Efficient Online Stream Clustering Based on Fast Peeling of Boundary Micro-Cluster: Supplementary Material

Jiarui Sun, Mingjing Du, *Member, IEEE,* Chen Sun, and Yongquan Dong

1. The Micro-cluster Structure

Fig. S1 illustrates a three-dimensional micro-cluster structure. s denotes the seed point (small blue ball) of micro-cluster mc. r is the predefined micro-cluster radius. Some data points (small white balls) in the data stream are absorbed by the micro-cluster mc.

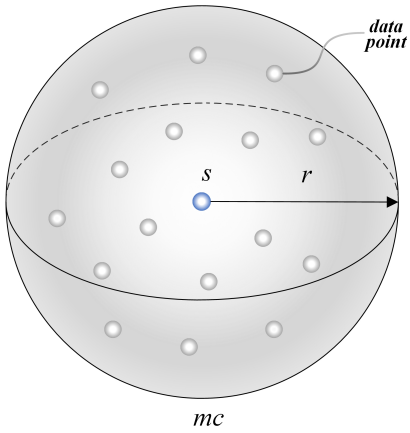


Fig. S1. The micro-cluster structure.

2. A Proof of Acyclicity

Proof. For a chain, we assume that it forms a loop. It means that at a given time t, there is a chain consisting of different micro-clusters $mc_1, mc_2, \ldots, mc_n$. Among these micro-clusters, there are the following relationships: $Rep(mc_1, t) = mc_2$, $Rep(mc_2, t) = mc_3$, $\ldots$, $Rep(mc_n, t) = mc_1$.

According to Definition 3, if $Rep(mc_1, t) = mc_2$, then $DKD(mc_1, t) < DKD(mc_2, t)$. Similarly, we have $DKD(mc_2, t) < DKD(mc_3, t)$, $\ldots$, $DKD(mc_{n-1}, t) < DKD(mc_n, t)$, $DKD(mc_n, t) < DKD(mc_1, t)$.

Then, we can derive that $DKD(mc_1, t) < DKD(mc_n, t)$ and $DKD(mc_n, t) < DKD(mc_1, t)$, which is obviously self-contradictory.

Therefore, the assumption does not hold, and there must be no loop in the chain. □

3. Details of The Complexity Analysis

To aid the reader in understanding the source of each cost, a detailed description of each step of the complexity analysis is provided.

Assuming that the number of micro-clusters in memory at time t is M_t ($M_t \ll n$, $M_t \in \mathbb{R}^d$), where n is the number of data points in the data stream $\boldsymbol{X}$, d is the dimensionality of the data stream $\boldsymbol{X}$, k is the number of nearest neighbors, and αk is the number of augmented nearest neighbors.

As a new data point arrive, FBPStream attempts to summarize it into active or inactive clusters. Finding the nearest micro-cluster to a data point requires traversing all micro-clusters and calculating their distances. During this process, it takes $O(M_t)$ to traverse the micro-clusters and $O(d)$ to calculate the distance from a point to a micro-cluster. Thus, for all points in the stream,

This work was supported in part by the National Natural Science Foundation of China under Grant 62006104, and Grant 61872168, and in part by the Natural Science Foundation of the Jiangsu Higher Education Institutions under Grant 20KJB520012. *(Corresponding author: Mingjing Du.)*

J. Sun, M. Du, C. Sun, and Y. Dong are with the School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China. (e-mail: sunjr@jsnu.edu.cn; dumj@jsnu.edu.cn; 3020200309@jsnu.edu.cn; tomdyq@163.com).

a total of $O(n) \times O(M_t) \times O(d) = O(dnM_t)$ is required. Decaying each micro-cluster in the *RP-Tree* in turn and detecting the status of the micro-cluster takes $O(M_t)$. For micro-clusters whose statuses have changed, it takes $O(M_t)$ to remove them from the *RP-Tree* and insert them into the *Outlier Pool*. Each of them only requires linear time. Meanwhile, removing a micro-cluster from the *Augmented k-NN Graph* involves traversing the graph to update the nearest-neighbor information of the other vertices, which requires $O(\alpha k M_t)$. Thus, for all points in the stream, a total of $O(n) \times (O(M_t) + O(M_t) + O(\alpha k M_t)) \approx O(\alpha k n M_t)$ is required. If an inactive micro-cluster absorbs a new data point and grows into an active micro-cluster, it is removed from the *Outlier Pool* and inserted into the *RP-Tree*, which requires an $O(1)$ operation. Meanwhile, inserting micro-clusters into the *Augmented k-NN Graph* involves traversing the graph to update the nearest-neighbor information using the binary insertion method, which requires $O(M_t log(\alpha k))$. Updating the decay-based kernel density of active micro-clusters requires at most $O(kM_t)$. Thus, for all points in the stream, a total of $O(n) \times (O(1) + O(M_t log(\alpha k)) + O(kM_t)) \approx O(knM_t)$ is required. If the new data point is not absorbed by any existing micro-cluster, a new micro-cluster need to be created, which requires $O(1)$. For the entire stream, at most $O(n)$ is required.

In summary, the time complexity required to absorb each data point in the data stream $\boldsymbol{X}$ is $O(dnM_t) + O(\alpha knM_t) + O(knM_t) + O(n) \approx O((d + \alpha k)nM_t)$.

The time complexity analysis for clustering the active micro-clusters is as follows. The watermark-based fast peeling process loops ω times and peels off the boundary micro-clusters by traversing the remaining active micro-clusters, which requires at most $O(\omega M_t)$. The process of clustering core micro-clusters traverses core micro-clusters using a hierarchical traversal approach, which requires at most $O(M_t + kM_t)$. The linkage of boundary micro-clusters relies on traversing the k-nearest neighbors of each boundary micro-cluster to find representatives that satisfy the conditions. It takes at most $O(kM_t)$. Due to the parallel clustering setup, $\max(O(M_t + kM_t), O(kM_t))$ is required for the clustering of core micro-clusters and the linkage of boundary micro-clusters. Assigning boundary micro-clusters uses a recursive implementation, which takes at most $O(M_t)$. Thus, for all points in the stream, a total of $O(n) \times (O(\omega M_t) + \max(O(M_t + kM_t), O(kM_t)) + O(M_t)) \approx O((\omega + k)nM_t)$ is required.

In conclusion, the time complexity of FBPStream is $O((d + \alpha k)nM_t) + O((\omega + k)nM_t) \approx O((d + \omega + \alpha k)nM_t)$.

The space complexity of FBPStream mainly comes from the micro-clusters and the *Augmented k-NN Graph* in memory. The storage cost of all micro-clusters in memory is $O(dM_t)$. The *Augmented k-NN Graph* is stored using an adjacency list, so the storage cost of all edge lists is at most $O(\alpha k d M_t)$. Therefore, the total space complexity is $O(dM_t) + O(\alpha k d M_t) \approx O(\alpha k d M_t)$.

4. Descriptions of Synthetic Datasets and Real-world Datasets

We generate the Stream_1 dataset to verify the ability of filtering outliers, coping with concept drifts, and recognizing arbitrarily shaped clusters. As shown in Fig. 5(a), it contains 8 clusters (each with 25,000 data points) and 20,000 noise points. The Stream_1 dataset consists of 8 clusters arranged in four groups and produced sequentially. In each group, data points appear at random (jumping from cluster to cluster). In order to test the performance of FBPStream in clustering data streams with varying densities and ambiguous boundaries, we also generate 3 synthetic datasets $\text{Stream}_2 \sim \text{Stream}_4$ (each containing 10,000 noise points). More specifically, Fig. 5(b) shows two clusters with varying densities. Fig. 5(c) illustrates four clusters with ambiguous boundaries, and four clusters with varying densities and ambiguous boundaries are shown in Fig. 5(d). The clusters in Stream_2, Stream_3, and Stream_4 are simultaneous, i.e., the data points appear randomly in any of the clusters. The Hyperplane[1] dataset is created with a random hyperplane generator. It is a data stream with fast incremental drift, where a 10-dimensional hyperplane changes position and orientation. The data points in the Hyperplane dataset are generated randomly between multiple hyperplanes.

As part of our experiments, we use several real-world datasets from the USP Data Stream Repository[2]. These datasets include NOAAweather, Powersupply, Electricity, and Insects. The NOAAweather dataset contains weather information collected from 1949 to 1999 in Bellevue, Nebraska. Its purpose is to predict whether it will rain on a given date. The Powersupply dataset contains the hourly power supply of the Italian electricity company. It is usually used to predict which hour the current supply belongs to (1 hour out of 24). Electricity is a dataset for the New South Wales electricity market in Australia, where prices are affected by supply and demand. The two label values represent changes in price (1 = up or 0 = down). The Insects dataset contains data collected from sensors in mosquito traps that measure the flight characteristics of captured insects. For each sample, there are 33 features from the obtained signal spectrum. An incrementally drifted and balanced version of this dataset is used in this paper. Rialto and Poker are two real-world datasets available from the driftDatasets repository[3]. The Rialto dataset consists of 27-dimensional RGB histograms encoded from the colorful facades of ten buildings adjacent to Venice's famous Rialto Bridge. The images are taken from a fixed webcam recording time-lapse video. The records cover 20 consecutive days in May-June 2016. In Poker, each record is a five-card draw from a standard 52-card deck. Each card contains ten attributes since each card is described by two attributes (suit and rank). The Airlines[4] dataset, from Ikonomovska, predicts whether a flight will be delayed based on the scheduled departures of commercial flights within the US. The Sensor[1] dataset contains

[1] https://www.cse.fau.edu/~xqzhu/stream.html
[2] https://sites.google.com/view/uspdsrepository
[3] https://github.com/vlosing/driftDatasets
[4] http://kt.ijs.si/elena_ikonomovska/data.html

information collected continuously over two months from 54 sensors deployed in Intel Berkeley Research Lab. Each record consists of temperature, humidity, light, sensor voltage data, and sensor ID. The remaining 2 real-world datasets are from the UCI Machine Learning Repository[5]. The Adult dataset is extracted from the 1994 Census database and can predict whether a person makes more than $50000 per year. The Covertype dataset is used to predict forest cover types for four wilderness areas located in the Roosevelt National Forest of northern Colorado based on cartographic variables. The 10 real-world datasets, collected in complex environments, resulted in data points appearing in different clusters according to specific patterns.

5. Evaluation Metrics

For all algorithms, we use two metrics to measure their performance on each dataset. These two metrics are Purity [1] and *Cluster Mapping Measure* (CMM) [2].

Purity is defined as:

$$Purity = \frac{1}{N}\sum_{k}\max_{j}|\boldsymbol{C}_k \cap \boldsymbol{T}_j| \tag{1}$$

where N is the total number of samples. $\boldsymbol{C}_k$ is all samples in the kth cluster obtained by clustering. $\boldsymbol{T}_j$ is all samples in the jth ground-truth class. $Purity \in [0, 1]$, and a larger value indicates better clustering.

Additionally, we use *Cluster Mapping Measure*, an external criterion designed specifically for data stream clustering. The CMM considers the freshness of the data points. Due to the evolution of the data stream, the underlying data distribution gradually changes. The CMM metric correctly reflects errors associated with emerging, splitting, or moving clusters. These situations are inherent to the streaming context.

CMM combines penalties imposed by each fault:

1) **Missed points**: As the stream evolves, constantly moving clusters may eventually miss points, so the CMM penalizes these missed points.
2) **Misplaced points**: Clusters may eventually overlap as they flow, so the CMM penalizes misplaced points.
3) **Noisy points**: CMM penalizes noisy points inserted into existing clusters.

In general, the CMM value ranges from 0 to 1. A larger CMM value indicates a higher clustering quality and a smaller CMM value indicates a lower clustering quality.

6. Parameters of All Algorithms

TABLE S1
PARAMETERS OF ALL ALGORITHMS.

Algorithm	#Num	#Params
FBPStream	5	$r \in [0.001, 1)$, λ=0.998, β=0.0021, $k \in [4, 14]$, $\omega \in \{0, 1, 2, 3\}$
EDMStream	5	$r \in [0.001, 1)$, a=0.998, λ=1, β=0.0021, $\delta \in [0.001, 1]$
CEDAS	3	$r \in [0.001, 1)$, $decay$=0.001, $minPts \in \{5, 10\}$
DBSTREAM	5	$r \in [0.001, 1)$, λ=0.0028, t_{gap}=100, $\alpha \in [0.001, 0.3]$, w_{min}=0.01
D-Stream	5	$len \in [0.01, 1)$, λ=0.998, C_m=3, C_l=0.8, β=0.3
DenStream	4	$\epsilon \in [0.001, 1)$, λ=0.0028, $\beta \in [0.5, 0.9]$, $\mu \in [2, 10]$
HDDStream	6	$\epsilon \in [0.001, 1)$, λ=0.0028, $\beta \in [0.5, 0.9]$, $\mu \in [2, 10]$, $\pi \in [2, 20]$, $\delta \in [2, 10]$
DWDP-Stream	6	$\epsilon \in [0.001, 1)$, λ=0.0028, $\beta \in [0.5, 0.9]$, $\mu \in [2, 10]$, $K \in [2, 54]$, $\alpha \in [0.1, 0.9]$
MR-Stream	9	$\epsilon \in [0.001, 1)$, a=1, λ=1.002, C_H=3, C_L=0.8, $H \in [2, 10]$, $h \in [2, 10]$, $\beta \in [1, 10]$, $\mu \in [1, 10]$
MuDi-Stream	4	$len \in [0.01, 1)$, λ=0.0028, $\alpha \in [0.5, 0.9]$, $minPts \in [3, 20]$
ESA-Stream	3	$len \in [0.01, 1)$, λ=0.998, $upbound$=1

7. Clustering Results of Algorithms on 2-dimensional Synthetic Datasets

As shown in Figs. S2-S3, the clustering results of all algorithms on Stream_1, Stream_2, Stream_3, and Stream_4 are presented in the form of micro-clusters or grids.

8. Statistical Tests

To compare the performance of all algorithms more visually and scientifically, we apply the Friedman test and the Nemenyi post-hoc test to further analyze the clustering results. A null hypothesis (H_0) suggests that the algorithms perform consistently, while an alternative hypothesis (H_1) suggests that the algorithms perform differently. As a result of counting the 11 algorithms on 15 datasets, the average ranks are presented in Table S2. Accordingly, we obtain a list of the comprehensive performances of all algorithms in descending order: FBPStream, ESA-Stream, MR-Stream, DBSTREAM, EDMStream, CEDAS, DenStream, HDDStream, DWDP-Stream, MuDi-Stream, and D-Stream.

[5]http://archive.ics.uci.edu/ml/datasets.php

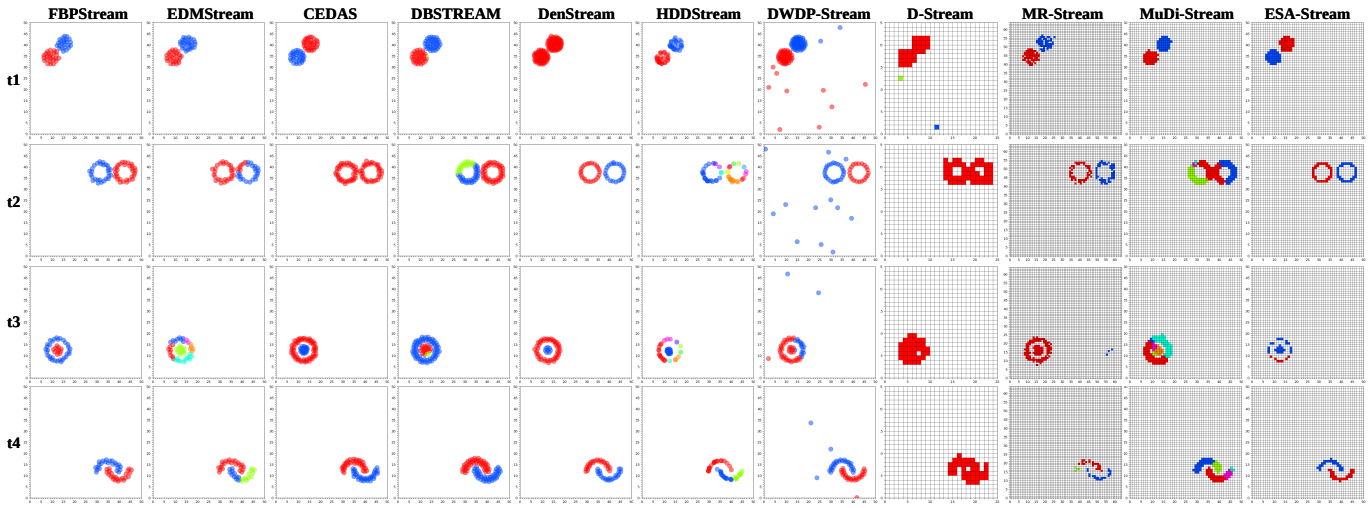


Fig. S2. Clustering results of algorithms on Stream_1. Each row shows the clustering results produced by all the algorithms at the same gap time, where the 4 gap times satisfy t1<t2<t3<t4.

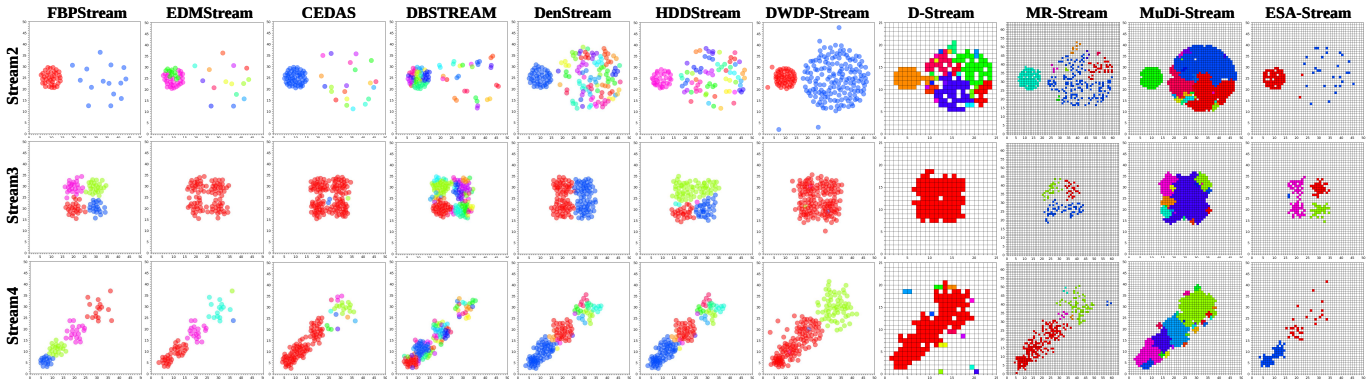


Fig. S3. Clustering results of algorithms on Stream_2, Stream_3, and Stream_4.

TABLE S2
AVERAGE RANKS OF ALGORITHMS.

Algorithm	FBPStream	EDMStream	CEDAS	DBSTREAM	DenStream	HDDStream	DWDP-Stream	D-Stream	MR-Stream	MuDi-Stream	ESA-Stream
AvgRank	1.17	5.23	5.30	5.13	6.60	7.33	8.13	9.77	4.60	8.40	4.20

Let the number of algorithms be M and the number of datasets multiplied by the number of metrics equals N. The statistical variable τ_F obeys an F distribution with degrees of freedom $M-1$ and $(N-1)\times(M-1)$. The formula is as follows:

$$\begin{cases} \tau_F = \frac{(N-1)\tau_{\chi^2}}{N(M-1)-\tau_{\chi^2}} \\ \tau_{\chi^2} = \frac{12N}{M(M+1)}\left(\sum_{i=1}^{M} r_i^2 - \frac{M(M+1)^2}{4}\right) \end{cases} \tag{2}$$

From Eq. (2), it can be concluded that $\tau_F = 29.347$. In particular, when the degrees of freedom for τ_F are $11-1=10$ and $(11-1)\times(30-1)=290$, the critical value at a 0.05 significance level is $F(10,290)=2.177$. Notably, it is evident that τ_F significantly exceeds the critical value $F(10,290)$. Therefore, H_0 is rejected and we need to make the next test (Nemenyi).

In the Nemenyi post-hoc test, the critical difference (CD) is calculated as $q_\alpha\sqrt{M(M+1)/6N} = 2.709$, where q_α represents the test threshold. With a significance level of 0.05, q_α is determined as 3.164 through the specified query. Nemenyi post-hoc test suggests that if the difference in average ranks is greater than CD, the algorithms perform significantly differently in terms of clustering.

Fig. S4 illustrates the results of the Nemenyi post-hoc test, where the green dots indicate the average rank of each algorithm, the red line represents the confidence interval of length 2.709, and the blue vertical line represents the upper confidence limit

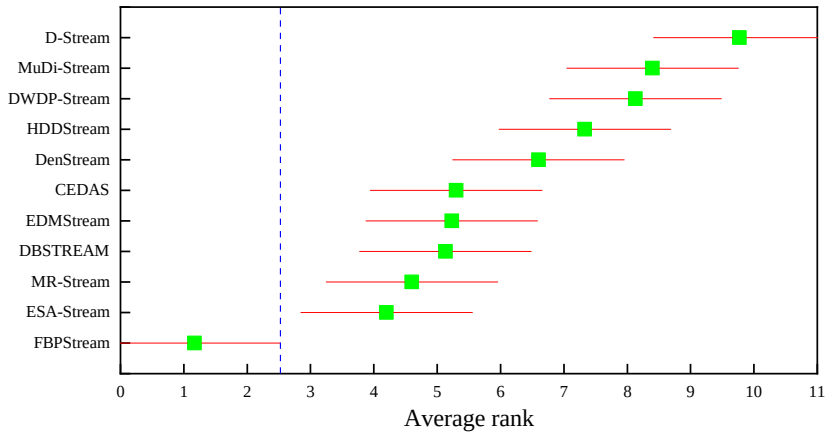


Fig. S4. Results of the Nemenyi post-hoc test.

of FBPStream. Observe that the blue vertical line of FBPStream does not overlap with the red lines of the other algorithms, which means the average rank difference between FBPStream and its competitors is higher than the critical difference. Thus, the conclusion that the proposed algorithm outperforms its competitors is statistically significant.

9. Ablation Experiments

In this section, we perform a series of ablation experiments to validate the effectiveness of two key contributions in our algorithm: the decay-based kernel density estimation method and the boundary micro-cluster peeling clustering strategy. Initially, the decay-based kernel density estimation (*DKD*) is omitted from the proposed algorithm, and micro-cluster weights are employed in its place. The modified algorithm is referred to as FBPStream-DKD. Then, the boundary micro-cluster peeling clustering strategy is removed from the proposed algorithm, and a straightforward merging technique is adopted that combines adjacent micro-clusters based on a predetermined distance threshold (i.e., $2r$). The modified algorithm is referred to as FBPStream-BP. FBPStream-DKD and FBPStream-BP are evaluated on the 15 datasets listed in Table II. Their parameter settings align with those specified in FBPStream for each dataset.

Table S3 provides detailed results of the ablation experiments. For each dataset, the best metric values are bolded. Moreover, Fig. S5 provides a visual comparison of clustering performance. As is evident, both modified algorithms are less effective than FBPStream (i.e., the proposed algorithm). Furthermore, FBPStream-BP exhibits lower performance than FBPStream-DKD. Specifically, compared to FBPStream, FBPStream-DKD reduces the purity by an average of 6% and CMM by an average of 6.1% on all datasets. The degradation in the performance of FBPStream-DKD stems from the fact that the modified algorithm ignores micro-cluster distribution information. Consequently, it encounters difficulties in capturing sufficient information for the accurate detection of multi-density clusters in a stream. FBPStream-BP's purity reduces by an average of 12.4% compared to FBPStream, while its CMM reduces by an average of 11.1%. Compared with FBPStream-DKD, the overall performance of FBPStream-BP is even lower. The results indicate that the proposed boundary micro-cluster peeling clustering strategy improves FBPStream's clustering performance significantly, particularly when confronted with the challenge of identifying clusters with ambiguous boundaries in a stream. In summary, the ablation experiments validate that the decay-based kernel density estimation method and the boundary micro-cluster peeling clustering strategy can enhance the overall clustering performance significantly.

10. Running Time of FBPStream Using Single-threaded and Multi-threaded

Finally, we test the effect of using different strategies (single/multi-threaded) in FBPStream on the efficiency of the algorithm on real-world datasets. For fairness of the comparison, both strategies use the same data size and gap values on the same dataset. Fig. S6 shows the results.

Obviously, FBPStream, which uses the multi-threaded techniques, runs faster. Specifically, the algorithm efficiency can be improved by clustering core micro-clusters and linking boundary micro-clusters by two threads, respectively. Compared to the single-threaded strategy, the multi-threaded strategy reduces running time by 1.933 seconds on average. Furthermore, the multi-threaded strategy performs 15.9% better than the single-threaded strategy in efficiency. We also compare the running time of the single-threaded strategy with CEDAS, DenStream, DWDP-Stream, MR-Stream, and ESA-Stream in Fig. S6. Although the single-threaded strategy makes the algorithm less efficient, it is still faster than most comparison algorithms. It means that FBPStream can still perform clustering efficiently in some stream scenarios requiring a single-threaded strategy.

TABLE S3
RESULTS OF ABLATION EXPERIMENTS.

Algorithm	$Stream_1$		$Stream_2$		$Stream_3$		$Stream_4$		Hyperplane	
	Purity	CMM	Purity	CMM	Purity	CMM	Purity	CMM	Purity	CMM
FBPStream	**0.985**	**0.843**	**0.993**	**0.847**	**0.983**	**0.862**	**0.977**	**0.854**	**0.671**	**0.726**
FBPStream-DKD	0.981	0.815	0.965	0.809	0.979	0.755	0.852	0.728	0.605	0.692
FBPStream-BP	0.804	0.631	**0.993**	**0.847**	0.822	0.675	0.731	0.730	0.598	0.657
Algorithm	NOAAweather		Powersupply		Adult		Electricity		Insects	
	Purity	CMM	Purity	CMM	Purity	CMM	Purity	CMM	Purity	CMM
FBPStream	**0.790**	**0.921**	**0.438**	**0.854**	**0.888**	**0.737**	**0.812**	**0.895**	**0.674**	**0.848**
FBPStream-DKD	0.720	0.893	0.306	0.796	0.871	0.715	0.801	0.852	0.534	0.756
FBPStream-BP	0.674	0.849	0.376	0.754	0.822	0.693	0.783	0.811	0.384	0.658
Algorithm	Rialto		Airlines		Poker		Covertype		Sensor	
	Purity	CMM	Purity	CMM	Purity	CMM	Purity	CMM	Purity	CMM
FBPStream	**0.428**	**0.850**	**0.661**	**0.905**	**0.704**	**0.933**	**0.935**	**0.893**	**0.670**	**0.832**
FBPStream-DKD	0.416	0.775	0.642	0.863	0.686	0.912	0.912	0.868	0.641	0.785
FBPStream-BP	0.404	0.729	0.647	0.899	0.651	0.835	0.866	0.834	0.616	0.777

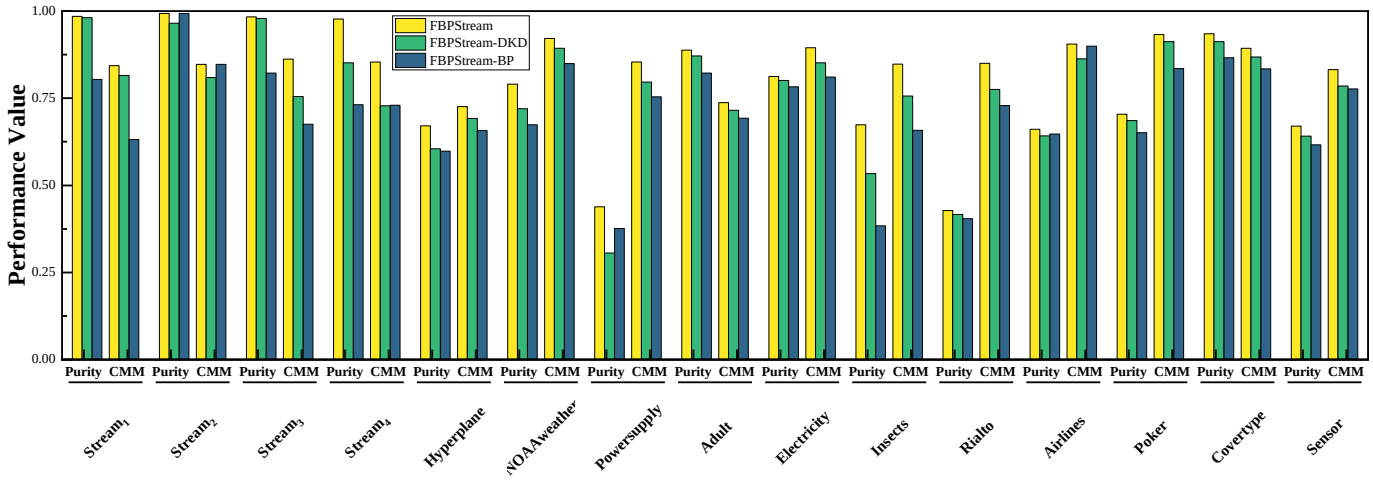


Fig. S5. A histogram of ablation experiment results.

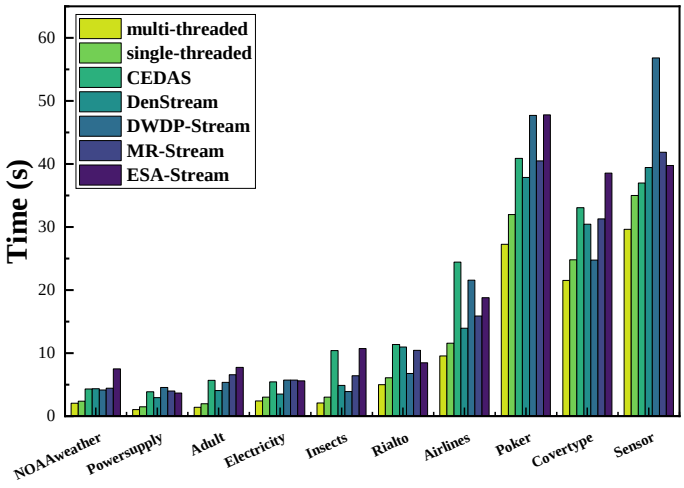


Fig. S6. Running time comparison of FBPStream using single-threaded and multi-threaded on real-world datasets.

11. FURTHER PARAMETER ANALYSIS

FBPStream has five parameters, namely the micro-cluster radius r, the decay factor λ, the weight threshold factor β, the number of nearest neighbors k, and the peeling level ω. In the main text, we have analyzed two of the most critical parameters (i.e., k and ω). Further analysis will be conducted to examine the effect of the remaining three parameters on the final clustering

results. When analyzing one parameter, the other parameters are fixed.

Firstly, we discuss how the micro-cluster radius r affects the clustering results of FBPStream. In order to facilitate visualization, we conduct experiments on the Stream$_1$ dataset. On the Stream$_1$ dataset, we set r to increment gradually from 0.01 to 0.1. Fig. S7 illustrates the clustering results of FBPStream at four different gap times (t1<t2<t3<t4) with varying r values. Fig. S8 provides monitoring of the variation of each metric (Purity, Number of micro-clusters, Memory usage, Running time). Upon observation, it is found that as r increases, the number of micro-clusters decreases. As additional data points are absorbed into a micro-cluster, the likelihood of a decline in clustering quality increases. As a result, purity also shows a decreasing trend as shown in Fig. S8(a). Moreover, as r increases, the number of micro-clusters decreases, memory usage decreases, and the proposed algorithm runs faster. Specifically, as r increases from 0.01 to 0.03, the number of micro-clusters reduces pronouncedly. Note that memory usage and running time show similar decreasing trends. As r increases from 0.03 to 0.1, all monitored metrics become stable. Therefore, stream clustering algorithms must balance clustering quality against clustering cost (e.g., memory and time), with the goal of obtaining clustering results of the highest possible quality within acceptable memory consumption and waiting time. To select an appropriate micro-cluster radius r, we should consider the complexity of the data stream. In the context of high-dimensional data streams, it is common practice to increase the radius of micro-clusters to cope with the sparsity of samples in high-dimensional spaces.

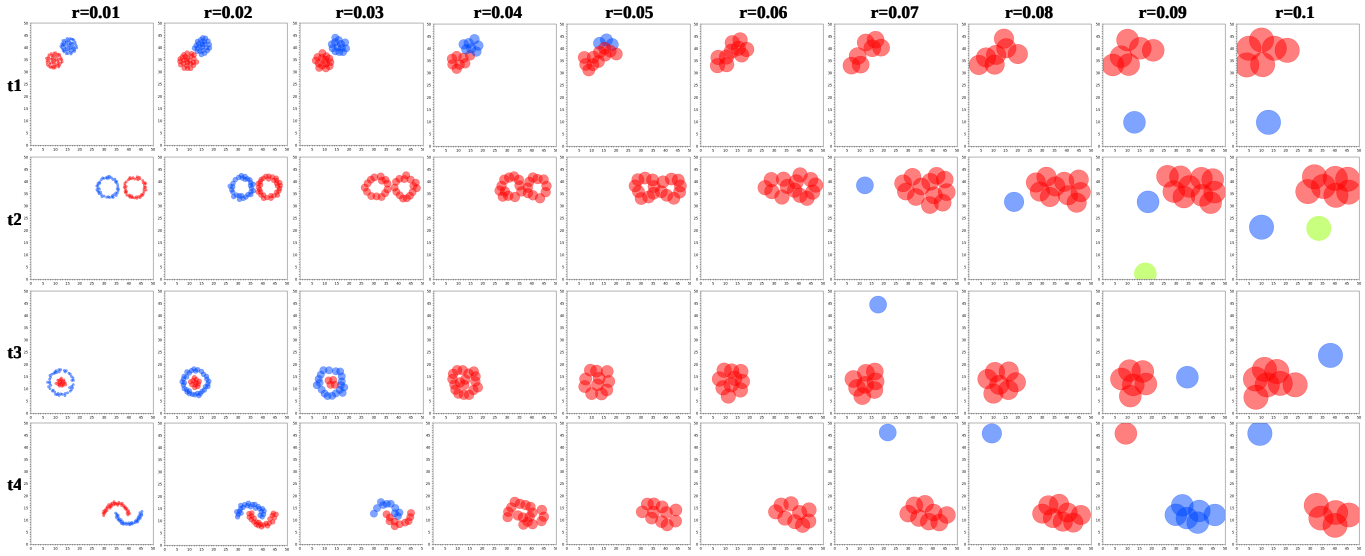
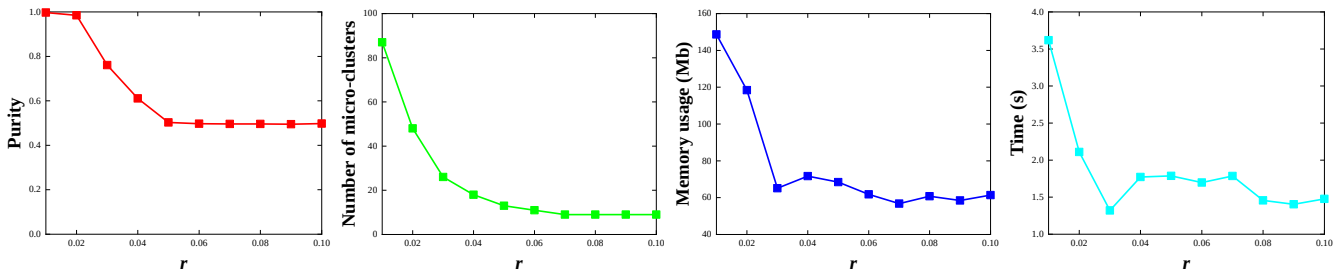


Fig. S7. Clustering results of FBPStream with varying r on Stream$_1$. Each row shows the clustering results with varying r at the same gap time, where the 4 gap times satisfy t1<t2<t3<t4.



(a) Purity (b) Number of micro-clusters (c) Memory usage (d) Running time

Fig. S8. Performance changes of FBPStream with varying r on Stream$_1$.

Secondly, we analyze how the decay factor λ affects clustering results. we conduct experiments on a synthetic dataset called Bench[6]. The data points of Bench are divided into two classes, which move diagonally in 2-dimensional space, overlapping in the center. On the Bench dataset, we set λ to decrease from 0.999 to 0.99. Fig. S9 illustrates the clustering results of FBPStream at two different gap times (t1<t2) while considering varying values of λ. We monitor each metric in Fig. S10. As shown in Fig. S9, when $\lambda = 0.999$, two clusters intersect at both t1 and t2 gap times. These clusters are recognized as the

[6]https://github.com/Xicks/DataStream_DataSets

same clusters by the algorithm. It is observed that as λ decreases, the length of clusters also decreases. Eventually, the two clusters become separated. As a result, the purity increases (shown in Fig. S10(a)). Due to the fact that λ decreases, the data weights decays faster, thus the older data has a reduced effect on the clustering results. Moreover, as λ decreases, the number of micro-clusters decreases, memory usage decreases, and the proposed algorithm runs faster. Following the settings of most algorithms (e.g., D-Stream [3], MR-Stream [4], EDMStream [5], and ESA-Stream [6]), this paper sets $\lambda = 0.998$.

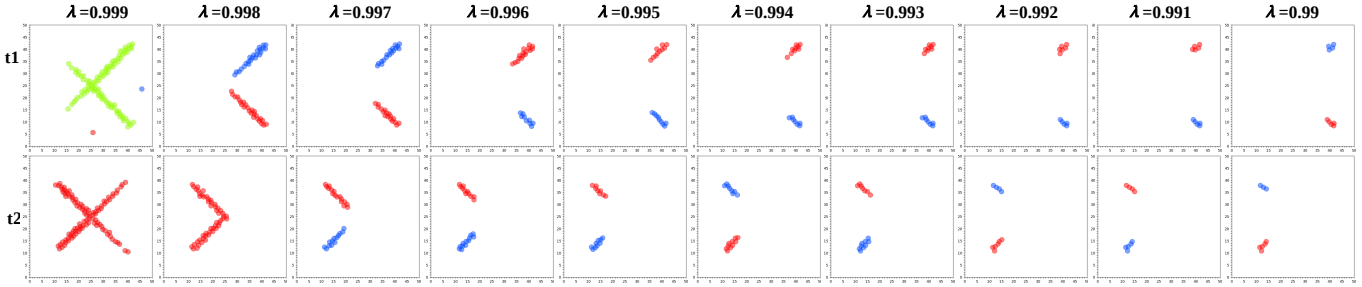
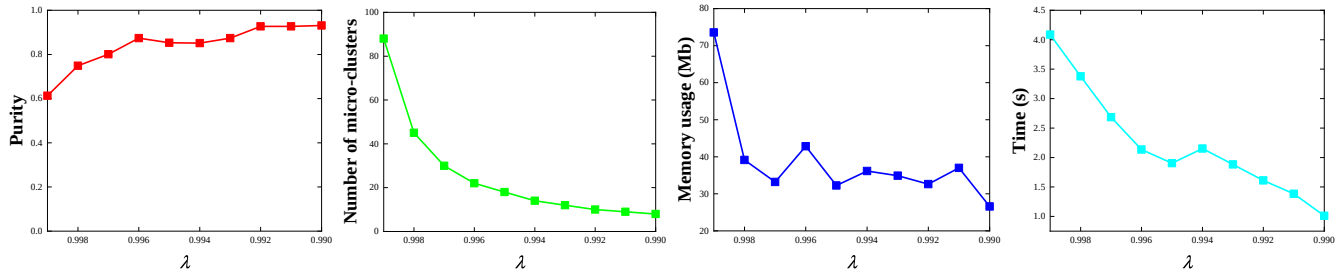


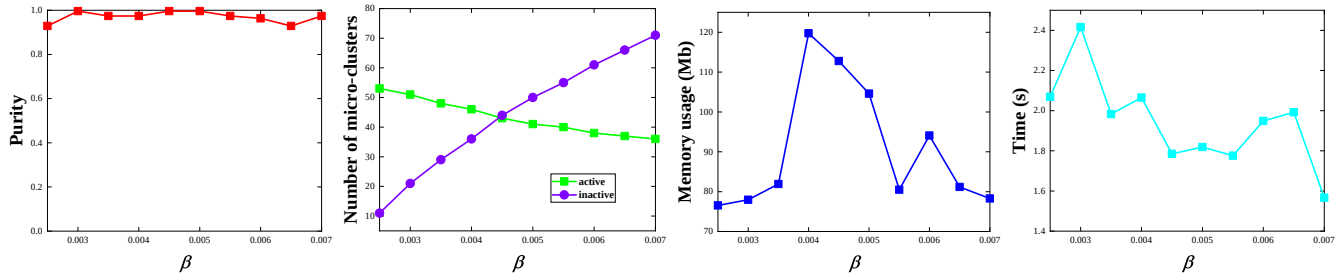
Fig. S9. Clustering results of FBPStream with varying λ on Bench. Each row shows the clustering results with varying λ at the same gap time, where the 2 gap times satisfy t1<t2.



(a) Purity (b) Number of micro-clusters (c) Memory usage (d) Running time

Fig. S10. Performance changes of FBPStream with varying λ on Bench.

Finally, we discuss how the weight threshold factor β affects the clustering results of FBPStream. We conduct experiments on the Stream$_1$ dataset. Since β belongs to the range $(1-\lambda, 1)$, when $\lambda = 0.998$, β belongs to the range $(0.002, 1)$. We set the β to increment gradually from 0.0025 to 0.007. Fig. S11 provides monitoring of the variation of each metric (Purity, Number of micro-clusters, Memory usage, Running time). From Fig. S11(b), it can be observed that as the β value increases, the number of active micro-clusters decreases and the number of inactive micro-clusters increases. Overall, the number of micro-clusters is approximately constant. The memory cost first increases and then decreases, and the execution speed of the algorithm shows an increasing trend. Fig. S11(a) illustrates that clustering purity is always stable. Clearly, β affects primarily the number of active and inactive micro-clusters. That is the larger the β value, the smaller the number of active micro-clusters and the shorter the algorithm running time. To maximize the capture of the original cluster structure, FBPStream requires obtaining as many active micro-clusters as possible. Hence, the β value is set as small as possible in this paper, i.e. $\beta = 0.0021$.



(a) Purity (b) Number of micro-clusters (c) Memory usage (d) Running time

Fig. S11. Performance changes of FBPStream with varying β on Stream$_1$.

In conclusion, we conduct meticulous experiments and analyze three parameters, namely the micro-cluster radius r, the decay factor λ, and the weight threshold factor β on the Stream$_1$ and Bench datasets. For each parameter, we provide theoretical explanations and suggestions for value selection.

12. The flowchart of FBPStream

Fig. S12 presents a comprehensive workflow of the proposed algorithm.

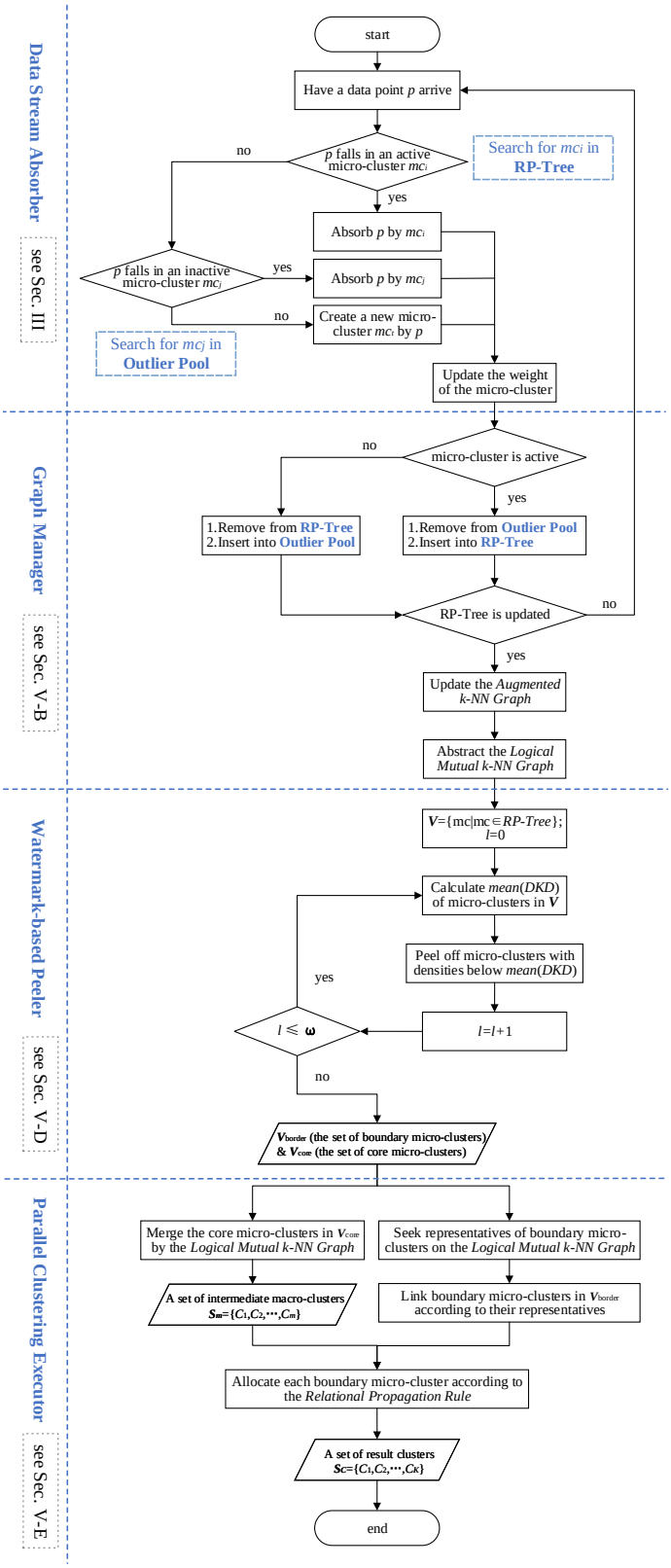


Fig. S12. The flowchart of FBPStream.

REFERENCES

[1] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Machine Learning*, vol. 55, pp. 311–331, 2004.

[2] H. Kremer, P. Kranen, T. Jansen, T. Seidl, A. Bifet, G. Holmes, and B. Pfahringer, "An effective evaluation measure for clustering on evolving data streams," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 868–876.

[3] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, California, USA, 2007, pp. 133–142.

[4] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Density-based clustering of data streams at multiple resolutions," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 3, pp. 1–28, 2009.

[5] S. Gong, Y. Zhang, and G. Yu, "Clustering stream data by exploring the evolution of density mountain," *Proceedings of the VLDB Endowment*, vol. 11, no. 4, pp. 393–405, 2017.

[6] Y. Li, H. Li, Z. Wang, B. Liu, J. Cui, and H. Fei, "ESA-Stream: Efficient self-adaptive online data stream clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 2, pp. 617–630, 2022.