Dynamic Stream Clustering Using Ants

Conor Fahy and Shengxiang Yang

Abstract
Data stream mining is the process of extracting knowledge from continuous sequences of data. It differs from conventional data mining in that a stream is potentially unbounded, data points arrive online and each data point can be examined only once. Furthermore, in non-stationary environments the statistical properties of the data can change over time. This paper presents a bio-inspired approach to clustering non-stationary data streams. The proposed algorithm, Ant-Colony Stream Clustering (ACSC), is based on the concept of artificial ants which identify clusters as nests of micro-clusters in dense areas of the data. Micro-clusters are N-dimensional spheres with a maximum radius ε. In ACSC the ε-neighbourhood, crucial in density clustering, is adaptive and doesn't require expert, data-dependent tuning. The algorithm uses the sliding window model and summary statistics for each window are stored offline. Experimental results over real and synthetic datasets show that the clustering quality of ACSC is comparable or favourable to leading stream-clustering algorithms while requiring fewer parameters and considerably less computation.

1 Introduction

Data stream clustering poses additional challenges to traditional clustering. The nature of an evolving stream implies that the number of clusters cannot be known *a-priori* as not every class will always be present in the stream. Non-stationary streams will likely exhibit *concept drift* where the statistical properties of the target classes

Conor Fahy
Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, The Gateway, Leicester LE1 9BH, U.K. e-mail: conor.fahy@dmu.ac.uk

Shengxiang Yang
Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, The Gateway, Leicester LE1 9BH, U.K. e-mail: syang@dmu.ac.uk

will change over time in unforeseen ways. Stream clustering algorithms must be able to quickly cope with this change. A stream can be potentially infinite but only a limited amount of memory is available so it is not feasible to ‘remember’ all the data scanned in the past. Furthermore, when dealing with a continuous sequence of information, it is only possible to examine the data once. The work by Aggarwa *et al.* [2] was the first to address this constraint of being unable to revisit evolving data. The authors suggest that a stream clustering algorithm should consist of two components: an online component that summarises the data and an offline component that clusters the summarised data. Their algorithm, denoted *CluStream*, is based on the *K-means* approach, which imposes limitations on the algorithm’s performance; it is not robust to outliers, it must assume a certain number of clusters are present, and it can only identify spherical clusters.

Density based clustering [6] overcomes the aforementioned limitations. Density based clustering detects areas of high density in the feature space. It can detect arbitrarily-shaped clusters and does not need to know the number of clusters a-priori. Density based clustering has been extended to the online and offline framework [4, 16, 17], but it has been observed [7, 17] that the offline component of the two phase scheme is very computationally demanding as a clustering algorithm must to be applied to the summarised data. The offline clustering is performed when a clustering request is made by a user. This poses a further problem as to discover changes in the data stream the offline phase must be executed frequently. To overcome this, both steps were merged in *Flockstream*[7]. FlockStream is a multi-agent system that uses a self-organising strategy to group similar points and is inspired by the flocking behaviour of birds [14]. Flockstream computes significantly less distance evaluations compared to DenStream while still achieving similar levels of cluster purity.

Although there is a large body of research on bio-inspired clustering algorithms, few are designed to cope with data streams. In [12], a method for clustering streams with ants was proposed. Each ant represents a datum and each has a colonial odour which identifies which cluster they belong to. Ants move from cluster to cluster along pheromone trails to find their most suitable cluster (‘most suitable’ being the closest in terms of Euclidean distance). The algorithm computes the clusters using *K-means*, which, again, is limiting in both the number and shape of the clusters that it can potentially find. A variation on the Leader Ant algorithm was proposed in [11] as an accurate and inexpensive agglomerative clustering algorithm. Other ant clustering algorithms are based on the ‘pick-and-drop’ model proposed in [5]. In this paradigm each datum is represented in a 2D space, scattered randomly initially. Ants move around the 2D space and decide probabilistically to pick and drop data based on the local density. The macro-clustering process is an emergent property from these local rules. This basic model has been extended [8, 9, 10] with good results. However the algorithm requires many iterations to organise the data and furthermore, the sorted data is just a spatial embedding and a further processing step is required. There are also a large number of studies on clustering using Ant Colony Optimisation [1], where the clustering problem is framed as an optimisation problem [19].

In this paper, we propose a bio-inspired approach, denoted Ant-Colony Stream Clustering (ACSC), to density based clustering for data streams. Clusters are identified as 'nests' of density-reachable micro-clusters. Ants probabilistically pick up micro-clusters based on local density and similarity to neighbouring micro-clusters. The ants move along pheromone trails to other dense areas of the data and probabilistically decide to drop the micro-cluster, creating pure clusters with high precision and recall scores.

To the best of our knowledge this is the first time an ant-based approach has been used to tackle the problem of density based stream clustering. The advantage of using an Ant Colony is that the exhaustive search for the nearest neighbour of each point is replaced with stochastic sampling from 'nests'. Ants cluster the data in parallel and communicate through local pheromone trails. These properties of sampling, locality and asynchronism suggest that the algorithm is scalable to larger datasets. To our knowledge it is also the first proposal of an adaptive ε- neighbourhood.

The rest of this paper is organized as follows: Section 2 presents our proposed ACSC in detail. Section 3 presents our experimental study based on several real and synthetic data streams, and Section 4 concludes this paper.

2 Proposed Ant-Colony Stream Clustering (ACSC) Algorithm

Dense areas in the feature space are identified and described using micro-clusters. A micro-cluster is a set of individual points which are close together or within a certain radius, the ε-neighbourhood determines this radius. A micro-cluster is an n-dimensional sphere with a radius r and centre p where $r \leq \varepsilon$. A micro-cluster is described using three components: N, the number of data points in the micro-cluster, LS, the linear sum of the data points, and SS, the sum of squared data points. From these, the radius and centre of the micro-cluster can be determined [2]. The structures LS and SS are n-dimensional arrays and the vectors have the important properties of additivity and increment-ability which allows micro-clusters to merge. Two micro-clusters m_i and m_j can merge into a single micro-cluster m_k if:

$$radius(m_k) \leq \varepsilon \tag{1}$$

Micro-clusters m_i and m_j are said to be *density reachable* if:

$$dist(cen_{m_i}, cen_{m_j}) \leq \varepsilon \tag{2}$$

The final clusters identified by ACSC are partitioned sets of density reachable micro-clusters. ACSC works in three steps: 1), the ε-neighbourhood is adaptively identified at each new window, 2) dense areas of the feature space are identified and finally, 3) these dense areas are clustered and their summary statistics are stored off-line.

Algorithm 1 ACSC Pseudocode

```
 1: while <Stream> do
 2:     simulate nMeetings and calculate $\varepsilon$
 3:     for <each data point> do
 4:         create microcluster from point
 5:         if <nests> then
 6:             find best nest
 7:             add microcluster to nest
 8:             if <no suitable nest> then
 9:                 create new nest
10:                 add microcluster to nest
11:         else
12:             create nest
13:             add microcluster to nest
14:     create ants
15:     assign ant to each nest
16:     while <!Stop Condition> do
17:         for <each ant> do
18:             if <!Sleeping> then
19:                 probabilistically pick-up micro-cluster Eq. (4)
20:                 if <Carrying> then
21:                     move to next nest based on pheremone trail
22:                     probabilistically drop microcluster Eq. (4)
23:                 update pheremone trail
24:                 update sleepCounter
25:     store summary statistics of current window
```

2.1 *Finding the ε-neighbourhood*

The ε parameter is crucial in density based clustering. It determines the maximum radius of a micro-cluster. If its too large then impure micro-clusters will form, too low and no micro-clusters will form at all. It is data specific and in non-stationary streams the density of areas in the feature space can change, so a good value for ε in one window might not be a good value in another window. In ACSC this value is adaptive. ACSC employs the sliding window model when dealing with data streams so at each iteration a fixed size chunk of data is considered. The process for finding ε at each window is based on the algorithm presented in [11]. ε is calculated as the mean value of n Euclidean distance measures, $dist(i,j) \in [0,1]$, between n randomly chosen i and j data points in the current window. In all experiments presented $n = WindowSize * .2$ where $WindowSize$ is an input parameter.

2.2 *Identifying Dense Areas in the Feature Space*

Each data point in the current window is first converted to a micro-cluster. This micro-cluster will have a radius of 0 and its $center = LS = SS =$ the original data point. So initially there will be *WindowSize* number of micro-clusters. The micro-clusters are then assigned, one by one, to bins representing dense areas of the data. In the biological metaphor these bins are nests and the micro-clusters are ants. Initially there are no nests, so the first ant creates the first nest. Subsequent ants can either a) merge with existing ants, b) join an existing nest or, c) create a new nest. Ants visit each nest in turn and determine the nest's suitability by simulating *nComp* comparisons with randomly chosen ants from each nest (in all experiments presented $nComp = WindowSize * .1$). During these comparisons, ant a's similarity with nest k is estimated by using:

$$Sim(a,k) = \frac{\sum_{j=1}^{n} dist(a,k_j)}{n} \tag{3}$$

If, during the calculation, the current ant can merge (Eq. (1)) with an ant already present in the nest then both ants will merge and the step is over. Otherwise, the similarity is calculated for each nest and the ant joins the most suitable nest *provided* the similarity is below ε, if not the ant creates a new nest. During this trip, as the ant tests its similarity with each nest, it keeps a record of each nest's suitability. Upon joining a nest it uses these scores to update the pheromone trail between the selected nest and its neighbouring nests. The pheromone trail to each neighbouring nest is a rolling average updated whenever a new ant joins the nest.

2.3 Inter-Nest Sorting

The previous step identified the dense areas of the feature space but often the nests identified are rough and too many. Inter nest sorting is performed in this step. Sorting ants are created and one is assigned to each nest. Each sorting ant decides probabilistically to pick-up a micro-cluster from its nest based on local density and similarity.

A micro-cluster m is selected randomly in nest k and compared to *nComp* micro-clusters in the same nest. The Euclidean distance from the centre of m to each of these micro-clusters is calculated and if the two micro-clusters are density reachable (Eq. 2) then a score s is incremented. The probability of picking-up the micro-cluster is given by:

$$P_{pick} = 1 - \frac{s}{nComp} \tag{4}$$

It is important to note that if the number of micro-clusters in nest k is fewer than *nComp* then only that amount of comparisons are made, however P_{pick} is still calculated using *nComp*, this ensures a higher probability of a pick-up in nests containing fewer micro-clusters. This leads to the dissolution of smaller nests. If the sorting ant successfully picks a micro-cluster it then moves to a neighbouring nest based on the pheromone trails described in the previous section. The ant deterministically selects the most promising trail. In the new nest the ant decides probabilistically to drop the micro-cluster. The calculation for dropping is the inverse of Eq. (4). If successful, the micro-cluster is dropped and the ant returns to its nest, otherwise the micro-cluster remains in its original nest. The pheromone trail between the original nest and the selected nest is updated with the latest similarity score Eq. (3).

Each sorting ant is native to its own nest and continues sorting until a) the nest is empty(all micro-clusters have been moved to another nest) or b) the sorting ant is ‘asleep’. Each sorting ant has a *sleepCounter*. If a sorting ant has an unsuccessful attempt at picking or dropping this counter is incremented. It is reset to zero after a successful attempt. When the sleepCounter reaches sleepMax (3 in all experiments presented), the sorting ant ‘sleeps’ and the nest is considered to be sorted.

This step ‘purifies’ each nest and encourages the dissolution of smaller nests which can be incorporated into larger similar nests. Outliers are identified as nests containing only one micro-cluster. The final clustering solution is represented by the set of non-empty nests. Each nest contains a set of density reachable micro-clusters which summarise the partitioned dense areas of the data. These summary statistics are stored off-line and the next window in the stream is considered.

3 Experimental Setup

The performance of ACSC is evaluated across five datasets using three metrics and is compared with two density stream clustering algorithms: DenStream [4] and CluStream [2]. Both DenStream and CluStream are evaluated using the Massive Online Analysis (MOA) [22] open source software.

3.1 Metrics

Three well known metrics are used to evaluate the performance of ACSC: Purity, F-Measure [20] and The Rand Index [21]. The datasets used are all labelled so clusters are measured with respect to the ground truth. *Purity* measures the purity of the clusters obtained by assigning each cluster to the most frequent class appearing in the cluster and summing the instances of this class. The *F-Measure* (or *F-Score* or *F1-Score*) is the harmonic mean of the precision and recall scores. In the following, CR_i represents the clusters identified by the algorithm. In every identified cluster, v_{ij} represents the number of data points identified in each class.

$$precision_{CR_i} = \frac{max(V_{i1},...,V_{im})}{\sum_{i=1}^{m} V_{ij}} \tag{5}$$

$$recall_{CR_i} = \frac{max(V_{i1},...,V_{im})}{\sum_{i=1}^{n} V_{ij}} \tag{6}$$

$$Score = 2 * \frac{precision * recall}{precision + recall} \tag{7}$$

Purity and The F-Measure can now be defined in terms of the total number of clusters, *n*, identified by the algorithm:

$$Purity = \frac{\sum_{i}^{n} precision_{CR_i}}{n} \tag{8}$$

$$F1 = \frac{\sum_{i}^{n} Score}{n} \tag{9}$$

The *Rand Index* is a measure of agreement between two partitions; the solution obtained by the algorithm and the *ideal* solution known from the ground truth. The Rand Index penalises both false positive and false negative decisions during clustering. Simply, it measures the percentage of decisions that are correct.

$$RandIndex = \frac{TruePositives + TrueNegatives}{TruePostives + FalsePositives + TrueNegatives + FalseNegatives} \tag{10}$$

3.2 Datasets

ACSC is evaluated across five non-stationary datasets; four synthetic and one real. The real dataset is the network Intrusion dataset used in the 1999 KDD competition [1]. It contains data consisting of 7 weeks of network-based intrusions inserted in the normal data. It is composed of two classes; one class contains 'normal' network connections and the other contains non-stationary malicious network intrusions. This dataset was selected as it contains substantial drift, the 'malicious' class is composed of 23 different types of attack. These attacks are described by 39 features. The synthetic datasets are taken from the non-stationary data archive [2] used in [15]. These four datasets were selected in order to test varying number of classes present in the data. 1CDT consists of one stationary class and one non-stationary class. 2CHT consists of two non-stationary classes. 4CR consists of four non-stationary classes, 4CE1CF consists of five classes; four of which are non-stationary. The details of each datatset used are presented in table 1.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2] https://sites.google.com/site/nonstationaryarchive/

	Classes	Features	Examples	Drift Interval	Type
1*CDT*	2	2	16,000	400	Synthetic
2*CHT*	2	2	16,000	400	Synthetic
4*CR*	4	2	144,400	400	Synthetic
4*CE*1*CF*	5	2	173,000	750	Synthetic
NetworkIntrusion	2	39	494,000	unknown	Real

Table 1: Description of datasets used in experiments

3.3 Clustering Quality Evaluation

Figures 1-3 show the performance of of ACSC as the stream progresses over time.

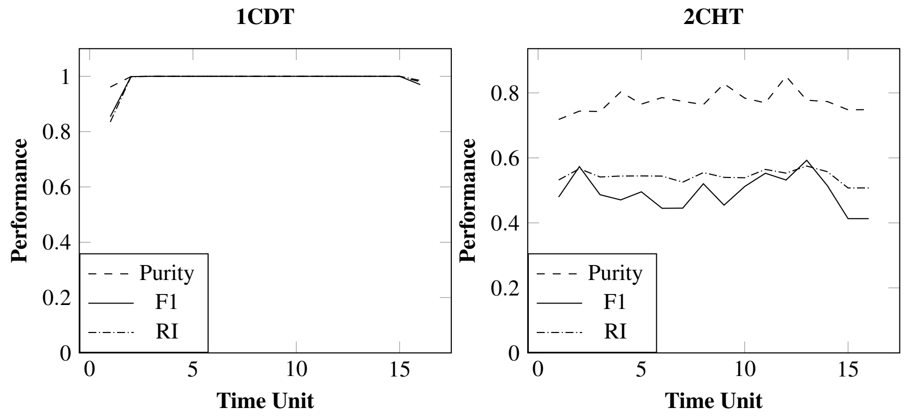

Fig. 1: Performance of ACSC on 1CDT and 2CHT over 16 windows

Table 2 presents a comparison of ACSC with DenStream over 100,000 data points. 1,000 points are considered in each window and are averaged in units of 10 for display purposes.

Table 3 presents the mean values of each algorithm across the entire stream. DenStream and CluStream are deterministic but ACSC is stochastic so the results displayed are the average, along with the standard deviation, over five runs.

3.4 Sensitivity of Window Size and Time Complexity

To evaluate window sensitivity ACSC was tested across 5 different window sizes: 500, 1000, 1500, 2000, and 5000. The mean purity, F-Measure and Rand Index are

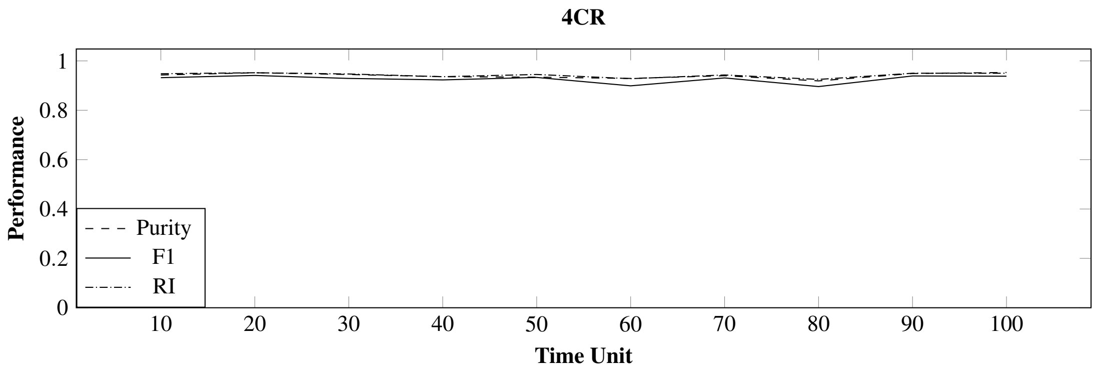

Fig. 2: Performance of ACSC on 4CR dataset over 100 windows of size 1,000

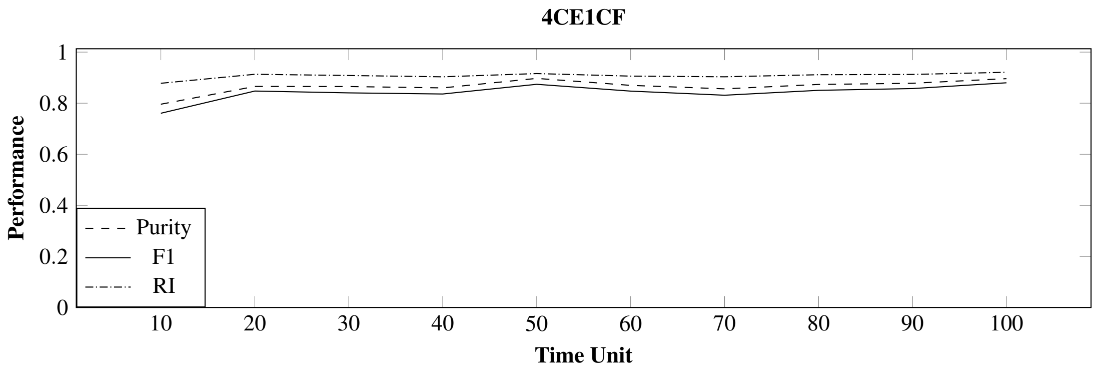

Fig. 3: Performance of ACSC on 4CE1CF over 100 windows of size 1,000

calculated across each window size in the stream and, for visualisation purposes, a new metric ‘score’ is introduced. Score is simply the average of all three metrics.

To understand how ACSC scales to different sizes the number of pairwise distance calculations is reported. These comparisons are the Euclidean distance comparisons between two micro-clusters. Intuitively, the greater the number of comparisons the longer the algorithm takes. Results across 2 synthetic and 1 real datasets are presented in figures 4,5 and 6.

In figure 7, the mean number of calculations ACSC performs on the Network Intrusion dataset is presented alongside the number performed by DenStream. It has already been observed [4] that DenStream performs considerably fewer calculations than CluStream. These results are based on a window size of 1,000 and using the default settings on the DenStream algorithm in MOA.

Time	**Purity**		**F1**		**R.Index**	
	ACSC	*DenS*	*ACSC*	*DenS*	*ACSC*	*DenS*
10	1	1	.79	.47	.56	.52
20	1	1	.75	.39	.51	.57
30	1	1	.70	.31	.44	.42
40	1	1	.67	.27	.37	.46
50	.98	1	.81	.70	.76	.88
60	.99	.99	.69	.68	.54	.64
70	1	1	.59	.78	.35	.55
80	.99	1	.71	.44	.42	.64
90	.98	1	.67	.35	.45	.54
100	1	1	.95	.82	.86	.85

Table 2: Comparitve performance of ACSC with DenStream over 100 windows of size 1,000

	DenStream			*CluStream*			*ACSC*		
	P	*F*	*R*	*P*	*F*	*R*	*P*	*F*	*R*
1*CDT*	.99	.82	.77	**1.0**	.88	.80	**1.0** (.01)	.**99**(.02)	.**98** (.02)
2*CHT*	.43	.27	.56	.24	.23	.55	.**79** (.03)	.**49** (.03)	.**57** (.01)
4*CR*	**1.0**	.67	.71	**1.0**	.89	.89	.94 (.04)	.**92** (.05) .	.**94** (.04)
4*CE*1*CF*	.**99**	.35	.56	.**99**	.**86**	.89	.87 (.04)	.85 (.04)	.**91** (.04)
Network Intrusion	**1.0**	.80	.81	.35	.13	.**86**	**1.0** (.00)	.**83** (.02)	.69 (.02)
Average	.88	.58	.68	.72	.60	.80	.**92**	.**82**	.**82**

Table 3: Average performance over the entire stream of each algorithm measured using purity (P), F-Measure (F) and Rand Index (R)

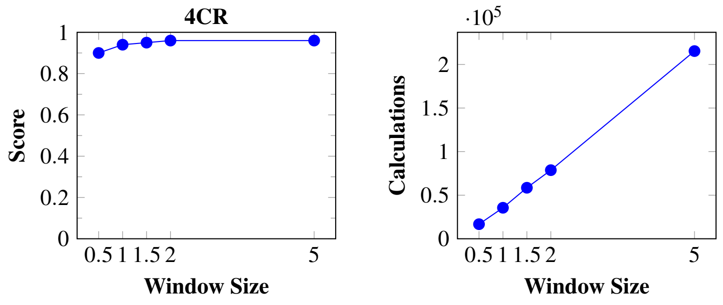

Fig. 4: Sensitivity of window size (left) on 4CR dataset and corresponding number of calculations

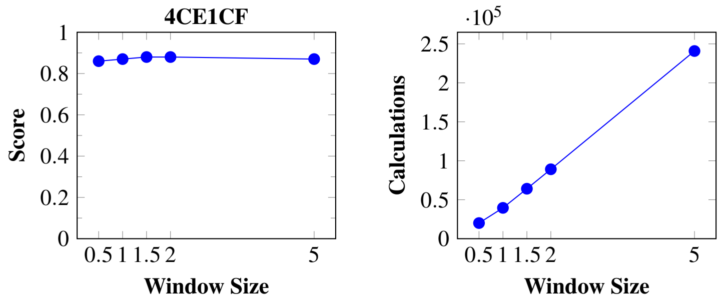

Fig. 5: Sensitivity of window size (left) on 4CE1CF and corresponding number of calculations

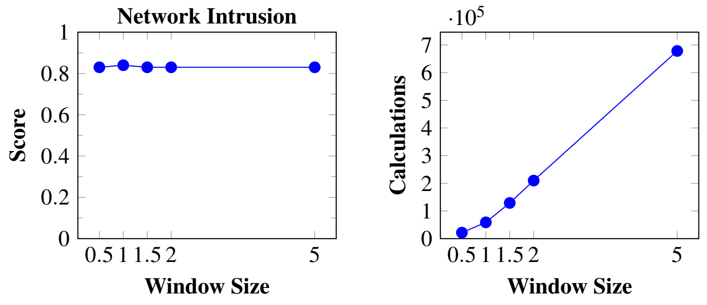

Fig. 6: Sensitivity of window size (left) on Network Intrusion Dataset and corresponding number of calculations

3.5 Discussion

Table 3 shows that ACSC , on average over all 5 datasets, outperforms both Den-Stream and ClusTream. The levels of cluster purity are comparable across each

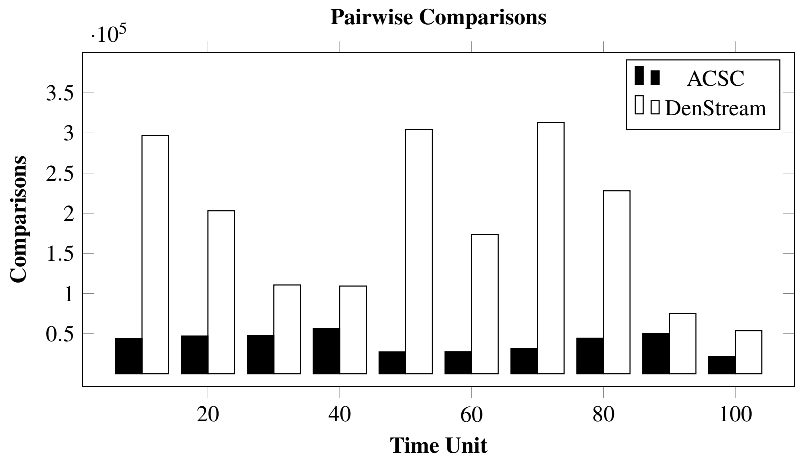

Fig. 7: Mean number of calculations performed on Network Intrusion dataset over 100 windows of size 1,000

dataset. Purity, in isolation, is not a very revealing evaluation metric as it does not consider the true topology of the data for example, assigning each data point to its own cluster would give 100% purity. It is, however, a useful metric when taken along side the F-Measure and Rand Index. ACSC achieves the best F-Measure and Rand Index scores on four out of five datasets and on average, is the best overall. The second phase of the algorithm, the inter-nest sorting phase, is the reason for this. The probabilistic functions for picking and dropping microclusters are biased towards the dissolution of smaller nests incorporating them into similar, larger nests. This improves the precision and recall scores (and hence the F-Measure) and creates clusters closer to the 'true' structure of the data. This is reflected in the Rand Index score.

Figures 1, 2 and 3 show the performance of the algorithm over time and table 2 compares the quality of ACSC and DenStream clusters over time on the real life Network intrusion dataset. It is interesting to note that ACSC performs comparatively but requires fewer parameters and considerably fewer calculations. Figure 7 shows that ACSC requires roughly 10 times fewer calculations. This is due to the fact that DenStream performs an exhaustive search for the nearest neighbour of each micro-cluster while ACSC uses a stochastic sampling method. If ACSC used a naive implementation whereby each ant (mico-cluster) compared itself with every other it would require O(N^2) time but each ant compares itself with a just a sample taken from each nest. The absolute worst case would require O(N^2) only if n data points in each window belonged to n different clusters. It is not possible to know the number of calculations a priori due to the stochastic nature of the algorithm, the size of ε and the number of clusters but experimental results show that the number of calculations is comparatively low and scales almost linearly with larger window sizes. Interestingly, the size of the window has little effect on the accuracy of the algorithm. This window size is just one of two parameters required (unlike Den-Stream, for example, which requires six). The second parameter is the sleep count for ants in the sorting phase. All experiments presented here use a value of 3. It is our hypothesis that a larger value will give improved results in data which contains classes that are very close in the feature space. This data would yield dense areas containing multiple classes and would require more sorting. Alternatively, well separated data would require a smaller value to avoid extra, unnecessary calculations. This hypothesis will be explorer in future research.

4 Conclusion

In this paper we proposed the Ant Colony Stream Clustering (ACSC) algorithm. It is based on the classic pick-and-drop Ant Colony algorithm to tackle the problem of density based clustering for streams. Clusters are identified as ‘nests’ of density reachable micro-clusters. Ants move along pheromone trails in dense areas of the data and probabilistically sort these nests based on local density and similarity with neighbouring micro-clusters.

ACSC was compared with two leading stream clustering algorithms across real and synthetic datasets and the results are encouraging. Experimental results show that ACSC performs competitively or favourably while requiring fewer parameters. Experiments also show that the algorithm requires comparatively fewer calculations. It uses the sliding window model and early results show that it scales almost linearly to larger window sizes. Further studies will investigate the scalability of the algorithm in terms of processing rates, data dimensionality and number of clusters. The ε parameter is adaptive and the sensitivity of this parameter has not been reported but will be expanded upon in future research. An aspect of the algorithm that can be improved is the off-line storage of the summary statistics. Ideally, these statistics could be stored in such a way as to give a ‘narrative’ to the stream so users can easily gauge where and how the stream is changing over time. This too, will be considered in future research.

Acknowledgments

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

References

1. Dorigo, M., Birattari, M., Sttzle, T. (2006). Ant colony optimization. Computational Intelligence Magazine, IEEE, 1(4), 28-39.
2. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29. pp. 81–92. VLDB '03, VLDB Endowment (2003), http://dl.acm.org/citation.cfm?id=1315451.1315460
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research 11, 1601–1604 (2010)
4. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SDM. vol. 6, pp. 328–339. SIAM (2006)
5. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chrétien, L.: The dynamics of collective sorting robot-like ants and ant-like robots. In: Proceedings of the 1st International Conference on Simulation of Adaptive Behavior From Animals to Animats. pp. 356–363 (1991)
6. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD. vol. 96, pp. 226–231 (1996)
7. Forestiero, A., Pizzuti, C., Spezzano, G.: A single pass algorithm for clustering evolving data streams based on swarm intelligence. Data Mining and Knowledge Discovery 26(1), 1–26 (2013)
8. Handl, J., Knowles, J., Dorigo, M.: Ant-based clustering and topographic mapping. Artificial life 12(1), 35–62 (2006)
9. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1(2), 95–113 (2007)
10. Hartmann, V.: Evolving agent swarms for clustering and sorting. In: Proceedings of the 7th Annual conference on Genetic and Evolutionary Computation. pp. 217–224. ACM (2005)
11. Labroche, N.: Fast ant-inspired clustering algorithm for web usage mining. In: Information Processing and Management of Uncertainty (2006)
12. Masmoudi, N., Azzag, H., Lebbah, M., Bertelle, C., Ben Jemaa, M.: How to use ants for data stream clustering. In: Proceedings of 2015 IEEE Congress on Evolutionary Computation, pp. 656–663 (2015)
13. Moise, G., Sander, J., Ester, M.: P3c: A robust projected clustering algorithm. In: Proceedings of the 6th International Conference on Data Mining (ICDM'06), pp. 414–425 (2006)
14. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: ACM SIGGRAPH computer graphics. vol. 21, pp. 25–34. ACM (1987)
15. Souza, V.M.A., Silva, D.F., Gama, J., Batista, G.E.A.P.A.: Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: Proceedings of SIAM International Conference on Data Mining, pp. 873–881 (2015)
16. Tu, L., Chen, Y.: Stream data clustering based on grid density and attraction. ACM Transactions on Knowledge Discovery from Data (TKDD) 3(3), 12 (2009)
17. Wan, L., Ng, W.K., Dang, X.H., Yu, P.S., Zhang, K.: Density-based clustering of data streams at multiple resolutions. ACM Transactions on Knowledge Discovery from Data (TKDD) 3(3), 14 (2009)
18. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: A new data clustering algorithm and its applications. Data Mining and Knowledge Discovery 1(2), 141–182 (1997)
19. Runkler, T. A. (2005). Ant colony optimization of clustering models. International Journal of Intelligent Systems, 20(12), 1233-1251.
20. Jardine, Nick, and Cornelis Joost van Rijsbergen. "The use of hierarchic clustering in information retrieval." Information storage and retrieval 7.5 (1971): 217-240.
21. Rand, William M. "Objective criteria for the evaluation of clustering methods." Journal of the American Statistical association 66.336 (1971): 846-850.

22. Bifet, Albert, et al. ”Moa: Massive online analysis.” Journal of Machine Learning Research 11.May (2010): 1601-1604.