Finding Multi-Density Clusters in Non-Stationary Data Streams Using an Ant Colony with Adaptive Parameters

Conor Fahy, Shengxiang Yang and Mario Gongora

Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University
The Gateway, Leicester LE1 9BH, United Kingdom
Email: {conor.fahy@dmu.ac.uk, syang@dmu.ac.uk, mgongora@dmu.ac.uk}

***Abstract*—Density based methods have been shown to be an effective approach for clustering non-stationary data streams. The number of clusters does not need to be known a priori and density methods are robust to noise and changes in the statistical properties of the data. However, most density approaches require sensitive, data dependent parameters. These parameters greatly affect the clustering performance and in a dynamic stream a good set of parameters at time t are not necessarily the best at time $t{+}1$. Furthermore, these parameters are global and so restrict the algorithm to finding clusters of the same density. In this paper, we propose a density based algorithm with adaptive parameters which are local to each discovered cluster. The algorithm, denoted Ant Colony Multi-Density Clustering (ACMDC), uses artificial ants to form nests in dense areas of the data. As the ants move between nests, their collective memory is stored in the form of pheromone trails. Clusters are identified as groups of similar nests. The proposed algorithm is evaluated across a number of synthetic data streams containing overlapping and embedded multi-density clusters. The performance of the algorithm is shown to be favourable to a leading density based stream-clustering algorithm despite requiring no tunable parameters.**

I. Introduction

Clustering dynamic data streams has additional constrains to static batch clustering. Streams can potentially be real time and infinite, so clustering needs to be performed quickly in a single pass of the data and the discovered clusters need to be summarised in a meaningful way. Furthermore, data streams can be non-stationary so the statistical properties of the stream can change over time in the form of concept drift (gradual change), concept shift (sudden change), or concept evolution where new classes appear in the stream. A stream-clustering algorithm needs to be able to adapt to these changes. Density based methods [8] have emerged as a suitable method of clustering streams. Density based methods identify clusters as areas of the feature space with high density separated by areas of low density. They are particularly suitable for streams because they can detect arbitrary shaped clusters, they are robust to noise, and the number of clusters does not have to be specified a priori.

The work by Aggarwal et al. [1] was one of the first to attempt density-based clustering for dynamic streams. The authors proposed a two stage approach to clustering: the online phase summarises the data and the offline phase clusters the summarised data. The concept of micro-clusters was introduced as a temporal extension to the cluster feature vector proposed in [15]. Micro-clusters are n-dimensional spheres that summarise a group of points which are close together in the feature space. This concept was later extended in DenStream [6]. Denstream uses a time dampened window model to assign greater importance to more recent data. A newly arriving point is assigned to its nearest micro-cluster in the on-line phase and the micro-clusters are grouped to form marco-clusters in the off-line phase using a concept of *density reachable* [8], which determines if two micro-clusters are connected. This method has been shown to achieve good results. However, it is computationally expensive and the offline phase must be executed frequently in order to discover changes in the stream. The main components of Denstream (mico-clusters, time-dampened window and the concept of density reachable) were extended in FlockStream [10], which merges both the online and offline steps into a single online step. Flockstream uses a swarm intelligence technique inspired by the flocking behaviour of birds [14] to group similar micro-clusters. A comprehensive review of density based stream clustering is given in [4] and a common shortcoming among all of these algorithms is their inability to detect clusters of varying densities.

This restriction means that the overall performance of the algorithm will degrade when the data contains multi-density clusters and it imposes restrictions on the *type* of clusters the algorithm can find. For example, embedded or overlapping clusters will not be discovered. This shortcoming also reduces the overall utility of the clustering solution. For example, consider a persistent, previously identified cluster. The density of the cluster could change, for example, if there were fewer samples of this concept in a particular window. The density has changed but the concept remains stationary. If the algorithm cannot recognise this new density, then the change would be misdiagnosed as concept drift.

The reason for this inability to detect varying densities is the use of global parameters for each cluster. For example, algorithms which employ micro-clusters as the summarisation mechanism [1], [6], [10] define the micro-cluster parameters globally. The ϵ-neighbourhood parameter defines the maximum radius for a micro-cluster and the $minPoints$ parameter defines how many points a micro-cluster should contain for it to be considered 'dense'. These parameters hold for every cluster and so restrict the algorithm to a single level of density.

Various solutions have been proposed for finding multi-density clusters in stationary batch data. Most are extensions of the DBSCAN algorithm [8]. These include MSDBSCAN [9], IS-DBSCAN [7], and DBSCAN-DLP [16]. A hierarchical algorithm based on the agglomerative k-means was proposed in [12]. However, these algorithms are not suitable for data streams because they require more than a single pass of the data or require prohibitively high computational time.

Recently, MuDi Stream [3] was proposed to overcome this problem of detecting clusters of varying density in a stream. MuDi is a density/grid based hybrid approach to clustering data streams. It uses the two phase online/offline approach. In the online phase, the stream is summarised using micro-clusters and these micro-clusters are partitioned into macro-clusters in the offline phase. A grid-based method is used to reduce the algorithm's time complexity, handle outliers and form new micro-clusters with varying radii. MuDi was shown to outperform other density based algorithms on datasets containing varying densities. However, it requires three sensitive data dependant parameters to be tuned: the outlier threshold, the density threshold and the grid granularity. This poses two problems; firstly, these parameters need to be appropriately tuned in order to generate a good clustering solution and secondly, in a dynamic stream, an appropriate set of parameters at time t might not be optimal at time $t+1$.

Motivated by the above analysis, we are proposing a density based stream clustering algorithm with adaptive parameters for finding multi-density clusters. Our proposed algorithm, called Ant Colony Multi-Density Clustering (ACMDC), merges both the online and offline phases into a single online phase and requires no tuneable parameters. ACMDC reads the stream in windows and the clustering process consists of two steps: initially, artificial ants representing data points collectively form nests in dense areas of the data. The ants generate pheromone trails between each nest and these trails form a similarity matrix between each established nest. In the second step, similar nests are grouped to form the final clustering solution.

The rest of the paper is outlined as follows. Our proposed algorithm is outlined in detail in Section II. An experimental study is presented in Section III and conclusions are offered in Section IV.

II. Proposed Algorithm

Density based clustering identifies clusters as areas of high density in the feature space separated by areas of low density. Points which are close (in terms of Euclidean distance) in the feature space are grouped into micro-clusters, where 'close' is determined by the ϵ-neighbourhood parameter which defines the maximum radius of a micro-cluster. The final clustering solution is the set of connected micro-clusters. A micro-cluster, containing N points $\{\vec{X_i}\}$, $i = \{1, ..., N\}$, is described using three components: N, LS and SS, where N is the number of data points in the micro-cluster, LS is the linear sum of the points (i.e., $\sum_{i=1}^{N} \vec{X_i}$), and SS is the squared sum of the points (i.e., $\sum_{i=1}^{N} \vec{X}_i^2$). LS and SS are n-dimensional vectors. From these three components we can obtain the centre c and radius r of the micro-cluster [1].

$$c = \frac{LS}{N} \tag{1}$$

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

Micro-clusters have the properties of additivity and incrementability. A point p can be absorbed into an existing micro-cluster m:

$$\begin{gathered} m.N \mathrel{+}= 1, \\ m.LS_i = m.LS_i + p_i, \\ m.SS_i = m.SS_i + p_i^2, \end{gathered} \tag{3}$$

where p_i is the i^{th} dimension of point p. Two existing micro-clusters m_i and m_j can merge into a new micro-cluster m_k, *iff* $radius(m_k) \le \epsilon$

$$m_k = (N_i + N_j, \vec{LS}_i + \vec{LS}_j, \vec{SS}_i + \vec{SS}_j) \tag{4}$$

Two micro-clusters m_i and m_k are said to be density reachable if:

$$dist(c_{m_i}, c_{m_k}) \le \epsilon, \tag{5}$$

where c_{m_i} and c_{m_k} are the centres of m_i and m_k, respectivey, and $dist(c_{m_i}, c_{m_k})$ is the Euclidean distance between c_{m_i} and c_{m_k}.

ACMDC employs the tumbling window model to read the stream. A tumbling window is a non-overlapping fixed size chunk of data. Once a window is read, the algorithm works in two steps to summarize, cluster and store the summary statistics off-line. In the first step, individual data points are treated as ants and the ants self organise and form 'nests' representing dense areas of the data. The similarity of each nest is 'remembered' and stored as pheromone trails between each nest. Nests are summarised as micro-clusters. In the second step, clusters are iteratively formed with adaptive parameters. Similar nests are clustered and merged, and the final solution returned by the algorithm is a set of partitioned dense areas of the data. The summarised clusters are stored off-line and a new tumbling window is evaluated.

An overview of the ACMDC framework is given in Algorithm 1. The following sections explain steps one and two in more detail.

Algorithm 1 ACMDC Overview

```
while <Stream> do
    Read tumbling window
        Create nests and pheromone trails (Algorithm 2)
        Find clusters (Algorithm 4)
        Store summary statistics off-line
```

A. Creating Nests

The step begins with $WindowSize$ data points (each point is treated as an ant) and an initial pre-set threshold $initSim$. Ants are iteratively assigned to nests representing dense areas of the data. The first ant creates the first nest and subsequent ants can either join an existing nest or form a new one. Each ant visits each nest in succession and evaluates a nest's suitability by comparing itself with $nComp$ randomly selected ants already in the nest, where $nComp$ is a preset parameter. An ant a estimates its similarity to nest k that has already n_k ants present in it (i.e., $k = \{k_1, k_2, ..., k_{n_k}\}$) as follows:

$$Sim(a,k) = \frac{\sum_{j=1}^{nComp} dist(a, k_j)}{nComp}, \tag{6}$$

where $nComp = n_k$ if $nComp > n_k$. Ant a joins the most similar nest provided its similarity score is equal to or below $initSim$. Otherwise, it creates a new nest. As each ant evaluates each nest, it 'remembers' its similarity with each. This collective memory is stored as pheromone trails between each nest in a Pheromone Matrix PM. When an ant joins (or forms a new) nest the ant updates the pheromone between the selected nest and all others. The pheromone trail between nest k and nest m is defined as the average of the similarities of all ants in nest k to nest m (Eq. (6)), as follows:

$$ph(k,m) = \frac{1}{n_k} \sum_{i=1}^{n_k} Sim(k_i, m), \tag{7}$$

where n_k is the number of ants in nest k and k_i is the i-th ant in nest k.

Once all ants have been assigned to nests, the contents of each nest are merged into a single micro-cluster (Eq. (4)). The pseudo-code for this step is outlined in Algorithm 2. At the end of this step, we have a set of n nests, each forming a single micro-cluster, and a pheromone matrix PM containing the similarity between each pair of nests, where $PM[k,m]$ is the similarity between nests k and m.

B. Finding Clusters

The previous step summarised $WindowSize$ points as a much smaller number of nests. However, the number of nests is still considerably larger than the number of natural clusters. In this step, clusters are discovered incrementally starting with the most dense, this allows the discovery of embedded and overlapping clusters. A new cluster C is seeded with the densest nest, formally:

$$seed = \max_{k \in Nests} (k.N) \tag{8}$$

Algorithm 2 Create Nests

Input: Tumbling window
Output: n Nests and Pheromone Matrix

```
for <each data point p> do
    if <nests> then
        Calculate its similarities to nests (Eq. (6))
        if <most suitable nest found> then
            Add $p$ to the most suitable nest
            Update pheromone trail (Eq. (7))
        else
            Create a new nest
            Add $p$ to the new nest
            Initialise pheromone trail
    else
        Create the first nest
        Add $p$ to the nest

for <each created nest> do
    create a new micro-cluster $m$
    add each point in the nest to $m$ (Eq. (3))

return Nests, Pheromone Matrix
```

Taking this nest as the seed we find its *closest neighbour*, i.e., the neighbour with the highest similarity score in the Pheromone Matrix PM:

$$neighbour = max(PM[:, seed]) \tag{9}$$

These two nests are added to cluster C and removed from the initial list of nests in order to prevent addition to future clusters. Similar nests are incrementally added to C.

In order to find similar nests, C requires two parameters; the ϵ-neighbourhood and a $threshold$. The ϵ-neighbourhood determines the minimum distance for two nests to be considered density reachable (Eq. (5)) and the threshold value determines if a nest added to a cluster C is a *border* nest. We define a border as a nest which is density reachable to C but has a density (N) of less than β times the density of the initial seed nest (Eq. (8)):

$$threshold = seed.N * \beta \tag{10}$$

For example, if the seed nest contained 100 points and $\beta = 0.1$, a border nest would contain 10 or fewer points.

At each new cluster, the ϵ-neighbourhood is initially 1.0 and adapts to the data using the distance between the seed nest and the seed's closest neighbour:

$$\lambda = PM[neighbour, seed] \tag{11}$$

Using λ, we update the ϵ parameter for the ϵ-neighbourhood as follows:

$$\epsilon = \epsilon - \lambda \tag{12}$$

The pseudo-code for seeding a new cluster is given in Algorithm 3.

```
Algorithm 3 Initialise Cluster
Input: Nests, Pheromone Matrix
Output: Cluster C
Create new cluster C
Find densest nest $n$ in Nests (Eq. (8))
Find $n$'s closest neighbour $neighbour$ (Eq. (9))
$\lambda :=$ pheromone between $n$ and $neighbour$ (Eq. (11))
$\epsilon := \epsilon - \lambda$
$initialDensity := n.N$
$Threshold := initialDensity * \beta$
add $n$ and $neighbour$ to $C$
delete $n$ and $neighbour$ from Nests
delete $n$ and $neighbour$ from Pheromone Matrix
return C
```

Once a new cluster C has been seeded, a boolean $newSeed$ is set to false and similar nests are added to C. Each nest is tested in succession to see if it is density reachable with a nest in C (Eq. (5)). If a nest is density reachable it is removed from the list of nests and added to C. If, at the end of an iteration, a nest has been added that is not a border point, C can expand its ϵ-neighbourhood (Eq. (12)) and perform another iteration with the increased ϵ. This continues until no further nests have been added, *or* if only border nests have been added. At this stopping condition, the nests in cluster C attempt to merge (Eq. (4)), C is added to the final list of clusters and the boolean $newSeed$ is set to true and the process repeats until all nests have been assigned to a cluster.

Due to expanding value of ϵ, outliers and noise points will be assigned to their own cluster because clusters expand until all nests are assigned. To overcome this we remove clusters where the number of nests in the cluster is equal to the number of data points in the cluster, i.e., every micro-cluster contains only one point. Micro-clusters containing only a single point are not similar enough with any other points to merge and can be considered to lie in low density areas of the feature space.

Pseudo-code for finding clusters is presented in Algorithm 4.

```
Algorithm 4 Find Clusters
Input: Nests, Pheromone Matrix
Output: Set of $k$ clusters
while <Nests> do
    if <newSeed = true> then
        C := Initialise cluster (Algorithm 3)
        newSeed := false
    for <Each nest n> do
        if <n is density reachable to C> then
            add $n$ to $C$
            delete $n$ from Nests
            delete $n$ from Pheromone Matrix
            if <n is not a border nest> then
                expand := true
    if <expand> then
        $\epsilon := \epsilon - \lambda$
    else
        merge micro-clusters in C (Eq. (4))
        add C to clusters
        newSeed := true

Remove outliers
return clusters
```

III. Experimental Study

To evaluate ACMDC we test its performance across 5 datasets using 2 well-known metrics. The performance of the algorithm is compared with that of DenStream [6]. DenStream is a leading density based stream clustering algorithm that also uses micro-clusters to summarise and cluster data. DenStream has six parameters and these have been manually tuned for each dataset to give the best performance. DenStream is evaluated using the Massive On-line Analysis [5] open source software.

A. Metrics

Two common metrics are used to evaluate the performance of the clustering solution; Purity and the Rand Index [13]. We know the labels of the data so the metrics are computed using the ground truth. Purity measures the frequency of the most common category occurring in each cluster and how homogeneous each cluster is. A score of 1 indicates the cluster contains only instances of the same class.

The purity of a clustering solution S, with N discovered clusters, $S = \{C_1, ..., C_N\}$ to be evaluated against a set of categories $L = \{L_1, ..., L_K\}$ is calculated by taking the average of the maximal precision value for each discovered cluster. The precision of a cluster C_i is defined as follows [2]:

$$precision(C_i) = \frac{|C_i \cap L_i|}{|C_i|} \tag{13}$$

where L_i is the most frequently occurring category in C_i.

The overall purity of the clustering solution is the average of the precision values of clusters within the solution, which is measured as follows:

$$P = \frac{1}{N}\sum_{i=1}^{N} precision(C_i) \tag{14}$$

The Rand Index is a measure of agreement between two clustering solutions; the solution provided by the algorithm and the *true* clustering solution known from the ground truth. It measures how many of the clustering decisions are correct, based on the numbers of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), as follows:

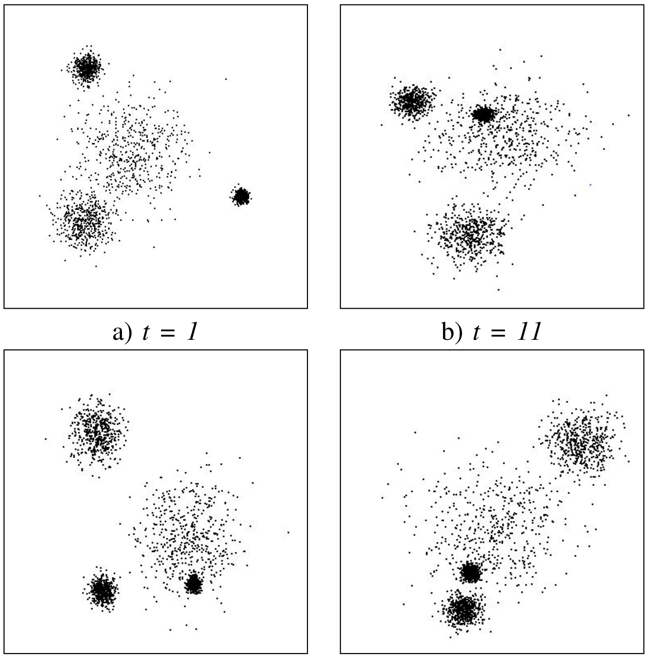
$$R = \frac{TP + TN}{TP + FP + TN + FN} \tag{15}$$

B. Datasets

Synthetic data streams were generated in order to test ACMDC in a streaming environment containing concept drift, concept evolution and multi-density clusters which can overlap and embed in other clusters. The points in each synthetic

TABLE I: Description of datasets used in experiments

Name	Dim.	Clusters	Examples	Drift
2D	2	3-5	100,000	1,000
4D	4	3-5	100,000	5,000
8D	8	5-10	150,000	7,500
16D	16	10-20	150,000	7,500
32D	32	10-20	200,000	15,000



a) *t = 1*   b) *t = 11*

c) *t = 30*   d) *t = 60*

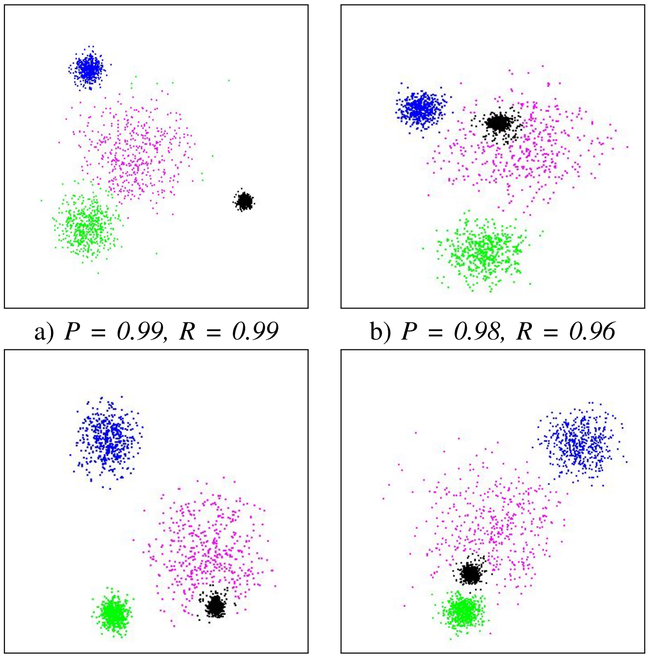
Fig. 1: Stream progression over 60 windows, containing 4 non-stationary clusters of varying densities.

cluster are drawn from a series of Gaussian distributions with each cluster having a different standard deviation. At regular intervals concept drift is simulated by changing the mean and variance of each cluster and concept evolution is simulated by adding or removing a cluster. Datasets were generated with dimensions ranging from 2 to 32, the number of natural clusters ranging from 3 to 20 and a varying number of samples and drift intervals.

The full details of each dataset are presented in Table I and sample windows containing 2 dimensional data are presented in Fig. 1. These illustrative examples were selected to display multi-density clusters (Fig. 1(a)), overlapping clusters (Fig. 1(d)), and embedded clusters (Fig. 1(b, c and d)).

C. Results

The clustering solution for the illustrative windows presented in Fig. 1 can be seen in Fig. 2. Inspecting the clusters visually, it can be seen that all four clusters are discovered despite the clusters not being clearly separated and having varying densities. More formally, the purity and Rand Index scores are also presented. It can be observed that the algorithm returns high purity clusters which are close to the true structure of the data.



a) *P = 0.99, R = 0.99*   b) *P = 0.98, R = 0.96*

c) *P = 0.99, R = 0.98*   d) *P = 0.97, R = 0.96*

Fig. 2: Discovered clusters in 2D windows.

TABLE II: Results using purity (P) and Rand Index (R)

Data	ACMDC		DenStream		
	P	R	P	R	ϵ
2D	$\mathbf{0.96}$	$\mathbf{0.97}$	0.94	0.92	(0.025)
4D	$\mathbf{0.99}$	$\mathbf{0.99}$	$\mathbf{0.99}$	0.94	(0.008)
8D	$\mathbf{0.98}$	$\mathbf{0.98}$	$\mathbf{0.98}$	0.82	(0.01)
16D	$\mathbf{0.95}$	$\mathbf{0.96}$	$\mathbf{0.95}$	0.73	(0.15)
32D	0.95	$\mathbf{0.90}$	$\mathbf{0.97}$	0.22	(0.055)
Avg.	$\mathbf{0.96}$	$\mathbf{0.96}$	$\mathbf{0.96}$	0.72	

The performance of the algorithm across each dataset is presented in Table II. These results are the average performance of each window over the entire stream. Comparative results with DenStream are also displayed. There is an extra column describing the DenStream results; the ϵ parameter. This is one of six tunable, data-dependent parameters. Each was tuned to give the best results, but because the ϵ-neighbourhood is the most sensitive it has been displayed. Over the five datasets, the purity of each algorithm is comparable but ACMDC performs favourably on the Rand Index metric. This is highlighted in Fig. 3, which displays the performance of each algorithm over the entire 32D stream. The stream contains 200,000 samples, the window size for ACMDC and evaluation frequency for DenStream are 5,000 giving a total of 40 windows. The purity on each is comparable but ACMD considerably outperforms DenStream on the Rand Index.

Figure 4 displays the performance of each algorithm over the entire 2D dataset. This dataset contains fewer samples

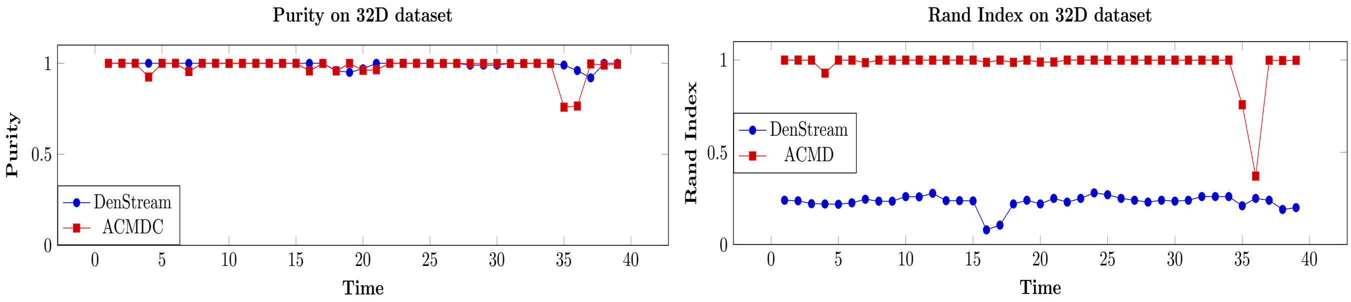


Fig. 3: Comparative performance over 32D stream progression

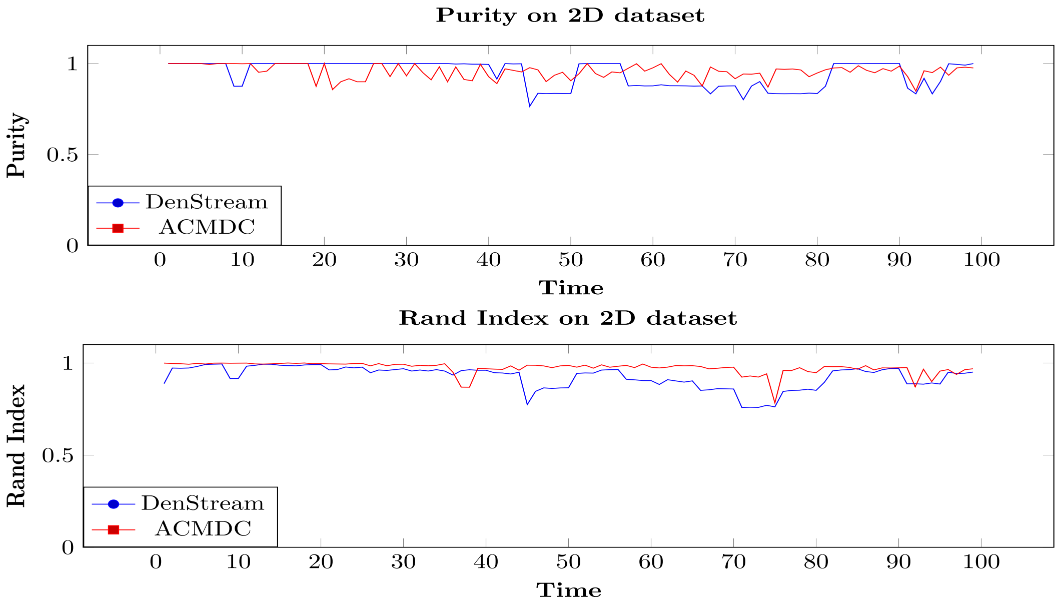


Fig. 4: Comparative performance over 2D stream progression

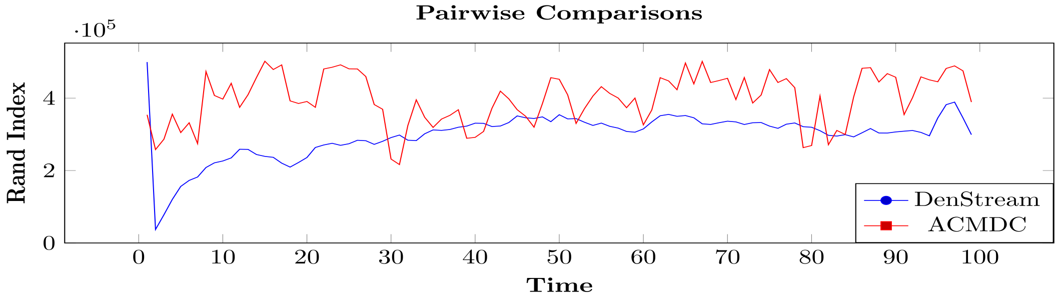


Fig. 5: Number of pairwise calculations performed on 2D stream progression

but shorter windows and more frequent evaluations (1,000) resulting in 100 windows. Both algorithms are comparable across both metrics with ACMD performing slightly better despite requiring no tunable parameters.

Figure 5 gives an indication of the speed requirements of ACMDC. In order to measure the speed requirement we count the number of pair-wise Euclidean distance comparisons performed by each algorithm. We assume that a higher number of calculations implies a greater time requirement. This will be especially true in high dimensional data. The results from DenStream are taken using an ϵ value of 0.25 and an evaluation frequency and horizon of $1,000$. ACMDC uses a tumbling window of size $1,000$. The overall average for each evaluation comes to roughly 297,000 for DenStream and roughly 397,000 for ACMDC. A speed decrease of roughly 25% on average.

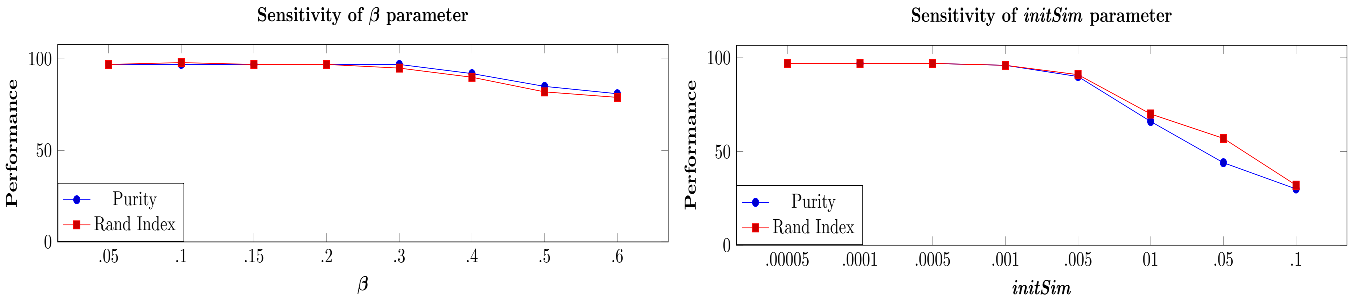


Fig. 6: Sensitivity of program variables on 2D data stream

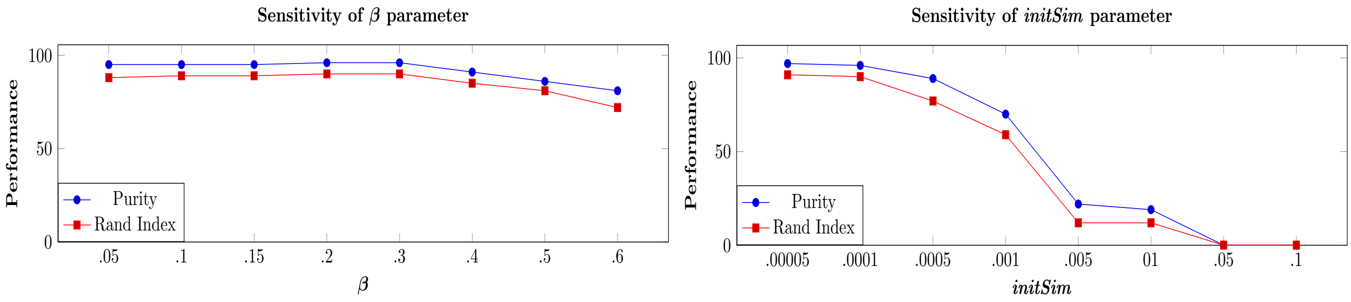


Fig. 7: Sensitivity of program variables on 32D data stream

D. Sensitivity Analysis

ACMDC requires three program variables. The first, $nComp$ which is used to determine the number of comparisons an ant will make with a nest, is set to $WindowSize{*}0.1$. This is taken from the results in [11] which uses a similar sampling method. The remaining two variables are analysed for sensitivity across two streams; the 2D stream and the 32D stream. Figs. 6 and 7 show the sensitivity of β (which determine border nests during cluster formation) and $initSim$ (which defines the minimal nest similarity during nest formation).

The results on both streams are similar. β returns similar scores for each value but the performance of the algorithm begins to degrade at roughly 0.3. $initSim$ is much more sensitive, it can be seen that too large a value completely degrades the algorithm's performance and a very small value is more appropriate. Based on this analysis, for all experiments presented in this study, β is set as 0.2 and $initSim$ is set to 0.0001.

IV. Conclusion

In this paper we have proposed Ant Colony Multi Density Clustering (ACMDC), a density based method for clustering data streams. ACMDC has adaptive parameters, local to each discovered cluster, which allow it to find clusters of varying density. The algorithm uses tumbling windows to read the data and the windows are clustered online. The clustering process consists of two steps: first, ants representing each data point form nests with similar ants and second, similar nests are grouped to form clusters. Discovered clusters are summarised using micro-clusters and the summary statistics for each window are stored offline.

ACMDC was evaluated across five synthetic data streams. The streams ranged from 2 to 32 dimensions describing 3 to 20 clusters of varying densities which can overlap, can be nested and are not clearly separated. Streams exhibit concept drift, concept shift and concept evolution. ACMDC was evaluated on two well-known metrics; purity and Rand Index, and was compared with DenStream, a leading density based stream clustering algorithm. It was shown that ACDMC outperforms DenStream on these datasets based on these two metrics. The purity levels are comparable but the Rand Index score, which reflects the true topology of the data, is much higher for ACDMC. The Rand Index for DenStream decreases as the number of clusters in the stream increases. This is because there are a greater number of densities in the data coupled with the fact that a higher number of clusters make it more probable that they will embed and overlap. DenStream is restricted to a single density and is unable to find nested clusters and so the performance degrades.

ACDMC finds clusters incrementally, starting with the most dense and finishing with the least dense. This allows for the discovery of nested and overlapping clusters and, ultimately, clusters of any density. The drawback of this approach is that outliers are grouped into their own (very sparse) cluster. ACMDC deals with outliers by removing any cluster that only contains micro-clusters which only describe a single point. Outliers are usually too dissimilar to other points to be described by a single micro-cluster and micro-clusters describing outliers are too dissimilar to merge. This approach fails if, as is realistic, two or more noise points lie in the same low dense area of the feature space. Future work will aim to address this and better deal with noise.

The time requirements of ACMDC was compared with that of DenStream. In order to measure this we count the number of pair-wise Euclidean distance comparisons performed by each algorithm. We assume that a higher number of calculations equates to a greater time requirement, especially with high dimensional data. Although comparable, ACMDC was shown to require a greater number of comparisons than DenStream, roughly 25% extra on average over the 2D stream.These extra distance calculations are the cost of the adaptive parameters.

ACMD requires no tune-able parameters, except for the window size, but it requires three hard-coded parameters. The $initSim$ parameter determines the minimum similarity in the first step of the algorithm. This has been shown to be sensitive to higher values (greater than 0.001) but works efficiently with much smaller values. Intuitively, this seems correct as it is more desirable for the ϵ-neighbourhood of a cluster to expand in smaller increments. The second parameter, β, determines the density threshold. This has been shown to be less sensitive than $initSim$, but, similar to $initSim$, larger values return a poorer performance. We define this as 0.2, which means that nests which are (at most) 20% as dense as the seed nest are to be considered as border nests when forming clusters.

In the future, we aim to improve the algorithm's handling of outliers and evaluate it more rigorously on a larger number of datasets across a wider range of metrics. We also aim to improve the utility of the discovered clusters. When dealing with dynamic data-streams, *change* is one of the most interesting aspects: detecting and adapting to change, as well as *describing* the change. In future work, we aim to better structure the summary statistics so as to provide a change detection mechanism based on clusters discovered in previous windows.

ACKNOWLEDGMENT

This work was funded by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1.

REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," Proc. of the 29th Int. Conf. on Very Large Data Bases, vol. 29, pp. 81–92, 2003.

[2] E. Amig, J. Gonzalo, J. Artiles, J. and F. Vedejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," Information Retrieval, vol. 12, no. 4, pp. 461–486, 2009.

[3] A. Amini, et al. "MuDi-Stream: A multi density clustering algorithm for evolving data stream," Journal of Network and Computer Applications, vol. 59, pp. 370–385, 2016.

[4] A. Amini, et al. "On density-based data streams clustering algorithms: A survey," Journal of Computer Science and Technology, vol. 29,no. 1, pp. 116–141, 2014.

[5] A. Bifet, G. Holmes, R. Kirby, and B. Pfahringer, "MOA: Massive online analysis," Journal of Machine Learning Research, vol. 11, pp. 1601–1604, 2010.

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

[7] C. Carmelo, et al. "Enhancing density-based clustering: Parameter reduction and outlier detection" Information Systems, vol. 38, no. 3, pp. 317–330, 2013.

[8] M. Ester, H. P. Kriegel, J. Sander, and X. Xu,"A density-based algorithm for discovering clusters in large spatial databases with noise," Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining, pp. 226–231, 1996.

[9] G. Esfandani, H. Abolhassani. "MSDBSCAN: multi-density scale-independent clustering algorithm based on DBSCAN," Proc. Int. Conf. on Advanced Data Mining and Applications, 2010.

[10] A. Forestiero, C. Pizzuti, and G. Spezzano, "A single pass algorithm for clustering evolving data streams based on swarm intelligence," Data Mining and Knowledge Discovery, vol. 26, no. 1, pp. 1–26, Nov. 2011.

[11] N. Labroche, N. Monmarche, and G. Venturini, "AntClust: ant clustering and web usage mining," Proceedings of the 2003 Genetic and Evolutionary Computation Conference, pp. 25–36, 2003.

[12] X. Li et al. "On cluster tree for nested and multi-density data clustering," Pattern Recognition, vol. 43, no. 9, pp. 3130–3143, 2010.

[13] W. M. Rand, "Objective criteria for the evaluation of clustering methods," Journal of the American Statistical Association, vol. 66, no. 336, pp. 846, Dec. 1971.

[14] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," ACM SIGGRAPH Computer Graphics, vol. 21, no. 4, pp. 25–34, Aug. 1987.

[15] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: A new data clustering algorithm and its applications," Data Mining and Knowledge Discovery, vol. 1, no. 2, pp. 141–182, 1997.

[16] Z. Xiong, et al. "Multi-density dbscan algorithm based on density levels partitioning." Journal of Information and Computational Science, vol. 9, no. 10, pp. 2739–2749, 2012.