TECHNICAL REPORT
DATA STREAM CLUSTERING
Affiliation: Deakin University
Date: 21-06-2024

Rohit Bohra, 22EE30039, rohitbohra03@kgpian.iitkgp.ac.in, Research Intern
Meet Modi, 22EE10092,mhm20@kgpian.iitkgp.ac.in , Research Intern
Priyavrat Jay Mishra, 22EE10093, priyavrat.misshra001@kgpian.iitkgp.ac.in, Research Intern
Md Mutahhir Azam, 22IE10029, mutahhir_azam@kgpian.iitkgp.ac.in, Research Intern

ACM Reference Format:
Xin Wang, Zhengru Wang, Zhenyu Wu, Shuhao Zhang, Xuanhua Shi, and Li Lu. 2023. Data Stream Clustering: An In-depth Empirical Study. Proc. ACM Manag. Data. 1, 2, Article 162 (June 2023), 26 pages. https://doi.org/ 10.1145/3589307

ABSTRACT

This research delves into the source code of recent research papers on data stream clustering, aiming to reproduce the results presented in the original works. The research paper we validated can be found here. https://doi.org/ 10.1145/3589307. We conducted an in-depth study of key design aspects of Dynamic Stream Clustering (DSC) algorithms, including (1) data summarization structures, (2) window models, (3) outlier detection mechanisms, and (4) offline refinement strategies. Each aspect includes multiple design choices, leading to different DSC algorithms with varying performance under diverse workloads. As part of this investigation, we implemented all these approaches within a unified framework, Sesame, https://github.com/intellistream/Sesame, using C++ to eliminate discrepancies caused by different programming languages and compilers. The **research paper also develops a novel DSC algorithm named Benne**, which integrates flexible design choices from the four key aspects. Benne demonstrates superior accuracy or efficiency compared to state-of-the-art algorithms across all tested workloads. The outcome of this research reaffirms the analysis that existing design options inherently trade-offs between accuracy and efficiency.

TABLE OF CONTENTS

INTRODUCTION

Data Stream Clustering (DSC) is a critical operation in data stream mining, essential for various real-world applications such as network intrusion detection, social network analysis, and weather forecasting. DSC aims to group incoming data tuples based on their real-time attribute similarities. Besides achieving high clustering accuracy, ensuring processing efficiency is crucial for DSC algorithms.

Despite the extensive application and research in DSC, selecting an appropriate DSC algorithm for real-world workloads with diverse characteristics remains challenging for researchers and practitioners. This difficulty arises from the fundamental design choices of DSC algorithms, which involve different trade-offs and performance behaviors. These design choices are interdependent, making determining the most effective combinations complex.

The paper we summarize conducts an in-depth study of the critical design aspects of DSC algorithms, focusing on (1) data summarization structures, (2) window models, (3) outlier detection mechanisms, and (4) offline refinement strategies. Each aspect encompasses multiple design choices, leading to various DSC algorithms with different behaviors under varying workloads.

The paper implemented its methodology on four real-world and two synthetic workloads with diverse characteristics. The paper also devises a novel DSC algorithm named Benne, which integrates flexible design choices from the four key aspects. Benne achieves superior accuracy or efficiency compared to the state-of-the-art algorithms across all tested workloads.

Below is a brief description of the various four design aspects of DSC Algorithms:-

Table 1. Summary of design aspects

Design Aspect	Design Choices	Efficiency	Accuracy	Notes
Summarizing Data Structure	CFT	high	high	efficient data insertion and some additional operations for handling cluster evolution
	CoreT	low	High	need to rebuild the whole tree for updating lazily
	DPT	high	high	contain additional density information for clustering
	MCs	low	high	keep basic structure of CFT and additional time information; needs to search for every clusters during data insertion
	Grids	high	low	no need for frequent distance calculation but not so much accurate during data insertion
	AMS	low	High	need to frequently rebuild the structure
Window Model	LandmarkWM	depends	depends	difficult to determine a suitable landmark configuration
	SlidingWM	high	low	fixed window size results in fewer clusters for computation
	DampedWM	low	depends	process all the data points with a decay function
Outlier Detection	NoOutlierD	high	low	do not use any outlier detection mechanism
	OutlierD	high	high	detect outlier clusters through density and reduce the number of temporal clusters
	OutlierD-B	low	high	well handle cluster evolution among outliers and avoid cluster pollution but consumes much time to maintain the buffer
	OutlierD-T	high	high	prevent removing active clusters and be more focused on recent cluster information
	OutlierD-BT	low	high	improve the algorithms' ability for clustering evolving stream but still be low efficient in maintaining buffer
Offline Refinement	Refine	minor overhead	minor impact	any existing batch-based clustering algorithms may apply to refine results
	NoRefine	no impact	no impact	do not apply any further refinement

*(The table is sourced from the research paper)

METHODOLOGY

The devised novel DSC Algorithm - “Benne,” is compared with eight other standard DSC Algorithms. The paper implemented its methodology on four real-world (FCT, KDD99, Insects, Sensor) and two synthetic (EDS, ODS) workloads with diverse characteristics. Below is a brief description of the Algorithms and the datasets used for validation:-

Table 3. A summary of representative *DSC* algorithms and their design decisions. The year attribute for each algorithm is when it was first published.

Algorithm	Year	Summarizing Data Structure		Window Model	Outlier Detection	Offline Refinement
		Name	*Catalog*			
BIRCH [38]	1996	CFT	Hierarchical	LandmarkWM	OutlierD	NoRefine
CluStream [5]	2003	MCs	Partitional	LandmarkWM	OutlierD-T	Refine
DenStream [12]	2006	MCs	Partitional	DampedWM	OutlierD-BT	Refine
DStream [14]	2007	Grids	Partitional	DampedWM	OutlierD-T	NoRefine
StreamKM++ [4]	2012	CoreT	Hierarchical	LandmarkWM	NoOutlierD	Refine
DBStream [20]	2016	MCs	Partitional	DampedWM	OutlierD-T	Refine
EDMStream [19]	2017	DPT	Hierarchical	DampedWM	OutlierD-BT	NoRefine
SL-KMeans [10]	2020	AMS	Partitional	SlidingWM	NoOutlierD	NoRefine

Table 4. Characteristics differences of selected workloads. Note that the outliers column refers to whether there are outliers in the final clustering results.

Workload	Length	Dimension	Cluster Number	Outliers	Evolving Frequency
FCT [1]	581012	54	7	False	Low
KDD99 [36]	4898431	41	23	True	Low
Insects [35]	905145	33	24	False	Low
Sensor [2]	2219803	5	55	False	High
EDS	245270	2	363	False	Varying
ODS	100000	2	90	Varying	High

*(The table is sourced from the research paper)

Specifications of our evaluation platform:-

1) Processor - 12th Gen Intel(R) Core(TM) i5-1230U 1.00 GHz
2) Memory - 8GB RAM
3) OS - Windows Subsystem for Linux
4) Kernel - 5.15.153.1-microsoft-standard-WSL2
5) Compiler - gcc (Ubuntu 13.2.0-23 ubuntu4) 13.2.0

Software and Dependencies:-

We used Cmake Software to build automation testing. Also, dependencies like Boost Gflags were used to execute this project successfully.

EXPERIMENTAL RESULTS

We ran all the standard algorithms and “Benne” and obtained the following metrics:-

```
algo_id: 0
algo: "Birch"
workload: "CoverType"
num_points: 10000
dim: 54
num_clusters: 7
arr_rate: 0
max_in_nodes: 400
max_leaf_nodes: 100
distance_threshold: 600
seed: 1
coreset_size: 100
radius: 0.1
delta: 10
beta: 0.0021
buf_size: 500
alpha: 0.998
lambda: 1
clean_interval: 2500
min_weight: 6.91692e-323
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
cluster_size: 20
outlier_size: 0
win_us: 54
ds_us: 90402
out_us: 787
ref_us: 49
sum_us: 1000084
on_20: 0.199996
on_40: 0.399873
on_60: 0.599744
on_80: 0.799636
on_100: 0.999605
lat_us: 24.7412
et_s: 1.00003
qps: 9999.15
num_res: 20
cmm: 0
purity: 0.3798
nmi: 0
```

```
algo_id: 2
algo: "CluStream"
workload: "CoverType"
num_points: 3000
dim: 54
num_clusters: 7
arr_rate: 0
max_in_nodes: 3
max_leaf_nodes: 3
distance_threshold: 3550
seed: 10
coreset_size: 100
radius: 10
delta: 10
beta: 0.0021
buf_size: 100
alpha: 0.998
lambda: 1
clean_interval: 2500
min_weight: 4.67642e-310
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 2
time_window: 200
num_online_clusters: 7
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 7
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
win_us: 0
ds_us: 16788
out_us: 0
ref_us: 2076
sum_us: 300005
on_20: 0.059745
on_40: 0.119584
on_60: 0.179566
on_80: 0.239481
on_100: 0.299293
lat_us: 25.035
et_s: 0.299811
qps: 9999.82
num_res: 44
cmm: 0
purity: 0.298667
nmi: 0
```

```
algo_id: 6
algo: "DStream"
workload: "CoverType"
num_points: 3000
dim: 54
num_clusters: 7
arr_rate: 0
max_in_nodes: 3
max_leaf_nodes: 3
distance_threshold: 3550
seed: 1
coreset_size: 100
radius: 0.1
delta: 10
beta: 0.001
buf_size: 500
alpha: 0.998
lambda: 0.998
clean_interval: 2500
min_weight: -3.10894e-257
base: 2
cm: 15
cl: 0.001
grid_width: 13
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
num_grids: 677171456
gap: 74
win_us: 0
ds_us: 462064370
out_us: 0
ref_us: 0
sum_us: 302667
on_20: 0.059965
on_40: 0.12527
on_60: 0.182558
on_80: 0.247444
on_100: 0.302528
lat_us: 3902.57
et_s: 0.302667
qps: 9911.87
num_res: 0
cmm: 0
purity: 0
nmi: 0
```

```
algo_id: 3
algo: "DenStream"
workload: "CoverType"
num_points: 3000
dim: 54
num_clusters: 7
arr_rate: 0
max_in_nodes: 3
max_leaf_nodes: 3
distance_threshold: 3550
seed: 1
coreset_size: 100
radius: 0.1
delta: 10
beta: 0.25
buf_size: 500
alpha: 0.998
lambda: 0.25
clean_interval: 2500
min_weight: -3.10894e-257
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 20
mu: 5
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
win_us: 2168
ds_us: 264381
out_us: 10754
ref_us: 68868
sum_us: 423763
on_20: 0.081822
on_40: 0.145162
on_60: 0.206022
on_80: 0.273776
on_100: 0.354234
lat_us: 25077.6
et_s: 0.354895
qps: 7079.42
num_res: 519
cmm: 0
purity: 0.592
nmi: 0
```

```
algo_id: 5
algo: "EDMStream"
workload: "CoverType"
num_points: 8000
dim: 54
num_clusters: 7
arr_rate: 0
max_in_nodes: 3
max_leaf_nodes: 3
distance_threshold: 3550
seed: 1
coreset_size: 100
radius: 250
delta: 1500
beta: 0.0021
buf_size: 500
alpha: 0.998
lambda: 1
clean_interval: 2500
min_weight: -3.10894e-257
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
win_us: 953
ds_us: 660703
out_us: 10456
ref_us: 199
sum_us: 819330
on_20: 0.186956
on_40: 0.333504
on_60: 0.479903
on_80: 0.639651
on_100: 0.818549
lat_us: 8945.6
et_s: 0.819131
qps: 9764.07
num_res: 113
cmm: 0
purity: 0.519135
nmi: 0
```

```
algo_id: 7
algo: "SLKMeans"
workload: "CoverType"
num_points: 3000
dim: 54
num_clusters: 7
arr_rate: 0
max_in_nodes: 3
max_leaf_nodes: 3
distance_threshold: 3550
seed: 10
coreset_size: 100
radius: 0.1
delta: 10
beta: 0.0021
buf_size: 500
alpha: 0.998
lambda: 1
clean_interval: 2500
min_weight: -3.10894e-257
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 1
landmark: 1000
sliding: 100
outlier_distance_threshold: 5000
outlier_cap: 10
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
win_us: 2077
ds_us: 199943
out_us: 0
ref_us: 290
sum_us: 324634
on_20: 0.060022
on_40: 0.119819
on_60: 0.179728
on_80: 0.24923
on_100: 0.323842
lat_us: 4441.86
et_s: 0.324343
qps: 9241.18
num_res: 5
cmm: 0.825702
purity: 0.470333
nmi: 0
```

```
[       OK ] System.CluStream (337 ms)
[ RUN      ] System.DBStream
algo_id: 4
algo: "DBStream"
workload: "CoverType"
num_points: 581012
dim: 2
num_clusters: 363
arr_rate: 0
max_in_nodes: 3
max_leaf_nodes: 3
distance_threshold: 3550
seed: 1
coreset_size: 100
radius: 20
delta: 10
beta: 0.0021
buf_size: 500
alpha: 0.2
lambda: 0.998
clean_interval: 400
min_weight: 0.5
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
win_us: 3982377
ds_us: 214313
out_us: 63285
ref_us: 55295
sum_us: 6479608
on_20: 2.8692
on_40: 3.68778
on_60: 4.73377
on_80: 5.63246
on_100: 6.42427
lat_us: 2.02747e+06
et_s: 6.42431
qps: 89667.8
num_res: 41
cmm: 0
purity: 0.608433
nmi: 0
```

```
algo_id: 1
algo: "StreamKMeans"
workload: "CoverType"
num_points: 3000
dim: 54
num_clusters: 7
arr_rate: 0
max_in_nodes: 3
max_leaf_nodes: 3
distance_threshold: 3550
seed: 10
coreset_size: 600
radius: 0.1
delta: 10
beta: 0.0021
buf_size: 500
alpha: 0.998
lambda: 1
clean_interval: 2500
min_weight: -3.10894e-257
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 7
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
win_us: 0
ds_us: 224893
out_us: 0
ref_us: 7796216
sum_us: 8190544
on_20: 0.059382
on_40: 0.119289
on_60: 0.190791
on_80: 0.238989
on_100: 0.39421
lat_us: 33213.4
et_s: 0.394328
qps: 366.276
num_res: 600
cmm: 0
purity: 0.822333
nmi: 0
```

(DBStream ignored due to high qps value - not expected - very much deviation)

The newly devised algorithm - **“Benne,”** had the following metrics when run with different design aspects to maximize the accuracy and efficiency respectively:-

1) To achieve the **highest accuracy,** Benne (Accuracy) consists of **MCs** summarizing data structure, **LandmarkWM, OutlierD-B,** and **NoRefine**. The Algorithm achieved a **purity of 1** when run on an **EDS workload(synthetic)**. The results are shown below:-


```
algo_id: 32
algo: "G12"
workload: "EDS"
num_points: 10000
dim: 2
num_clusters: 363
arr_rate: 0
max_in_nodes: 100
max_leaf_nodes: 50
distance_threshold: 10
seed: 1
coreset_size: 100
radius: 0.1
delta: 10
beta: 0.0021
buf_size: 500
alpha: 0.998
lambda: 1
clean_interval: 2500
min_weight: 4.67642e-310
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 1000
outlier_cap: 5
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
cluster_size: 80
outlier_size: 6372
win_us: 1714
ds_us: 9004
out_us: 8863
ref_us: 541
sum_us: 1000610
on_20: 0.200005
on_40: 0.399861
on_60: 0.599823
on_80: 0.79973
on_100: 0.999547
lat_us: 6.56441
et_s: 1.00007
qps: 9993.9
num_res: 6452
cmm: 0
purity: 1
nmi: 0
```

2) In contrast, **Benne (Efficiency)** comprises **CFT** summarizing data structure, **LandmarkWM**, **OutlierD**, and **NoRefine to achieve the highest efficiency**. The results are shown below:-

```
algo_id: 23
algo: "G3"
workload: "EDS"
num_points: 45690
dim: 2
num_clusters: 75
arr_rate: 0
max_in_nodes: 100
max_leaf_nodes: 100
distance_threshold: 50
seed: 1
coreset_size: 100
radius: 0.1
delta: 10
beta: 0.0021
buf_size: 500
alpha: 0.998
lambda: 1
clean_interval: 2500
min_weight: 4.67642e-310
base: 2
cm: 5
cl: 0.8
grid_width: 12
min_points: 10
epsilon: 50
mu: 7
num_last_arr: 60
time_window: 6
num_online_clusters: 10
delta_grid: 0.2
num_samples: 100
landmark: 1000
sliding: 10
outlier_distance_threshold: 500
outlier_cap: 2000
outlier_density_threshold: 100
neighbor_distance: 200
k: 2
run_offline: 1
obj: 0
queue_size_threshold: 10000
dim_threshold: 30
variance_threshold: 100
outliers_num_threshold: 200
outliers_dist_threshold: 50
cluster_size: 89
outlier_size: 0
win_us: 70
ds_us: 17813
out_us: 18377
ref_us: 43
sum_us: 2172410
on_20: 0.91408
on_40: 1.82765
on_60: 2.15548
on_80: 2.16386
on_100: 2.17234
lat_us: 5314.83
et_s: 2.17237
qps: 21031.9
num_res: 89
cmm: 0
purity: 0.786006
nmi: 0
```

DISCUSSION AND CONCLUSION

Below are some key results from the paper:-

1. **Accuracy and Efficiency Tradeoff**:
    - Maximum accuracy and maximum efficiency cannot be achieved simultaneously. There is always a tradeoff between the two.
2. **Design Aspects**:
    - The algorithm's accuracy and efficiency are influenced by its design aspects, which include:
        - **Summarizing Data Structure**: Hierarchical summarizing data structures enhance efficiency, whereas partitional structures improve accuracy.
        - **Window Model**: The effectiveness of a window model is highly dependent on the evolution of clusters over time.
        - **Outlier Detection**: Implementing outlier detection mechanisms, particularly those utilizing buffers, significantly enhances clustering accuracy.
        - **Offline Refinement**: In most cases, offline refinement is unnecessary as it does not improve clustering results.

These insights highlight the importance of considering the tradeoffs between accuracy and efficiency when designing and selecting data clustering algorithms.

Github Link of repository containing our project :-
https://github.com/RohitBohra-001/Deakin-FTP-Project-Submission

ACKNOWLEDGMENTS

We express our deepest gratitude to **Dr. Ye Zhu** for providing us with this incredible opportunity and their invaluable guidance throughout this research project. Their expertise and insights have been instrumental in the completion of this work.

Additionally, We are grateful to the **Indian Institute of Technology, Kharagpur, India,** and **Deakin University, Australia,** for providing the necessary resources and support. This project would not have been possible without their assistance.

REFERENCES

1) Xin Wang, Zhengru Wang, Zhenyu Wu, Shuhao Zhang, Xuanhua Shi, and Li Lu. 2023. Data Stream Clustering: An In-depth Empirical Study. Proc. ACM Manag. Data. 1, 2, Article 162 (June 2023), 26 pages. https://doi.org/in C++in C++ 10.1145/3589307

2)https://github.com/intellistream/Sesame