colony Optimisation (ACO) algorithm [36] is used to group similar microclusters optimally during the online phase. In MDSC, a customised ϵ-neighbourhood value is assigned to each cluster to enable "multi-density" clusters to be discovered.

Finally, it is worth mentioning the ISDI algorithm in [37], which is equipped with a windowing routine to analyse and stream data from multiple sources, a timing alignment method and a deduplication algorithm. This algorithm was designed to deal with data streams coming from different sources in the Internet of Things (IoT) systems and can transform multiple data streams, having different attributes, into cleaner datasets suitable for clustering. Thus, it represents a powerful tool allowing for the use of streams classifiers, as, e.g., the one proposed in this study, in IoT environments.

3. Motivations, Objectives and Methods

Clustering data streams is still an open problem with room for improvement [38]. Increasing the classification efficiency in this dynamic environment has a great potential in several application fields, from intrusion detection [39] to abnormality detection in patients' physiological data streams [40]. In this light, the proposed methodology draws its inspiration from key features of the successful methods listed in Section 2, with the final goal of improving upon the current state-of-the-art.

A hybrid algorithm is then designed by employing, along with standard methods, as, e.g., CF vectors and the landmark time windows model, modern heuristic optimisation algorithms. Unlike similar approaches available in the literature [36,41,42], the optimisation algorithm is here used during the online phase to create optimal conditions for the offline phase. This novel approach is described in detail in Section 4.

To select the most appropriate optimisation paradigm, three widely used algorithms, i.e., WOA, BAT and DE, were selected from the literature and compared between them. We want to clarify that the choice of using three metaheuristic methods, rather than other exact or iterative techniques, was made to be able to deal with the challenging characteristics of the optimisation problem at hand, e.g., the dimensionality of the problem can vary according to the dataset, and the objective functions is highly non-linear and not differentiable, which make them not applicable or time inefficient.

A brief introduction of the three selected algorithms is given below in Section 3.1. Regardless of the specific population based algorithm used for performing the optimisation step, each candidate solution must be encoded as an n-dimensional real valued vector representing the K cluster centres for initialising the following density based clustering method.

Two state-of-the-art deterministic data stream clustering algorithms, namely DenStream and CluStream, are also included in the comparative analysis to further validate the effectiveness of the proposed framework.

The evaluation methodology employed in this work consisted of running classification experiments over the datasets in Section 3.2 and measuring the obtained performances through the metrics defined in Section 3.3.

3.1. *Metaheuristic Optimisation Methods*

This section gives details on the implementation of the three optimisation methods used to test the proposed system.

3.1.1. The Whale Optimization Algorithm

The WOA algorithm is a swarm based stochastic metaheuristic algorithm inspired by the hunting behaviour of humpback whales [11]. It is based on a mathematical model updated by iterating the three search mechanisms described below:

- the "shrinking encircling prey" mechanism is exploitative and consists of moving candidate solutions (i.e., the whales) in a neighbourhood of a the current best solution in the swarm (i.e., the prey solution) by implementing the following equation:

$$\vec{x}(t+1) = \vec{x}_{\text{best}}(t) - \vec{A} * \vec{D}_{\text{best}} \quad \text{with} \quad \begin{cases} \vec{A} = 2\vec{a} * \vec{r} - \vec{a} \\ \vec{D}_{\text{best}} = 2\vec{r} * \overrightarrow{x_{\text{best}}}(t) - \vec{x}(t) \end{cases} \tag{1}$$

  where: (1) $\vec{a}$ is linearly decreased from two to zero as iterations increase (to represent shrinking, as explained in [7]); (2) $\vec{r}$ is a vector whose components are randomly sampled from $[0, 1]$ (t is the iteration counter); (3) the "$*$" notation indicates the pairwise products between two vectors.
- the "spiral updating position" mechanism is also exploitative and mimics the swimming pattern of humpback whales towards prey in a helix shaped form through Equations (2) and (3):

$$\vec{x}(t+1) = e^{bl} * \cos(2\pi l) * \left|\vec{d}\right| + \vec{x}_{\text{best}}(t) \tag{2}$$

  with:

$$\vec{d} = \left|\vec{x}_{\text{best}}(t) - \vec{x}(t)\right| \tag{3}$$

where b is a constant value for defining the shape of the logarithmic spiral; l is a random vector in $[-1, 1]$; the "$|\ldots|$" symbol indicates the absolute value of each component of the vector;

- the "search for prey" mechanism is exploratory and uses a randomly selected solution $\vec{x}_{\text{rand}}$ as an "attractor" to move candidate solutions towards unexplored areas of the search space and possibly away from local optima, according to Equations (4) to (5):

$$\vec{x}(t+1) = \vec{x}_{\text{rand}}(t) - \vec{A} * \vec{D} + \text{rand} \tag{4}$$

with:

$$\vec{D}_{\text{rand}} = \left| 2\vec{a} * \vec{r} * \overrightarrow{x_{\text{rand}}}(t) - \vec{x} \right|. \tag{5}$$

The reported equations implemented a search mechanism that mimics movements made by whales. Mathematically, it is easier to understand that some of them refer to explorations moves across the search space, while others are exploitation moves to refine solutions within their neighbourhood. To have more information on the metaphor inspiring these equations, their formulations and their role in driving the research within the algorithm framework, one can see the survey article in [6]. A detailed scheme describing the coordination logic of the three previously described search mechanism is reported in Algorithm 1.

Algorithm 1 WOA pseudocode.

```
1: Generate initial whale positions $x_i$, where $i = 1, 2, 3, \ldots, NP$
2: Compute the fitness of each whale solution, and identify $x_{\text{best}}$
3: while $t <$ max iterations do
4:     for $i = 1, 2, \ldots, NP$ do
5:         Update $a, A, C, l, p$
6:         if $p < 0.5$ then
7:             if $|A| < 1$ then
8:                 Update the position of current whale $x_i$ using Equation (1)
9:             else if $|A| \geq 1$ then
10:                $x_{\text{rand}} \leftarrow$ random whale agent
11:                Update the position of current whale $x_i$ with Equation (4)
12:            end if
13:        else if $p \geq 0.5$ then
14:            Update the position of current whale $x_i$ with Equation (2)
15:        end if
16:    end for
17:    Calculate new fitness values
18:    Update $X_{\text{best}}$
19:    $t = t + 1$
20: end while
21: Return $x_{\text{best}}$
```

With reference to Algorithm 1, the initial swarm is generated by randomly sampling solutions in the search; the best solution is kept up to date by replacing it only when an improvement on the fitness value occurs; the optimisation process lasts for a prefixed number of iterations, here indicated with *max budget*; the probability of using the shrinking encircling rather than the spiral updating mechanism was fixed at 0.5.

3.1.2. The BAT Algorithm

The BAT algorithm was a swarm based searching algorithm inspired by the echolocation abilities of bats [12]. Bats use sound wave emissions to generate an echo that measures the distance of its prey based on the loudness and time difference of the echo and sound wave. To reproduce this system and exploit it for optimisation purposes, the following perturbation strategy must be implemented:

$$f_i = f_{\min} + (f_{\max} - f_{\min}) \cdot \beta \tag{6}$$

$$v_i(t+1) = v_i(t) + (x_i(t) - x_{\text{best}}) \cdot f_i \tag{7}$$

$$x_i(t+1) = x_i(t) + v_i(t) \tag{8}$$

where x_i is the position of the candidate solution in the search space (i.e., the bat), v_i is its velocity, f_i is referred to as the "wave frequency" factor and β is a random vector in $[0,1]^n$ (where n is the dimensionality of the problem). f_{min} and f_{max} represent the lower and upper bounds of the frequency, respectively. Typical values are within 0 and 100. When the bat is close to the prey (i.e., current best solution), it gradually reduces the loudness of its sound wave while increasing the pulse rate. The pseudocode depicted in Algorithm 2 shows the the working mechanism of the BAT algorithm.

Algorithm 2 BAT pseudocode.

```
1: Generate initial bats $X_i$ ($i = 1,2,3,\ldots,NP$) and their velocity vectors $v_i$
2: Compute the fitness values, and find $x_{\text{best}}$
3: Initialise pulse frequency $f_i$ at $x_i$
4: Initialise pulse rate $r_i$ and loudness $A_i$
5: while $t <$ max iterations do
6:     for $i = 1,2,3,\ldots,NP$ do
7:         $x_{\text{new}} \leftarrow$ move $x_i$ to a new position with Equations (6)–(8)
8:     end for
9:     for $i = 1,2,3,\ldots,NP$ do
10:        if rand() $> r_i$ then
11:            $x_{\text{new}} \leftarrow x_{\text{best}}$ added with a random
12:        end if
13:        if rand() $< A_i$ and $f(x_{\text{new}})$ improved then
14:            $X_i \leftarrow x_{\text{new}}$
15:            Increase $r_i$ and decrease $A_i$
16:        end if
17:    end for
18:    Update $x_{\text{best}}$
19:    $t = t + 1$
20: end while
21: Return $x_{\text{best}}$
```

To have more detailed information on the equations used to perturb the solutions within the search space in the BAT algorithm, we suggest reading [43].

3.1.3. The Differential Evolution

The Differential Evolution (DE) algorithms are efficient metaheuristics for global optimisation based on a simple and solid framework, first introduced in [8], which only requires the tuning of three parameters, namely the scale factor $F \in [0,2]$, the crossover ratio $CR \in [0,1]$ and the population size NP. As shown in Algorithm 3, despite using crossover and mutation operators, which are typical of evolutionary algorithms, it does not require any selection mechanism as solutions are perturbed one at a time by means of the one-to-one spawning mechanising from the SI field. Several DE variants can be obtained by using different combinations of crossover and mutation operators [44]. The so-called "DE/best/1/bin" scheme was adopted in this study, which employs the best mutation strategy and the binomial crossover approach. The pseudocode and other details regarding these operators are available in [10].

Algorithm 3 DE pseudocode.

```
1: Generate initial population $x_i$ with $i = 1, 2, 3, \ldots, NP$
2: Compute the fitness of each individual, and identify $x_{\text{best}}$
3: while $t <$ max iterations do
4:     for $i = 1, 2, 3, \ldots, NP$ do
5:         $X_m \leftarrow$ mutation                                  ▷ "best/1" as explained in [10]
6:         $x_{\text{off}} \leftarrow \text{crossover}(X_i, X_m)$       ▷ "bin" as explained in [10]
7:         Store the best individual between $x_{off}$ and $x_i$ in the $i^{\text{th}}$ position of a new population
8:     end for
9:     end
10:    Replace the current population with the newly generated population
11:    Update $x_{\text{best}}$
12: end while
13: Return $x_{\text{best}}$
```

3.2. Datasets

Four synthetic datasets were generated using the built-in stream data generator of the "Massive Online Analysis" (MOA) software [45]. Each synthetic dataset represents different data streaming scenarios with varying dimensions, clusters numbers, drift speed and frequency of concept evolution. These datasets are:

- the *5C5C* dataset, which contains low-dimensional data with a low rate of data changes;
- the 5C10C dataset, which contains low-dimensional data with a high rate of data changes;
- the 10D5C dataset, which is a 5C5C variant containing high-dimensional data;
- the 10D10C dataset, which is a *5C10C* variant containing high-dimensional data.

Moreover, the KDD-99 dataset [46], containing real network intrusion information, was also considered in this study. It must be highlighted that the original KDD-99 dataset contained 494,021 data entries, representing network connections generated in military network simulations. However, only 10% of the entries were randomly selected for this study. Each data entry contained 41 features and one output column to distinguish the attack connection from the normal network connection. The attacks can be further classified into 22 attack types. Streams are obtained by reading each entry of the dataset sequentially.

Details on the five employed datasets are given in Table 1.

Table 1. Name and description of the synthetic datasets and real dataset.

Name	Dimension	Cluster No.	Samples	Drift Speed	Event Frequency	Type
5D5C	5	3–5	100,000	1000	10,000	Synthetic
5D10C	5	6–10	100,000	5000	10,000	Synthetic
10D5C	10	3–5	100,000	1000	10,000	Synthetic
10D10C	10	6–10	100,000	5000	10,000	Synthetic
KDD–99	41	2–23	494,000	Not Known	Not Known	Real

3.3. Performance Metrics

To perform an informative comparative analysis, three metrics were cherry picked from the data stream analysis literature [41,42]. These are referred to as the F-measure, purity and Rand index [47].

Mathematically, these metrics are expressed with the following equations:

$$\text{F-Measure} = \frac{1}{k} \sum_{i=1}^{k} \text{Score}_{C_i} \tag{9}$$

$$\text{Purity} = \frac{1}{k} \sum_{i=1}^{k} \text{Precision}_{C_i} \tag{10}$$

$$\text{Rand Index} = \frac{\text{True Positive} + \text{True Negative}}{\text{All Data Instances}} \tag{11}$$

where:

$$\text{Precision}_{C_i} = \frac{V_{i\text{sum}}}{n_{C_i}} \tag{12}$$

$$\text{Score}_{C_i} = 2 \cdot \frac{\text{Precision}_{C_i} \cdot \text{Recall}_{C_i}}{\text{Precision}_{C_i} + \text{Recall}_{C_i}} \tag{13}$$

$$\text{Recall}_{C_i} = \frac{V_{i\text{sum}}}{V_{i\text{total}}} \tag{14}$$

and:

- C is the solution returned by the clustering algorithm (i.e., the number of clusters k);
- C_i is the the ith cluster ($i = \{1, 2, \ldots, k\}$);
- V_i is the class label with the highest frequency in C_i;
- $V_{i\text{sum}}$ is the number of instances labelled with V_i in C_i;
- $V_{i\text{total}}$ is the total number of V_i instances identified in the totality of clusters returned by the algorithm.

The F-measure represents the harmonic mean of the precision and recall scores, where the best value of one indicates ideal precision and recall, while zero is the worst scenario.

Purity is used to measure the homogeneity of the clusters. Maximum purity is achieved by the solution when each cluster only contains a single class.

The Rand index computes the accuracy of the clustering solution from the actual solution, based on the ratio of correctly identified instances among all the instances.

4. The Proposed System

This article proposes "OpStream", an Optimised Stream clustering algorithm. This clustering framework consisted of two main parts: the initialisation phase and the online phase.

During the initialisation phase, a number λ of data points are accumulated through a landmark time window, and the unclassified points are initialised into groups of clusters via the centroid approach, i.e., generating K centroids of clusters among the points.

In the initialisation phase, the landmark time window is used to collect data points, which are subsequently grouped into clusters by generating K centroid. The latter are generated by solving K-centroid cost optimisation problems with a fast and reliable metaheuristic for optimisation. Hence, their position is optimal and leads to high quality predictions.

Next, during the online phase, the clusters are maintained and updated using the density based approach, whereby incoming data points with similar attributes (i.e., according to the ϵ-neighbourhood method) form dense microclusters in between two data buffers, namely p-microclusters and o-microclusters. These are converted into microclusters with CF information to store a "light" version of previous scenarios in this dynamic environment.

In this light, the proposed framework is similar to advanced single phase methods. However, it requires a preliminary optimisation process to boost its classification performances.

Three variants of OpStream were tested by using the three metaheuristic optimisers described in Section 3. These stochastic algorithms (as the optimisation process is stochastic) are compared against the two DenStream and CluStream state-of-the-art deterministic stream clustering algorithms.

The following sections describe each step of the OpStream algorithm.

4.1. The Initialisation Phase

This step can be formulated as a real valued global optimisation search problem and addressed with the metaheuristic of black-box optimisation. To achieve this goal, a cost function must be designed to allow for the individualisation of the optimal position of the centroid of a cluster. These processes have to be iterated K times to then form K clusters by grouping data according to their distance from the optimal centroids.

The formulation of the cost function plays a key part. In this research, the "Cluster Fitness" (CF) function from [48] was chosen as its maximisation leads to a high intra-cluster distance, which is desirable. Its mathematical formulation, for the κth ($\kappa = 1, 2, 3, \ldots K$) cluster, is given below:

$$CF_\kappa = \frac{1}{K} \sum_{\kappa=1}^{K} S_\kappa \tag{15}$$

from where it can be observed that it is computed by averaging the K clusters' silhouettes "S_κ". These represent the average dissimilarity of all the points in the cluster and are calculated as follows

$$S_\kappa = \frac{1}{n_k} \sum_{i \in C_\kappa} \frac{\beta_i - \alpha_i}{\max\{\alpha_i, \beta_i\}} \tag{16}$$

where α_i and β_i are the "inner dissimilarity" and the "outer dissimilarity", respectively.

The former value measures the average dissimilarity between a data point i and other data points in its own cluster C_{κ^*}. Mathematically, this is expressed as:

$$\alpha_i = \frac{1}{(n_{k^*} - 1)} \sum_{\substack{j \in C_{\kappa^*} \\ j \neq i}} \text{dist}(i, j) \tag{17}$$

with $\text{dist}(i, j)$ being the Euclidean distance between the two points and n_{k^*} is the total number of points in cluster C_{κ^*}. The lower the value, the better the clustering accuracy.

The latter value measures the minimum distance between a data point i to the centre of all clusters, excluding its own cluster C_{κ^*}. Mathematically, this is expressed as:

$$\beta_i = \min_{\substack{\kappa=1,\ldots,K \\ k \neq \kappa^*}} \left(\frac{1}{n_k} \sum_{\substack{j \in C_\kappa \\ k \neq \kappa^*}} \text{dist}(i, j) \right) \tag{18}$$

where n_{k^*} is the number of points in cluster C_{κ^*}. The higher the value, the better the clustering.

These two values are contained in $[-1, 1]$, whereby one indicates the ideal case and -1 the most undesired one.

A similar observation can be done for the fitness function CF_κ [48]. Hence, the selected metaheuristics have to be set up for a maximisation problem. This is not an issue since every real valued problem of this kind can be easily maximised with an algorithm designed for minimisation purposes by simply timing the fitness function by -1, and vice versa.

Regardless of the dimensionality of the problem n, which depends on the dataset (as shown in Table 1), all input data were normalised within $[0, 1]$. Thus, the search space for all the optimisation process was the hyper-cube defined as $[0, 1]^n$.

4.2. The Online Phase

Once the initial clusters have been generated, by optimising the cost function formulated in Section 4.1, clustered data points must be converted into microclusters. This step requires the extraction of CF vectors. Subsequently, a density based approach was used to cluster the data stream online.

4.2.1. Microclusters' Structure

In OpStream, each CF must contain four components, i.e., CF $=$ [N, $\overrightarrow{\text{LS}}$, $\overrightarrow{\text{SS}}$, timestamp], where

- N$\in \mathbb{N}$ is the number of data points in the microclusters;

- $\overrightarrow{\mathrm{LS}} \in \mathbb{R}^n$ is the Linear Sum of the data points in the microcluster, i.e.,

$$\overrightarrow{\mathrm{LS}} = \sum_{i=1}^{N} \overrightarrow{x_i};$$

- $\overrightarrow{\mathrm{SS}} \in \mathbb{R}^n$ is the squared sum of the data points in the microclusters, i.e.,

$$\overrightarrow{\mathrm{SS}}[j] = \sum_{i=1}^{N} \left(\overrightarrow{x_i}[j]\right)^2; \qquad j = 1, 2, 3, \ldots, n$$

- timestamp indicates when the microclusters was last updated, and it is needed to implement the ageing mechanism, used to remove outdated microclusters while new data accumulated in the time window are available, defined via the following equation:

$$\text{age} = T - \text{timestamp} \tag{19}$$

  where T is the current timestamp in the stream and a threshold, referred to as β, is used to discriminate between suitable and outdated data points.

  From CF, the centre c and radius r of a microclusters are computed as follows:

$$c = \frac{\overrightarrow{\mathrm{LS}}}{\mathrm{N}} \tag{20}$$

$$r = \sqrt{\frac{\overrightarrow{\mathrm{SS}}}{\mathrm{N}} - \left(\frac{\overrightarrow{\mathrm{LS}}}{\mathrm{N}}\right)^2} \tag{21}$$

as indicated in [18,42].

The obtained r value is used to initialise the ϵ-neighbourhood approach (i.e., $r = \epsilon$), leading to the formation of microclusters as explained in Section 2. This microclusters, which are derived from a cluster formed in the initialisation phase, is now stored in the p-microclusters buffer.

4.2.2. Handling Incoming Data Points

In OpStream, for each new time window, a data point p is first converted into a "degenerative" microcluster m_p containing a single point and having the following initial CF properties:

$$\begin{aligned} m_p.\mathrm{N} &= 1 \\ m_p.\overrightarrow{\mathrm{LS}}_i &= p_i \qquad i = 1, 2, 3 \ldots, n \\ m_p.\overrightarrow{\mathrm{SS}}_i &= p_i^2 \qquad i = 1, 2, 3 \ldots, n \\ m_p.\text{timestamp} &= T \end{aligned}$$

Subsequently, initial microclusters have to be merged. This task can efficiently be addressed by considering pairs of microclusters, say, e.g., m_i and m_j, and computing their Euclidean distance $\text{dist}(c_{m_i}, c_{m_j})$. If m_i is the cluster to be merged, its radius r must be worked out as shown in Section 4.2.1 and then be merged with m_i if:

$$\text{dist}(c_{m_i}, c_{m_j}) \leq \epsilon \qquad (\epsilon = r). \tag{22}$$

Two microclusters satisfying the condition expressed with Equation (22) are said to be "density reachable". The process described above is repeated until there are no longer density reachable microclusters. Every time two microclusters are merged, e.g., m_i and m_j, the CF properties of the newly generated microclusters, e.g., m_k, are assigned as follows:

$$\begin{aligned} m_k.\mathrm{N} &= m_i.\mathrm{N} + m_j.\mathrm{N} \\ m_k.\overrightarrow{\mathrm{LS}} &= m_i.\overrightarrow{\mathrm{LS}} + m_j.\overrightarrow{\mathrm{LS}} \\ m_k.\overrightarrow{\mathrm{SS}} &= m_i.\overrightarrow{\mathrm{SS}} + m_j.\overrightarrow{\mathrm{SS}} \\ m_k.\mathrm{timestamp} &= T \end{aligned}$$

where T is the time at which the two microclusters were merged.

When the condition in Equation (22) is no longer met by a microcluster, this is moved to the p-microclusters buffer. If the newly added microclusters and other clusters in the p-microclusters buffer are density reachable, then they are merged. Otherwise, a new independent cluster is stored in this buffer.

This mechanism is performed by a software agent, referred to as the "Incoming Data Handler" (IDH), whose pseudocode is reported in Algorithm 4 to further clarify this process and allow for its implementation.

Algorithm 4 IDH pseudocode.

```
1: Input: Data point $p$
2: Convert $p$ into microcluster $m_p$
3: Initialise $merged$ = false
4: for $mc$ in p-microclusters do
5:     if $merged$ is false then
6:         if $m_p$ is density reachable to $mc$ then
7:             if new radius $\leq \epsilon_{mc}$ then
8:                 Merge $m_p$ with $mc$
9:             else
10:                Add $m_p$ to p-microclusters
11:            end if
12:            $merged = true$
13:        end if
14:    end if
15: end for
16: if $merged$ is false then
17:     for each $mc$ in o-microclusters do
18:         if $merged$ is false then
19:             if $m_p$ is density reachable to $mc$ then
20:                 if new radius $\leq \epsilon_{mc}$ then
21:                     Merge $m_p$ with $mc$
22:                     $merged = true$
23:                 end if
24:             end if
25:         end if
26:     end for
27: end if
28: if $merged$ is false then
29:     Add $m_p$ to o-microclusters
30: end if
31: end
32: return
```

4.2.3. Detecting and Forming New Clusters

Once microclusters in the o-microclusters buffer are all merged, as explained in Section 4.2.2, only the minimum possible number of microclusters with the highest density exists. The microclusters with the highest number of points N is then moved to an empty set C to initialise a new cluster. After calculating its centre c, with Equation (20), and radius r, with Equation (21), the ϵ-neighbourhood method is again used to find density reachable microclusters. Among them, a process is undertaken to detect the so-called border microclusters [35] inside C, which obviously are not present during the first iteration as *C* initially contains only one microcluster. Border microclusters are defined as density reachable microclusters that have a density level that is below the density threshold of the first microclusters present in C. Having a threshold that is too high, cluster C will not expand, whilst having a value that is too low, cluster C will contain dissimilar microclusters. Based on the experimental data from the original paper [35], a 10% threshold yields good performance.

Once the border microclusters are identified, only surrounding microclusters that are density reachable to the non-border microclusters are moved to form part of C, according to the process indicated in Section 4.2.2. Figure 1 graphically depicts C. The microclusters marked in red colour do not form as part of C because it is density reachable only to border microclusters of C.

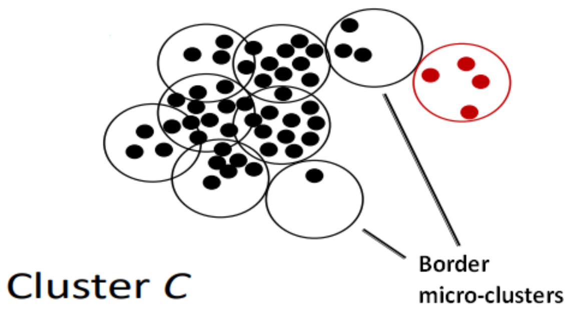

Figure 1. A graphical representation of the "border microclusters" concept [35].

This process is iterated as shown in Algorithm 5. The final version of *C* is finally moved to the most appropriate buffer according to its size, i.e., if "C.N $\geq$ minClusterSize", all its microclusters are merged together, and the newly generated cluster *C* is moved to the p-microclusters buffer. If this does not occur, the cluster *C* is not generated by merging its microclusters, but they are simply left in the o-microclusters buffer. The recommended method to fix the minClusterSize parameter is:

$$\text{minClusterSize} = \begin{cases} 2 & \text{if } 10\% \text{ of } \lambda \leq 2 \\ 10\% \text{ of } \lambda & \text{otherwise} \end{cases} \tag{23}$$

These tasks are performed by the New Cluster Generator (NCG) software agent, whose pseudocode is shown in Algorithm 5.

Algorithm 5 New cluster generation pseudocode.

```
1: Input: o-microclusters
2: while o-microclusters is not empty do
3:     Initialise cluster $C$ using $mc$ with highest $N$
4:     addedMc = true
5:     while addedMc is true do
6:         addedMc = false
7:         for $mc$ in o-microclusters do
8:             if $mc$ is density reachable to any non-border mc in $C$ then
9:                 Add $mc$ into $C$
10:                Remove $mc$ from o-microclusters
11:                addedMc = true
12:            end if
13:        end for
14:    end while
15:    if the size of $C$ is $\geq$ minClusterSize then    ▷ minClusterSize is initialised with Equation (23)
16:        Merge microclusters $mc$ in $C$
17:        Add $C$ into p-microclusters
18:    end if
19: end while
```

4.3. OpStream General Scheme

The proposed OpStream method involves the use of several techniques, such as metaheuristic optimisation algorithms, density based and k-means clustering, etc., and requires a software infrastructure coordinating activities as those performed by the IDH and NCG agents. Its architecture is outlined with the pseudocode in Algorithm 6.

Algorithm 6 OpStream pseudocode

```
1: Launch AS                                              ▷ initialised with Equation (24)
2: initialisedFlag = false
3: while stream do
4:     Add data point $p$ into window
5:     if initialisedFlag is true then
6:         Handle incoming data streams with IDH,              ▷ i.e., Algorithm 4
7:     end if
8:     if window is full then
9:         if initialisedFlag is false then
10:            Optimise centres' positions, and initialise clusters   ▷ e.g., with Algorithm 1, 2 or 3
11:            initialisedFlag = true
12:        else
13:            Look for and generate new clusters with NCG,        ▷ i.e., Algorithm 5
14:        end if
15:    end if
16: end while
```

It must be added that an Ageing System (AS) is constantly run to remove outdated clusters. Despite its simplicity, its presence is crucial in dynamic environments. An integer parameter β (equal to four in this study) is used to compute the age threshold as shown below:

$$\textit{age threshold} = \beta \cdot \lambda \tag{24}$$

so that if a microcluster has not been updated in four consecutive windows, it will be removed from the respective buffer.

5. Experimental Setup

As discussed in Section 3, OpStream's performances were evaluated across four synthetic datasets and one real dataset using three popular performance metrics. Two deterministic state-of-the-art stream clustering algorithms, i.e., DenStream [18] and CluStream [25], were also run with the suggested parameter settings available in their original articles for caparison purposes.

The WOA algorithm was initially picked to implement the OpStream framework, as this framework is currently being intensively exploited for classification purposes, but two more variants employing BAT and DE (as described in Section 3) were also run to: (1) show the flexibility of OpStream in the use of different optimisation methods; (2) display its robustness and superiority to deterministic approaches regardless of the optimiser used; (3) establish the preferred optimisation method over the specific datasets considered in this study. For the sake of clarity, these three variants are referred to as WOAS-OpStream, BAT-OpStream and DE-OpStream to represent the respective metaheuristic optimiser used. To reproduce the results presented in this article, the employed parameter setting of each metaheuristic, as well as other algorithmic details are reported below:

- WOA: swarm Size = 20;
- BAT: swarm size = 20, $\alpha = 0.53$, $\gamma = 4.42$, $r_i = 0.42$, $A_i = 0.50$ $(i = 1, 2, 3, \ldots, n)$;
- DE: population Size = 20, $F = 0.5$, $CR = 0.5$;
- the "*max Iterations*" value is set to 10 for all three algorithms to ensure a fair comparison (the computational budget was purposely kept low due to the real-time nature of the problem);

- the three optimisation algorithms were equipped with the "toroidal" correction mechanism to handle infeasible solutions, i.e., solutions generated outside of the search space (a detailed description of this operator is available in [10]).

Furthermore, the following parameter values were also required to run the OpStream framework:

- $\lambda = 1000, \epsilon = 0.1, \beta = 4$;

Section 7 explains the role played by these parameters and how their suggested values were determined.

Thus, a total of five clustering algorithms was considered in the experimentation phase. These were executed, with the aid of the MOA platform [45], 30 times over each dataset (the instances' order was randomly changed for each repetition) to produce, for each evaluation metric, average $\pm$ standard deviation values. To further validate our conclusions statistically, the outcome of the Wilcoxon rank-sum test [49] (with the confidence level equal to 0.05) is also reported in all tables with the compact notation obtained from [50], where (1) a "$+$" symbol next to an algorithm indicates that it was outperformed by the reference algorithm (i.e., WOA-OpStream); (2) a "$-$" symbol indicates that the reference algorithm was outperformed; (3) a "$=$" symbol shows that the two stochastic optimisation processes were statistically equivalent.

6. Results and Discussion

A table was prepared for each evaluation metric, each one displaying the average value, standard deviation and the outcome of the Wilcoxon rank-sum test (W) over the 30 performed runs. The best performance on each dataset is highlighted in boldface.

Table 2 reports the results in terms of the F-measure. According to this metric, the three OpStream variants generally outperformed the deterministic algorithms. The only exception is registered over the *KDDC–99* dataset, where DenStream displayed the best performance. From the statistical point of view, WOA-OpStream was significantly better than CluStream (with five "$+$" out of five cases), clearly preferable to DenStream (with four "$+$" out of five cases), equivalent to the DE-OpStream variant and approximately equivalent to BAT-OpStream.

Table 2. Average F-measure value $\pm$ standard deviation and Wilcoxon rank-sum test (reference = Whale Optimisation Algorithm (WOA)-OpStream) for WOA-OpStream against BAT-OpStream, Differential Evolution (DE)-OpStream, DenStreamand CluStreamon each dataset. W, Wilcoxon rank-sum test.

Dataset	WOA-OpStream	BAT-OpStream	W	DE-OpStream	W	DenStream	W	CluStream	W
5D5C	**0.924 $\pm$ 0.042**	0.907 $\pm$ 0.041	+	0.923 $\pm$ 0.040	=	0.645 $\pm$ 0.016	+	0.584 $\pm$ 0.033	+
5D10C	0.868 $\pm$ 0.042	0.873 $\pm$ 0.048	=	**0.879 $\pm$ 0.036**	=	0.551 $\pm$ 0.019	+	0.602 $\pm$ 0.008	+
10D5C	0.903 $\pm$ 0.031	0.899 $\pm$ 0.028	=	**0.904 $\pm$ 0.030**	=	0.619 $\pm$ 0.021	+	0.398 $\pm$ 0.006	+
10D10C	0.873 $\pm$ 0.035	**0.878 $\pm$ 0.028**	=	0.876 $\pm$ 0.027	=	0.543 $\pm$ 0.020	+	0.380 $\pm$ 0.004	+
KDDC–99	0.460 $\pm$ 0.000	0.460 $\pm$ 0.000	=	0.460 $\pm$ 0.000	=	**0.650 $\pm$ 0.000**	-	0.140 $\pm$ 0.000	+

Similarly, regarding Table 3, WOA-OpStream showed a slightly better statistical behaviour than BAT-OpStream, and it was statistically equivalent to DE-OpStream, also in terms of purity. However, according to this metric, the stochastic classifiers did not outperform the deterministic ones, but had quite similar performances. In terms of the average value over the 30 repetitions, DE-OpStream and DenStream had the highest purity.

Finally, the same conclusions obtained with the F-measure were drawn by interpreting the results in Table 4, where the Rand index metric was used to evaluate the classification performances. Indeed, all three OpStream variants statistically outperformed the deterministic methods. This goes to show that the proposed method was performing very well regardless of the optimisation strategy, and it was always better or competitive with state-of-the-art algorithms. Unlike the case in Table 2, the best performances were in terms of the average value or those obtained with DE rather than WOA.

However, the difference between the two variants was minimal, and the Wilcoxon rank-sum test did not detect differences between the two variants.

Table 3. Average purity ± standard deviation and Wilcoxon rank-sum test (reference = WOA-OpStream) for WOA-OpStream against BAT-OpStream, DE-OpStream, DenStream and CluStream on each dataset.

Dataset	WOA-OpStream	BAT-OpStream	W	DE-OpStream	W	DenStream	W	CluStream	W
5D5C	0.998 ± 0.006	0.996 ± 0.007	+	0.998 ± 0.006	=	**1.000 ± 0.000**	=	0.998 ± 0.004	=
5D10C	0.992 ± 0.016	0.984 ± 0.022	=	0.987 ± 0.022	=	**1.000 ± 0.000**	-	0.998 ± 0.004	-
10D5C	**1.000 ± 0.000**	0.998 ± 0.004	+	**1.000 ± 0.000**	=	**1.000 ± 0.000**	=	**1.000 ± 0.000**	=
10D10C	0.999 ± 0.002	**1.000 ± 0.002**	=	**1.000 ± 0.002**	=	**1.000 ± 0.000**	=	**1.000 ± 0.000**	=
KDDC–99	**1.000 ± 0.000**	**1.000 ± 0.000**	=	**1.000 ± 0.000**	=	**1.000 ± 0.000**	=	0.420 ± 0.000	+

Table 4. Average Rand index ± standard deviation and Wilcoxon rank-sum Test (reference = WOA-OpStream) for WOA-OpStream against BAT-OpStream, DE-OpStream, DenStream and CluStream on each dataset.

Dataset	WOA-OpStream	BAT-OpStream	W	DE-OpStream	W	DenStream	W	CluStream	W
5D5C	**0.951 ± 0.018**	0.945 ± 0.019	+	0.951 ± 0.017	=	0.825 ± 0.005	+	0.596 ± 0.041	+
5D10C	0.944 ± 0.020	0.947 ± 0.020	=	**0.949 ± 0.016**	=	0.753 ± 0.013	+	0.625 ± 0.018	+
10D5C	0.934 ± 0.017	0.932 ± 0.016	=	**0.935 ± 0.017**	=	0.814 ± 0.007	+	0.432 ± 0.033	+
10D10C	0.939 ± 0.020	0.941 ± 0.018	=	**0.942 ± 0.017**	=	0.746 ± 0.016	+	0.400 ± 0.020	+
KDDC–99	0.620 ± 0.000	0.620 ± 0.000	=	0.620 ± 0.000	=	0.820 ± 0.000	-	**0.940 ± 0.000**	-

Summarising, OpStream displayed the best global performance, with WOA-OpStream and DE-OpStream being the most preferable variants. Statistically, WOA-OpStream and DE-OpStream had equivalent performances over different datasets and according to three different evaluation metrics. In this light, the WOA variant was preferred as it required the tuning of only two parameters, against the three required in DE, to function optimally.

A final observation can be done by separating the results from the synthetic datasets and *KDDC–99*. If in the first case, the supremacy of OpStream was evident; a deterioration of the performances can be noted when the later dataset was used. In this light, one can understand that the proposed method presented room for improvement of handling data streams with an uneven distribution of class instances as those presented in *KDDC–99* [51].

7. Further Analyses

In the light of what was observed in Section 6, the WOA algorithm was preferred over DE and BAT to perform the optimisation phase. Hence, it was reasonable to consider the WOA-OpStream variant as the default OpStream algorithm implementation.

This section concludes this research with a thorough analysis of this variant in terms of sensitivity, scalability, robustness and flexibility to handle overlapping multi-density clusters.

7.1. Scalability Analysis

A scalability analysis was performed to test how OpStream behaved, in terms of execution time (seconds) needed to process 100,000 data points per dataset, over datasets having increasing dimension values or an increasing number of clusters. Datasets suitable for this purpose are easily generated with the MOA platform, as previously done for the comparative analysis in Section 6.

This experimentation was performed on a personal computer equipped with an AMD Ryzen 5 2500 U Quad-Core (2.0 GHz) CPU Processor and 8 GB RAM. OpStream was run with the following parameter settings: $\lambda = 1000$, $\epsilon = 0.1$, $\beta = 4$, WOA swarm size equal to 20 and maximum number of allowed iterations equal to 10.

Execution time is plotted over increasing dimension values (for the the data points) in Figure 2.

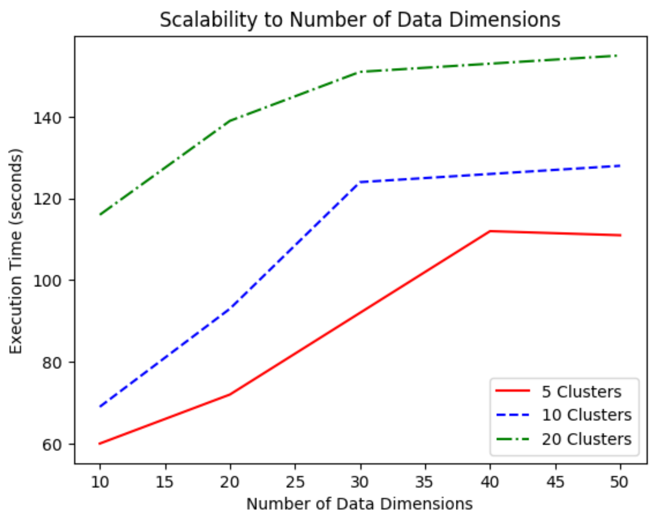

Figure 2. Scalability to the number of data dimensions (data dimension value).

Execution time is plotted over increasing number of clusters (in the datasets) in Figure 3.

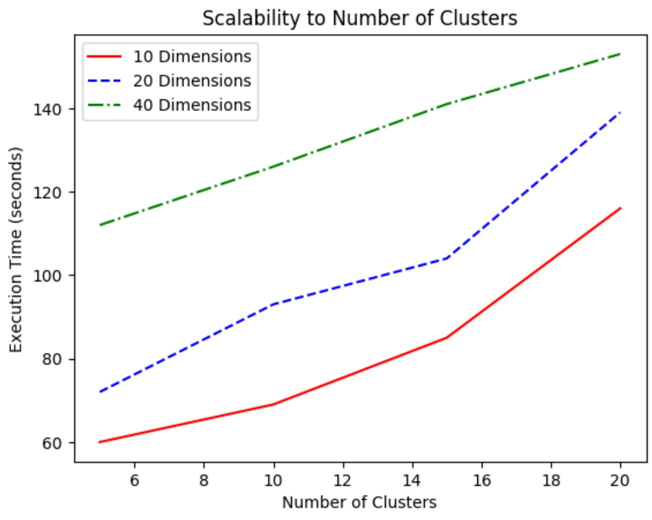

Figure 3. Scalability (number of clusters).

Regardless of the number of clusters, the execution time seemed to grow linearly with the dimensionality of the data points, for low dimension values, to then saturate when the dimensionality was high. The lower the number of clusters, the later the saturation phenomenon took place. With five clusters, this occurred at approximately 40 dimension values. In the case of 20 clusters, saturation occurred earlier at approximately 25 dimension values. This was one of the strengths of the proposed method, as its time complexity did not require polynomial times.

Conversely, no saturation took place when the execution time was measured by increasing the number of clusters. In this case as well, the time complexity seemed to grow linearly with the number of clusters.

7.2. Noise Robustness Analysis

The MOA platform allowed for the injection of increasing noise levels into the datasets *5D10C* and *10D5C*.

The five noise levels indicated in Figures 5 and 6 were used, and the OpStream algorithm was run 30 times for each one of the 10 classification problems (i.e., five noise levels $\times$ 2 datasets) with the same parameter setting used in Section 7.1. Results were collected to display the average F-measure, purity and Rand index relative to *5D10C*, i.e., Table 5, and *10D5C*, i.e., Table 6.

Table 5. Average OpStream performances over *5D10C* at multiple noise levels.

Noise Level	F-Measure	Purity	Rand Index
0%	0.846	0.986	0.934
3%	0.798	0.988	0.909
5%	0.808	0.983	0.892
8%	0.768	0.993	0.881
10%	0.774	0.972	0.880

Table 6. Average OpStream performances over or *10D5C* at multiple noise levels.

Noise Level	F-Measure	Purity	Rand Index
0%	0.902	1.000	0.936
3%	0.856	1.000	0.899
5%	0.854	1.000	0.889
8%	0.848	1.000	0.884
10%	0.835	1.000	0.865

From these results, it is clear that OpStream was able to retain approximately 95% of its original performance as long as the level did not exceed the 5% level. Then, performances slightly decreased. OpStream seemed to be robust to noise, in particular when classifying datasets with high-dimensional data points and a low number of clusters.

7.3. *Sensitivity Analysis*

Five parameters must be tuned before using OpStream for clustering dynamic data streams. In this section, the impact of each parameter on the classification performance is analysed in terms of the Rand index value.

To perform a thorough sensitivity analysis

- the size λ of the landmark time window model was examined in the range $[100, 5000] \in \mathbb{N}$;
- the ϵ value for the ϵ-neighbourhood method was examined within $[0, 1] \in \mathbb{R}$;
- the effect of the age threshold was examined by tuning β in the interval $[1, 10] \in \mathbb{N}$;
- the WOA swarm sizes under analysis were obtained by adding 5 candidate solutions per experiment, from an initial value of 5 candidate solutions to a maximum of 30 candidate solutions;
- the computational budget for the optimisation process, expressed in terms of "max iterations"number, was increased by 5 iterations per experiment starting with 5 up to a maximum of 30 iterations.

OpStream was run on three datasets for this sensitivity analysis, namely *5D10C*, *10D5C* and *KDDC–99*, and the results are graphically shown in the figures reported below.

Figure 4 shows that too high window sizes were not beneficial, and the best performances were obtained in the rage of $[500, 2000]$ data points. In particular, a peak was obtained with a size of 1000 for the two artificially prepared datasets. Conversely, slightly inferior sizes might be preferred for the *KDDC–99* dataset. In general, there was no need to use more than 2000 data points, as the performance would remain constant or slightly deteriorate.

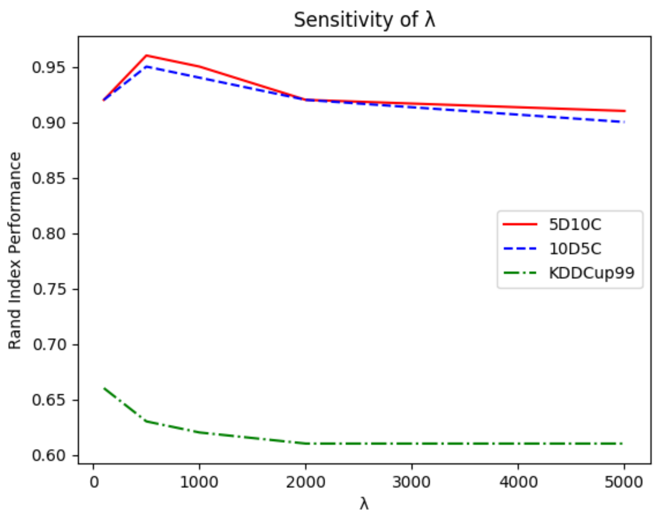

Figure 4. Sensitivity to the windows size parameter λ.

With reference to Figure 5, it was evident that ϵ did not require fine tuning in a wide range as the best performances were obtained within $[0.1, 0.2]$ and then linearly decreased over the remaining admissible values. This can be easily explained as too low values would prevent microclusters from merging while too high values would force OpStream to merge dissimilar clusters. In both cases, the outcome would be a very poor classification. This observation facilitated the tuning process as it meant that it was worth trying values for ϵ of 0.1 and 0.2 and perhaps one or two intermediary values.

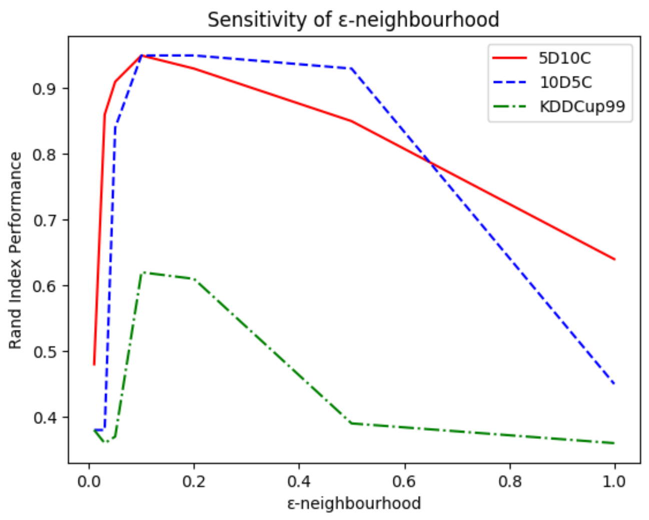

Figure 5. Sensitivity to the ϵ-neighboured parameter ϵ.

As for β, the curves in Figure 6 show that OpStream was not sensitive to the value chosen for removing outdated clusters as long as $\beta \geq 2$. This meant that clusters could be technically left in the buffers for a long time without affecting the performance of the classifier. From a more practical point of view, for memory issues, it was preferable to free buffers from unnecessary microclusters in a timely manner. A sensible choice is $\beta = 4$, as too low values might prevent similar clusters from being merged due to the lack of time required for performing such a process.

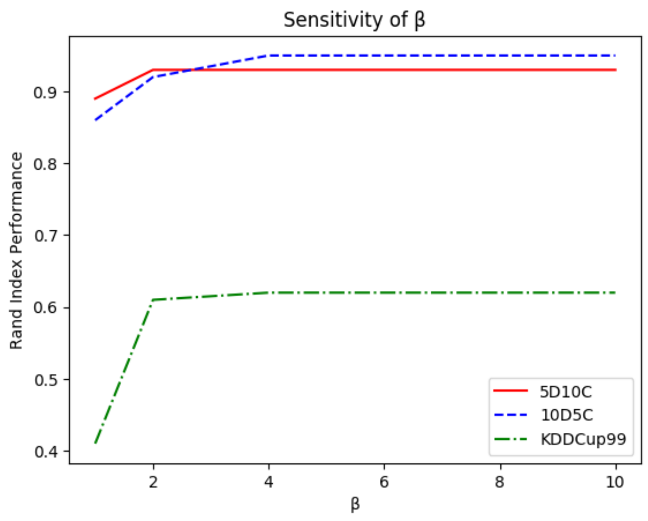

Figure 6. Sensitivity to the Ageing System (AS) parameter β.

It can be noted that a small number of candidate solutions was used for the optimisation phase. This choice was made for multiple reasons. First, it has been recently shown that a high number of solutions can increase the structural biases of the algorithm [52], which is not wanted as the algorithm has to be "general-purpose" to handle all possible scenarios obtained in the dynamic domain. Second, due to the time limitations related to the nature of this application domain, a high number of candidate solutions was to be avoided as it would slow down the converging process. This is not admissible in the real-time domain where also the computational budget is kept very low. Third, as shown in Figure 7, the WOA method used in OpStream seemed to work efficiently regardless of the employed number of candidate solutions, as long as it was greater than 20.

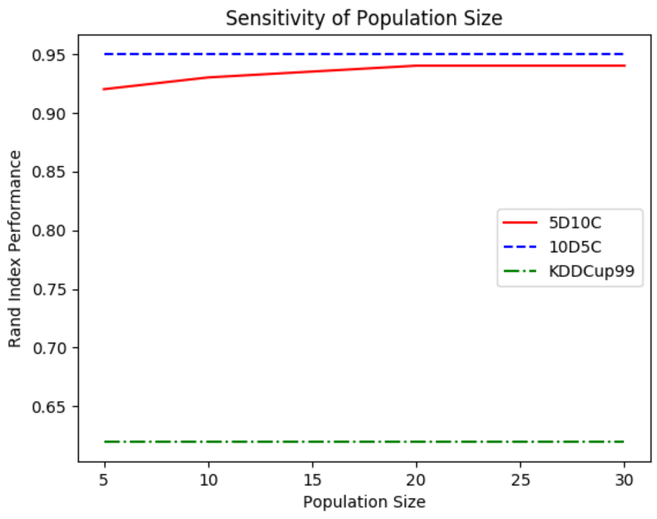

Figure 7. WOA sensitivity to the swarm size.

Similar conclusions can be made for the computational budget. According to Figure 8, it was not necessary to prolong the duration of the WOA optimisation process for more than 10 iterations. This makes sense in dynamic domains, where the problem changes very frequently, thus making the exploitation phase less important.

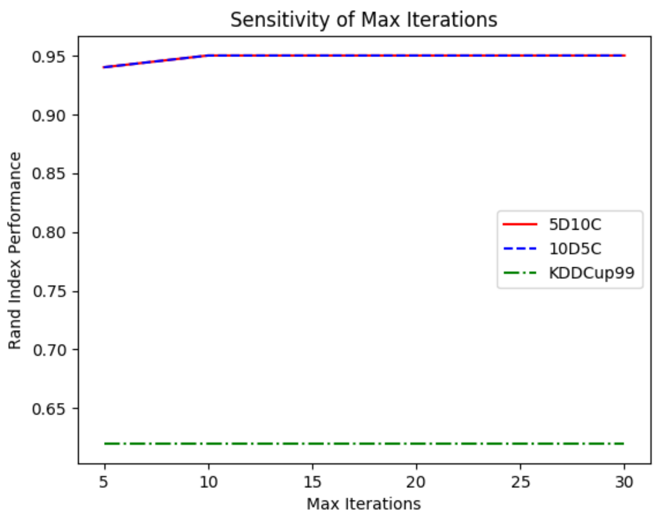

Figure 8. Sensitivity to $maxIterations$.

7.4. *Comparison with Past Studies on Intrusion Detection*

One last comparison was performed to complete this study. This was performed on a specific application domain, i.e., network intrusion detection, by means of the "KDD–cup 99" database [53]. The comparison algorithms employed in this work, i.e., DenStream and CluStream, were both tested on this dataset in their original papers [18,25], respectively. Despite the fact that OpStream is not meant for datasets with overlapping multi-density clusters, as in KDD–cup 99, we executed it over such a dataset to test its versatility. Results are displayed in Table 7 where the last column indicates the average performance of the clustering method by computing:

$$\text{AVG} = \frac{\text{F-Measure} + \text{Purity} + \text{Rand Index}}{3}. \tag{25}$$

Table 7. Results obtained with the KDD–cup 99 [53] dataset for intrusion detection.

Algorithm	F-Measure	Purity	Rand Index	AVG
OpStream	0.46	1.00	0.62	0.69
DenStream	0.65	1.00	0.82	0.82
ClusStream	0.14	0.42	0.94	0.50

Surprisingly, OpStream had an AVG better performance than CluStream, due to the fact that it significantly outperformed it in terms of F-measure and purity and displayed a state-of-the-art behaviour in terms of the purity value. As expected, DenStream provided the best performance, thus being preferable in this application domain unless a fast real-time response is required. In the latter case, its high computational cost could prevent DenStream from being successfully used [18].

8. Conclusions and Future Work

Experimental numerical results showed that the proposed OpStream algorithm was a promising tool for clustering dynamic data streams as it was competitive and outperformed the state-of-the-art on several occasions. This approach could then be applied in several challenging application domains where satisfactory results are difficult to obtain with clustering methods. Thanks to its optimisation driven initialisation phase, OpStream displays high accuracy, robustness to noise in the dataset and versatility. In particular, we found out that its WOA implementation was efficient, scalable (both in term of dataset dimensionality and number of clusters) and resilient to parameters' variations. Moreover, due to a low number of parameters to be tuned in WOA, this optimisation algorithm was preferred over other approaches returning similar accuracy values as DE and BAT. Finally, this study clearly showed that hybrid clustering methods are promising and more suitable than classic approaches to address challenging scenarios.

Possible improvements can be done to address some of the aspects arising during the experimental section. First, the deterioration of the performance over unevenly distributed datasets, as *KDDC-99*, will be investigated. A simple solution to this problem is to embed non-density based clustering algorithms into the OpStream framework. Second, since the proposed methods do not benefit from preceding optimisation processes (as shown in Figure 8), probably because of the dynamic nature of the problem, the optimisation algorithm employing "restart" mechanisms will be implemented and tested. These algorithms usually work on a very short computational budget and handle dynamic domains better than others by simply re-sampling the initial point where a local search routine is applied, as, e.g., [54], or by also adding to it information from the previous past solution with the "inheritance" method [55–57].

It is also worthwhile to extend OpStream to handle overlapping multi-density clusters in dynamic data streams, as these cases are not currently addressable and are common in some real-world scenarios, such as network intrusion detection [51] and Landsat satellite image discovery [58].

Author Contributions: All authors contributed to the draft of the manuscript and read and approved the final manuscript. J.M.Y. made a major contribution by implementing the proposed system and the code to run all experiments. F.C. made a major contribution in writing the manuscript with J.M.Y. and in the implementation and correct use of the optimisation algorithms. E.H., V.S. and A.M. contributed in writing the manuscript, checking the validity of the proposed evaluation methods and their appropriateness.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Modi, K.; Dayma, R. Review on fraud detection methods in credit card transactions. In Proceedings of the 2017 International Conference on Intelligent Computing and Control (I2C2), Coimbatore, India, 23–24 June 2017.
2. Moodley, R.; Chiclana, F.; Caraffini, F.; Carter, J. Application of uninorms to market basket analysis. *Int. J. Intell. Syst.* **2019**, *34*, 39–49. [CrossRef]
3. Moodley, R.; Chiclana, F.; Caraffini, F.; Carter, J. A product-centric data mining algorithm for targeted promotions. *J. Retail. Consum. Serv.* **2019**. [CrossRef]
4. Zarpelão, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A survey of intrusion detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. [CrossRef]
5. Masud, M.M.; Chen, Q.; Khan, L.; Aggarwal, C.; Gao, J.; Han, J.; Thuraisingham, B. Addressing Concept-Evolution in Concept-Drifting Data Streams. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 14–17 December 2010; pp. 929–934.
6. Gharehchopogh, F.S.; Gholizadeh, H. A comprehensive survey: Whale Optimization Algorithm and its applications. *Swarm Evol. Comput.* **2019**, *48*, 1–24. [CrossRef]
7. Hardi, M.; Mohammed, S.U.U.; Rashid, T.A. A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm. *Comput. Intell. Neurosci.* **2019**, *2019*, 25. [CrossRef]
8. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
9. Caraffini, F.; Kononova, A.V. Structural bias in differential evolution: A preliminary study. *AIP Conf. Proc.* **2019**, *2070*, 020005. [CrossRef]
10. Caraffini, F.; Kononova, A.V.; Corne, D. Infeasibility and structural bias in Differential Evolution. *Inf. Sci.* **2019**, *496*, 161–179. [CrossRef]
11. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
12. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. *Nat. Inspired Coop. Strateg. Optim.* **2010**, *284*, 65–74. [CrossRef]
13. Chen, G.; Luo, W.; Zhu, T. Evolutionary clustering with differential evolution. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1382–1389.

14. Carnein, M.; Trautmann, H. evoStream—Evolutionary Stream Clustering Utilizing Idle Times. *Big Data Res.* **2018**, *14*, 101–111. [CrossRef]
15. Nasiri, J.; Khiyabani, F. A Whale Optimization Algorithm (WOA) approach for Clustering. *Cogent Math. Stat.* **2018**, *5*. [CrossRef]
16. Nandy, S.; Sarkar, P. Chapter 8–Bat algorithm–based automatic clustering method and its application in image processing. In *Bio-Inspired Computation and Applications in Image Processing*; Academic Press: Cambridge, MA, USA, 2016; pp. 157–185.
17. Kokate, U.; Deshpande, A.; Mahalle, P.; Patil, P. Data Stream Clustering Techniques, Applications, and Models: Comparative Analysis and Discussion. *Big Data Cogn. Comput.* **2018**, *2*, 32. [CrossRef]
18. Cao, F.; Ester, M.; Qian, W.; Zhou, A. Density based Clustering over an Evolving Data Stream with Noise. In Proceedings of the 2006 SIAM Conference on Data Mining, Bethesda, MD, USA, 20–22 April 2006; Volume 2006, pp. 328–339.
19. Sun, J.; Fujita, H.; Chen, P.; Li, H. Dynamic financial distress prediction with concept drift based ontime weighting combined with Adaboost support vector machine ensemble. *Knowl. Based Syst.* **2017**, *120*, 4–14. [CrossRef]
20. Brzezinski, D.; Stefanowski, J. Prequential AUC: Properties of the area under the ROC curve for data streams with concept drift. *Knowl. Inf. Syst.* **2017**, *52*, 531–562. [CrossRef]
21. ZareMoodi, P.; Kamali Siahroudi, S.; Beigy, H. Concept-evolution detection in non-stationary data streams: A fuzzy clustering approach. *Knowl. Inf. Syst.* **2019**, *60*, 1329–1352. [CrossRef]
22. Carnein, M.; Trautmann, H. Optimizing Data Stream Representation: An Extensive Survey on Stream Clustering Algorithms. *Bus. Inf. Syst. Eng. Int. J. Wirtsch.* **2019**, *61*, 277–297. [CrossRef]
23. Gao, X.; Ferrara, E.; Qiu, J. Parallel clustering of high-dimensional social media data streams. In Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, China, 4–7 May 2015; pp. 323–332.
24. Gao, L.; Jiang, Z.Y.; Min, F. First-Arrival Travel Times Picking through Sliding Windows and Fuzzy C-Means. *Mathematics* **2019**, *7*, 221. [CrossRef]
25. Aggarwal, C.C.; Yu, P.S.; Han, J.; Wang, J. A Framework for Clustering Evolving Data Streams. In Proceedings of the 2003 VLDB Conference, Berlin, Germany, 9–12 September 2003; Volume 29, pp. 81–92.
26. Madhulatha, T.S. Overview of streaming-data algorithms. *arXiv* **2012**, arXiv:1203.2000.
27. Hartigan, J.A.; Wong, M.A. Algorithm AS 136: A K-Means Clustering Algorithm. *Appl. Stat.* **1979**, *28*, 100–108. [CrossRef]
28. O'Callaghan, L.; Mishra, N.; Meyerson, A.; Guha, S.; Motwani, R. Streaming-data algorithms for high-quality clustering. In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, 26 February–1 March 2002; pp. 685–694.
29. Spinosa, E.J.; de Leon, F.; de Carvalho, A.P.; Gama, J.A. OLINDDA: A Cluster based Approach for Detecting Novelty and Concept Drift in Data Streams. In Proceedings of the 2007 ACM Symposium on Applied Computing, Seoul, Korea, 11–15 March 2007; pp. 448–452.
30. Forestiero, A.; Pizzuti, C.; Spezzano, G. A single pass algorithm for clustering evolving data streams based on swarm intelligence. *Data Min. Knowl. Discov.* **2013**, *26*, 1–26. [CrossRef]
31. Forestiero, A.; Pizzuti, C.; Spezzano, G. FlockStream: A Bio-Inspired Algorithm for Clustering Evolving Data Streams. In Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence, Newark, NJ, USA, 2–4 November 2009.
32. Alswaitti, M.; Albughdadi, M.; Isa, N.A.M. Density based particle swarm optimization algorithm for data clustering. *Expert Syst. Appl.* **2018**, *91*, 170–186. [CrossRef]
33. Shamshirband, S.; Hadipoor, M.; Baghban, A.; Mosavi, A.; Bukor, J.; Varkonyi-Koczy, A.R. Developing ANFIS-PSO Model to Predict Mercury Emissions in Combustion Flue Gases. *Mathematics* **2019**. [CrossRef]
34. Kong, F.; Jiang, J.; Huang, Y. An Adaptive Multi-Swarm Competition Particle Swarm Optimizer for Large-Scale Optimization. *Mathematics* **2019**, *7*, 521. [CrossRef]
35. Fahy, C.; Yang, S. Finding and Tracking Multi-Density Clusters in Online Dynamic Data Streams. *IEEE Trans. Big Data* **2019**. [CrossRef]
36. Dorigo, M.; Di Caro, G. Ant colony optimization: a new meta-heuristic. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477.

37. Tu, D.Q.; Kayes, A.S.M.; Rahayu, W.; Nguyen, K. ISDI: A New Window based Framework for Integrating IoT Streaming Data from Multiple Sources. In Proceedings of the 33rd International Conference on Advanced Information Networking and Applications, AINA 2019, Matsue, Japan, 27–29 March 2019; pp. 498–511.
38. Krempl, G.; Žliobaite, I.; Brzeziński, D.; Hüllermeier, E.; Last, M.; Lemaire, V.; Noack, T.; Shaker, A.; Sievi, S.; Spiliopoulou, M.; et al. Open Challenges for Data Stream Mining Research. *SIGKDD Explor. Newsl.* **2014**, *16*, 1–10. [CrossRef]
39. Yin, C.; Xia, L.; Wang, J. Data Stream Clustering Algorithm Based on Bucket Density for Intrusion Detection. In *Advances in Computer Science and Ubiquitous Computing*; Park, J.J., Loia, V., Yi, G., Sung, Y., Eds.; Springer: Singapore, 2018; pp. 846–850.
40. Huang, G.; Zhang, Y.; Cao, J.; Steyn, M.; Taraporewalla, K. Online mining abnormal period patterns from multiple medical sensor data streams. *World Wide Web* **2014**, *17*, 569–587. [CrossRef]
41. Fahy, C.; Yang, S.; Gongora, M. Finding Multi-Density Clusters in non-stationary data streams using an Ant Colony with adaptive parameters. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastián, Spain, 5–8 June 2017; pp. 673–680.
42. Fahy, C.; Yang, S.; Gongora, M. Ant Colony Stream Clustering: A Fast Density Clustering Algorithm for Dynamic Data Streams. *IEEE Trans. Cybern.* **2019**, *49*, 2215–2228. [CrossRef]
43. Yang, X.S.; Hossein Gandomi, A. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
44. Opara, K.R.; Arabas, J. Differential Evolution: A survey of theoretical analyses. *Swarm Evol. Comput.* **2019**, *44*, 546–558. [CrossRef]
45. Bifet, A.; Holmes, G.; Kirkby, R.; Pfahringer, B. MOA: Massive Online Analysis. *J. Mach. Learn. Res.* **2010**, *11*, 1601–1604.
46. University of California. *KDD Cup 1999*; University of California: Irvine, CA, USA, 2007.
47. Rand, W.M. Objective Criteria for the Evaluation of Clustering Methods. *J. Am. Stat. Assoc.* **1971**, *66*, 846–850. [CrossRef]
48. Hedar, A.R.; Ibrahim, A.M.M.; Abdel-Hakim, A.E.; Sewisy, A.A. K-Means Cloning: Adaptive Spherical K-Means Clustering. *Algorithms* **2018**, *11*, doi:10.3390/a11100151. [CrossRef]
49. Wilcoxon, F. Individual comparisons by ranking methods. *Biom. Bull.* **1945**, *1*, 80–83. [CrossRef]
50. Caraffini, F. The Stochastic Optimisation Software (SOS) Platform. Available online: https://doi.org/10.5281/zenodo.3237024 (accessed on 1 December 2019).
51. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 dataset. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009.
52. Kononova, A.V.; Corne, D.W.; Wilde, P.D.; Shneer, V.; Caraffini, F. Structural bias in population based algorithms. *Inf. Sci.* **2015**, *298*, 468–490. [CrossRef]
53. Rosset, S.; Inger, A. KDD-cup 99: Knowledge Discovery in a Charitable Organization's Donor Database. *SIGKDD Explor. Newsl.* **2000**, *1*, 85–90, doi:10.1145/846183.846204. [CrossRef]
54. Caraffini, F.; Neri, F.; Gongora, M.; Passow, B. Re-sampling Search: A Seriously Simple Memetic Approach with a High Performance. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Workshop on Memetic Computing, Singapore, 16–19 April 2013; pp. 52–59.
55. Iacca, G.; Caraffini, F. Compact Optimization Algorithms with Re-Sampled Inheritance. In *Applications of Evolutionary Computation*; Kaufmann, P., Castillo, P.A., Eds.; Springer: Cham, Switzerland, 2019; pp. 523–534.
56. Caraffini, F.; Iacca, G.; Yaman, A. Improving (1+1) covariance matrix adaptation evolution strategy: A simple yet efficient approach. *AIP Conf. Proc.* **2019**, *2070*, 020004. [CrossRef]
57. Caraffini, F.; Neri, F.; Epitropakis, M. HyperSPAM: A study on hyper-heuristic coordination strategies in the continuous domain. *Inf. Sci.* **2019**, *477*, 186–202. [CrossRef]
58. Li, X.; Ye, Y.; Li, M.J.; Ng, M.K. On cluster tree for nested and multi-density data clustering. *Pattern Recognit.* **2010**, *43*, 3130–3143. [CrossRef]