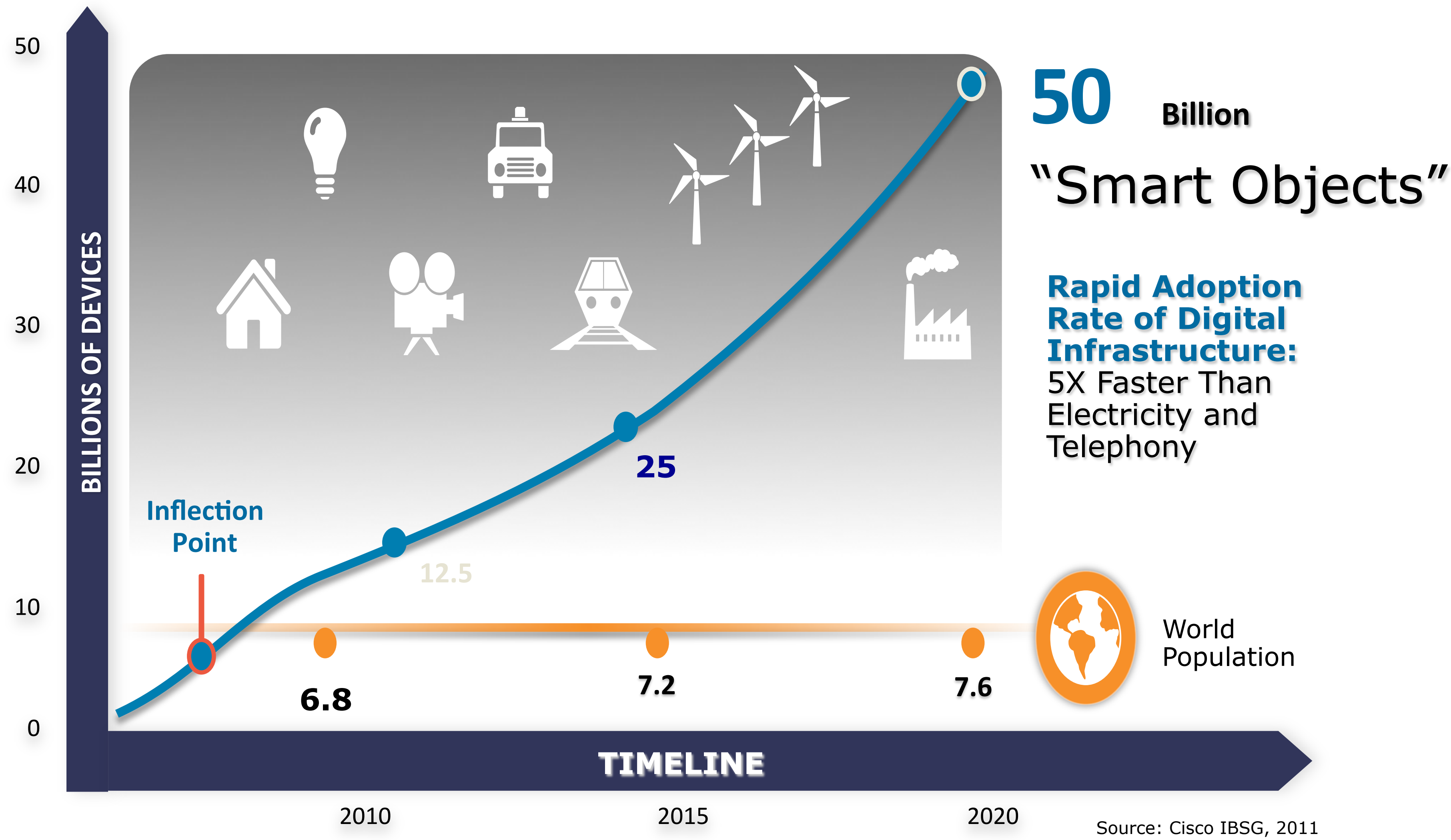
Data Stream Clustering for IoT

Gabriela Nicolescu
Heterogeneous Embedded Systems Laboratory
Polytechnique Montréal

gabriela.nicolescu@polymtl.ca

Evolution of IoT connected devices



Source: Cisco IBSG, 2011

Data in Internet of Things

- IoT networks will generate more than 400 zettabytes (trillion gigabytes) of data a year by 2018 (CISCO, 2015)

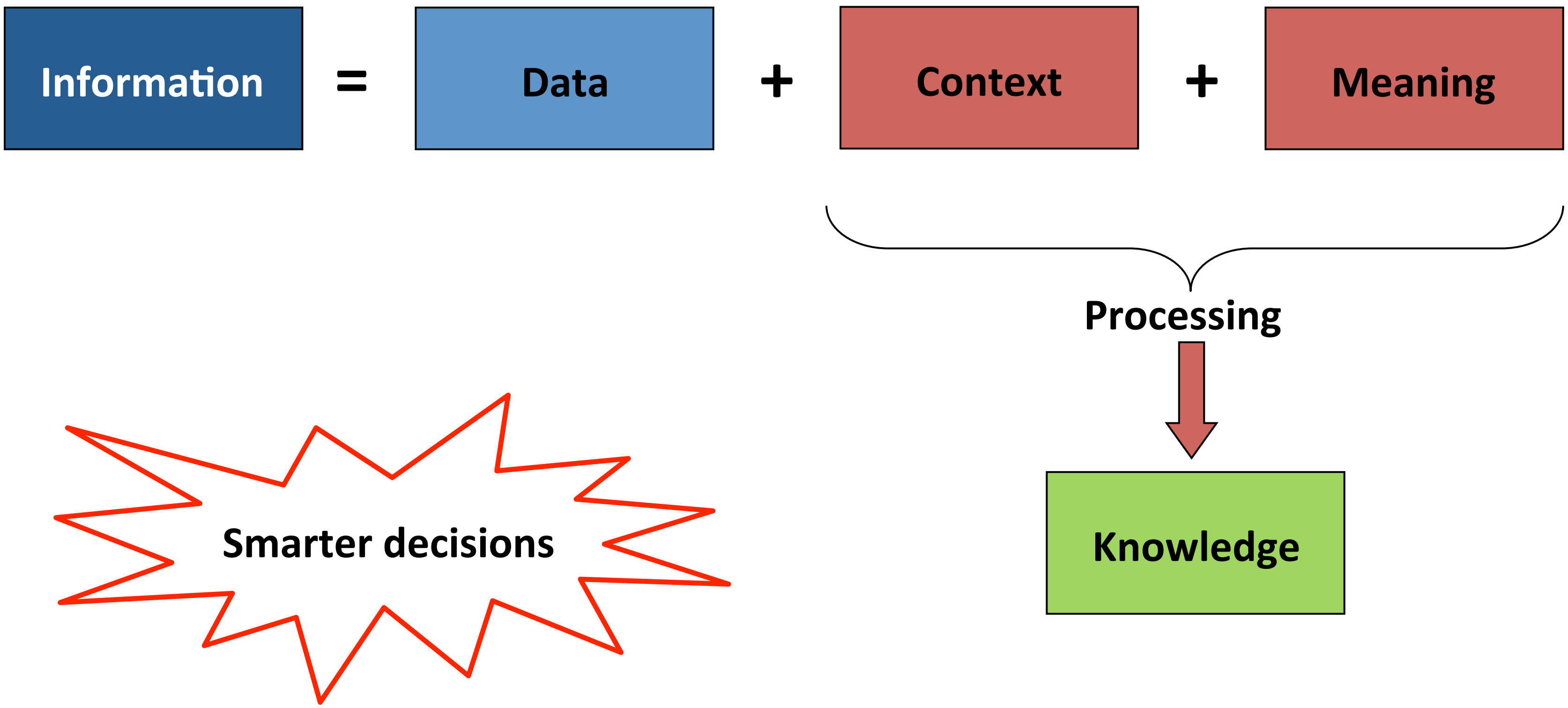
- Data generated by IoT devices **arrive continuously** and needs to be **processed on the fly**

  ➢ **Streaming data**

Examples of streaming data

- Sensor, monitoring & surveillance: video streams, RFIDs
- Security monitoring
- Web logs and Web page click streams
- Telecommunication calling records
- Business: credit card transaction flows
- Network monitoring and traffic engineering
- Financial market: stock exchange
- Engineering & industrial processes: power supply & manufacturing

  ➢ **Massive data sets**

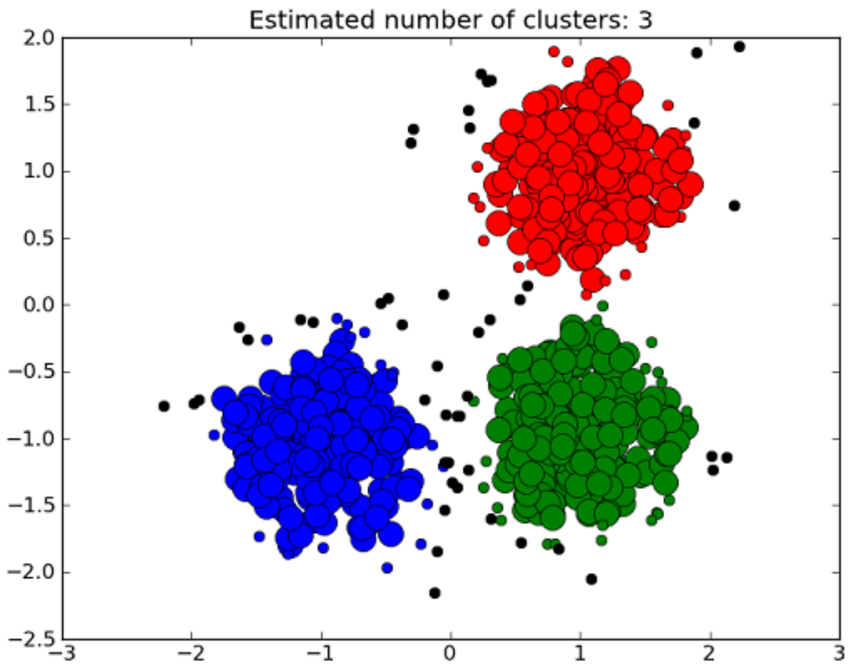


- ✓ **Advanced data mining technique required**
- ✓ Streaming data arrive continuously, is unbounded and non-stationary
- ✓ Data is **scanned once** and hidden patterns are constructed **online**

Need for on-line data mining

- Data stream clustering is the online process classifying a group of abstract objects

- Most used clustering techniques
  - Density-based
  - Partitioning
  - Hierarchical
  - Grid-based



	Clustering Technique	Shape of Cluster	No. of clusters	Outlier's handling	Variable Density	Time Complexity
DBSCAN	Density-based	Arbitrary	X	✓	X	$O(n \log n)$
Incremental DBSCAN	Density-based	Arbitrary	X	✓	X	$O(n) + O(n \log n)$
OPTICS	Density-based	Arbitrary	X	✓	X	$O(n \log n)$
DENCLUE	Density-based	Arbitrary	X	✓	X	$O(n \log n)$
Sequential K-means	Partitioning	Hyper-Spherical	Required	X	-	$O(n^{dk+1})$
CLARANS	Partitioning	Hyper-Spherical	Required	✓	-	$O(n^2)$
CURE	Hierarchical	Non-spherical	X	✓	X	$O(n^2 \log n)$
CHAMELEON	Hierarchical	Arbitrary	Required	X	✓	$O(n \log + nk + k^2 \log k)$
BIRCH	Hierarchical	Hyper-Spherical	Required	✓	✓	$O(n)$
STING	Grid-based	Arbitrary	X	✓	✓	$O(c)$
Wave Cluster	Grid-based	Arbitrary	X	✓	✓	$O(n)$

n = No. of data samples d = no. of data dimensions
k = no.of clusters c = no. of grid cells

Proposed Algorithm – Considerations

- Data streams are **continuous and unbounded**
- Data samples are **linearly scanned** (processed) only once before being discarded
- **No** assumption or prior knowledge of the number of clusters
- Data stream flows are grouped on **arbitrary shaped clusters**
- Ability to **handle outliers**
- Algorithm **scalability** to the number of incoming data samples

Clustering Steps

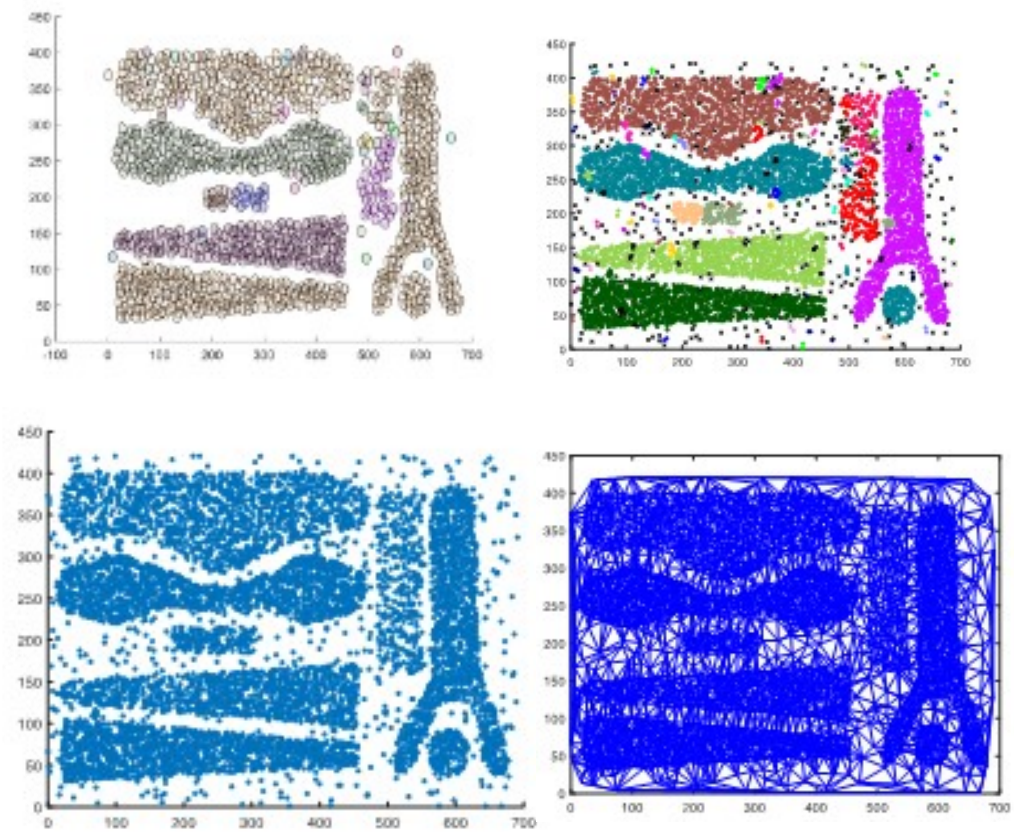
Neighborhood discovering using **incremental Delaunay triangulation**

↓

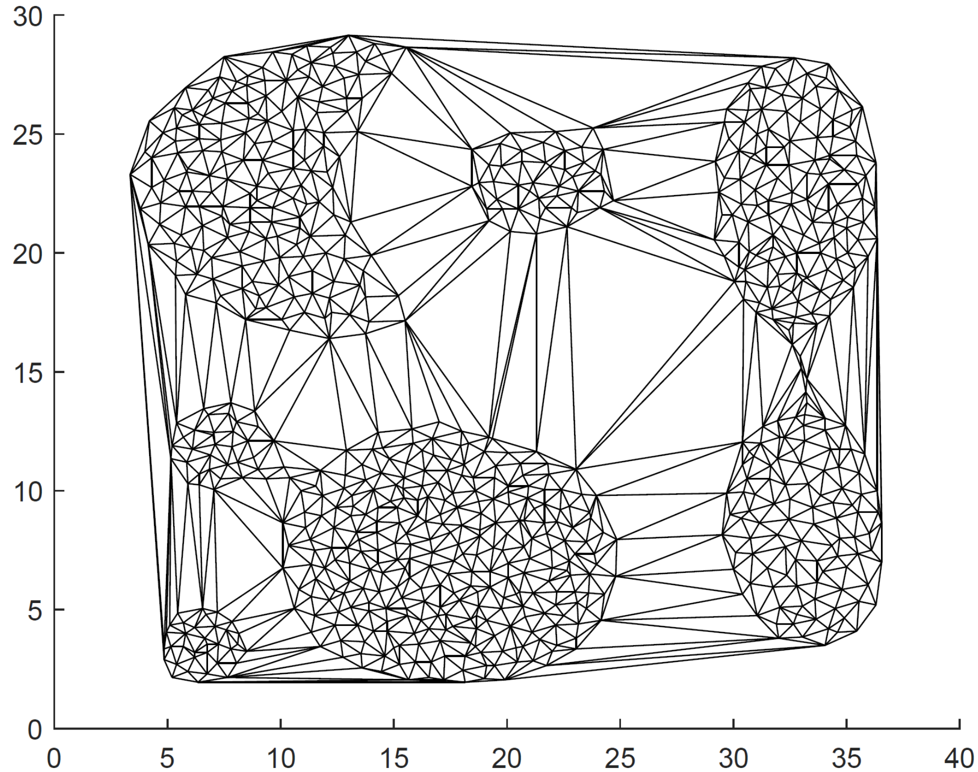
Micro-clustering

↓

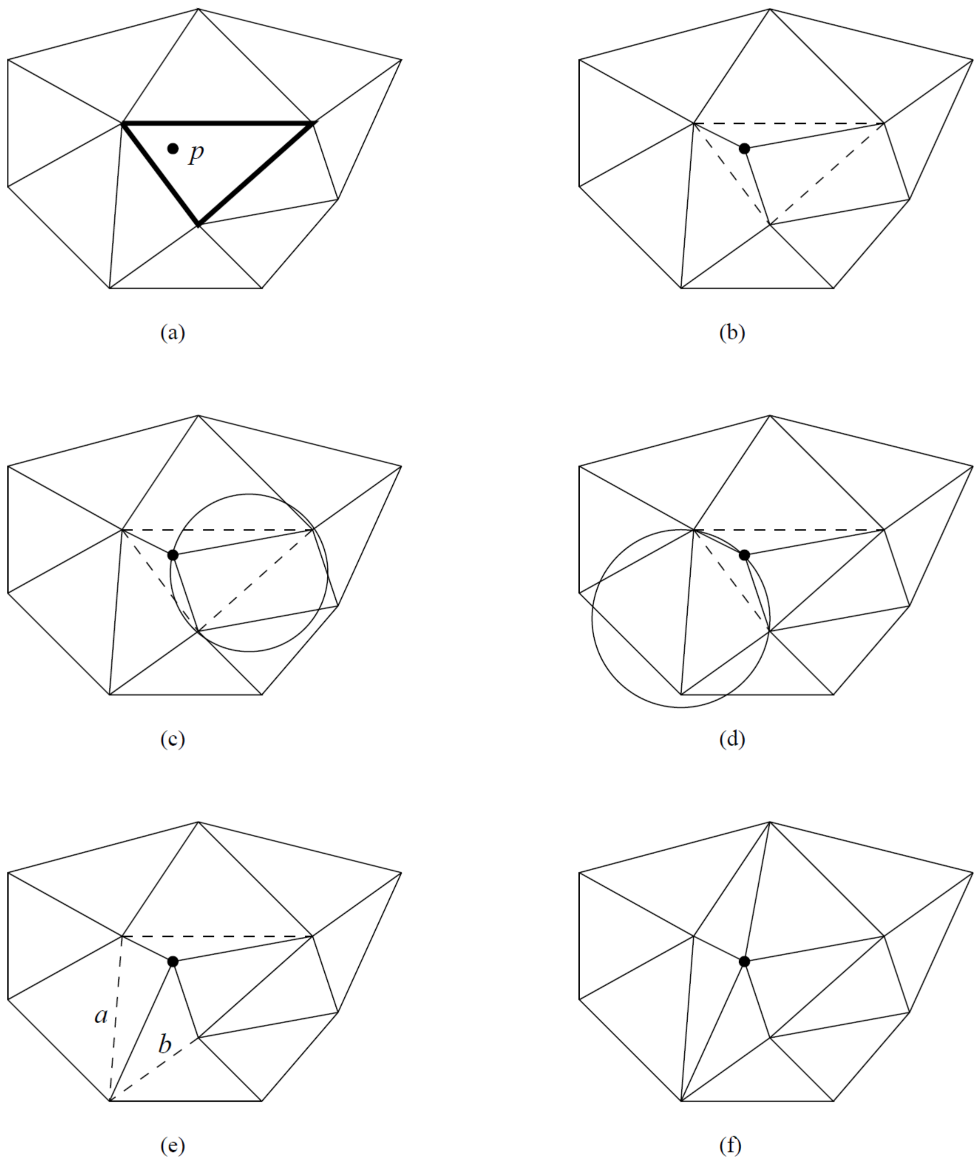
Re-clustering



Neighbourhood discovering



Construction of spatial proximity relationships using incremental Delaunay triangulation of a simulated dataset. The resulting clustering shows the potential clusters of the dataset.

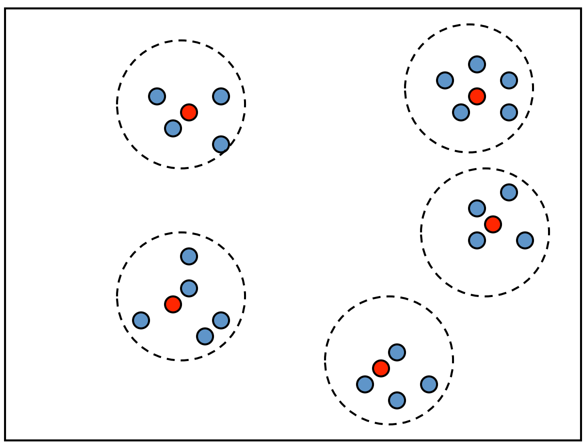


(a) (b) (c) (d) (e) (f)

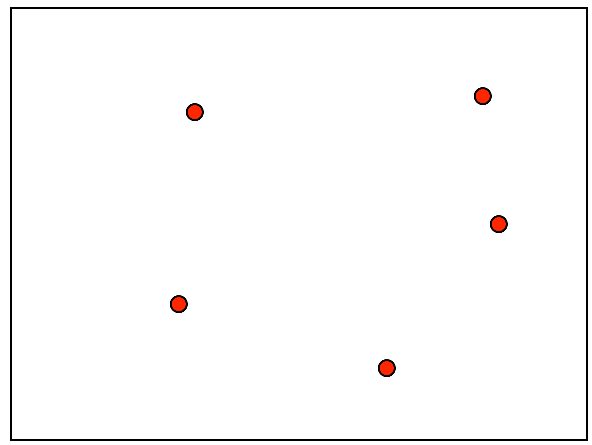
a. The triangle containing the new point *p* is located
b. New edges are created to connect p to the vertices of the containing triangle
c. The old edges of the triangle are inspected to verify that they still satisfy the empty circumcircle condition. If the condition is satisfied the edge remains unchanged.
d. If it is violated the offending edge is flipped, that is, replaced by the other diagonal of the surrounding quadrilateral.
e. In this case two more edges become candidates for inspection
f. The process continues until no more candidates remain, resulting in the triangulation.

Micro-clusters Building

- A Micro-Cluster is a set of individual data points that are close to each other and will be treated as a single unit in further Macro-clustering or re-clustering stage.



Snapshot of data groups



Snapshot of Micro-Clusters in memory

Micro-clusters building

What to Store in a Micro-Cluster ?

- Effcient data compression in a set of micro-clusters (Snapshot of data groups that keeps changing over time as new samples arrive)
    - *Micro-cluster id*
    - *Macro-cluster id*
    - *Centroid coordinates*
    - *Maturity index*
    - *Closest neighbor*
    - *Maturity threshold*
    - *Magnetic distance*

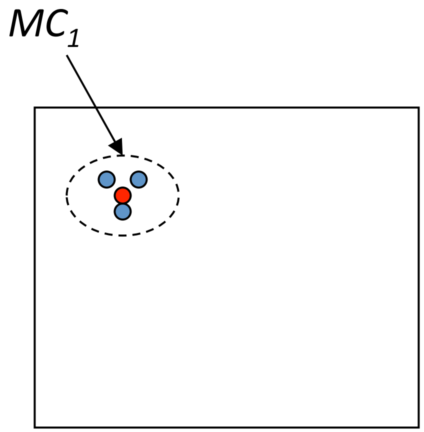
Micro-clusters building

- How to deal with a new incoming data point?
  1. Join one of the old micro-clusters if the data point is within the closest micro-cluster magnetic field distance

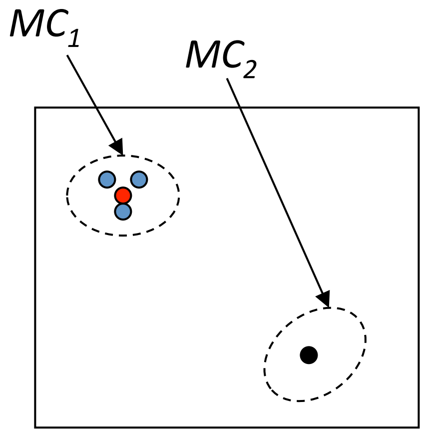
  2. Create a new micro-cluster by its own if it falls in an empty region

Key idea: Additivity Property

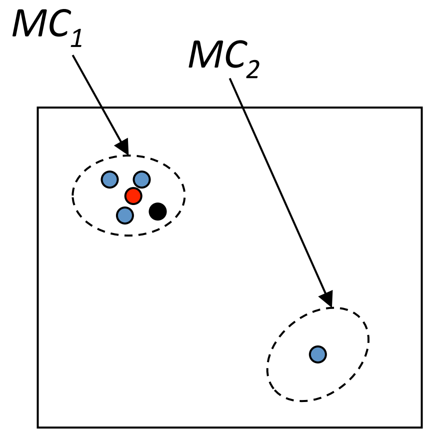
Micro-clusters building



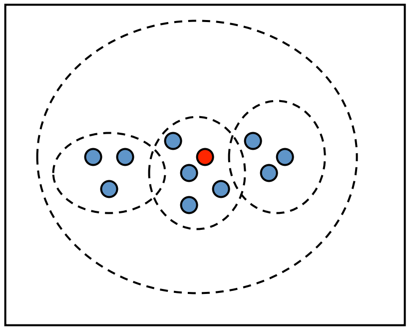
Snapshot of micro-cluster at instant t



New incoming data point falls in an empty space

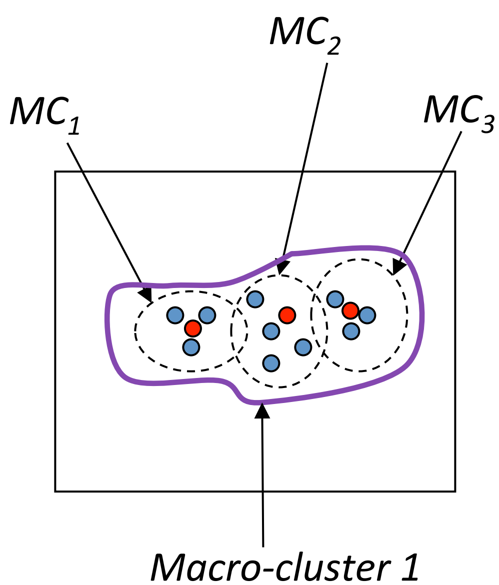


New incoming data point falls within a micro-cluster magnetic distance



Re-clustering to construct macro-clusters

Re-culstering

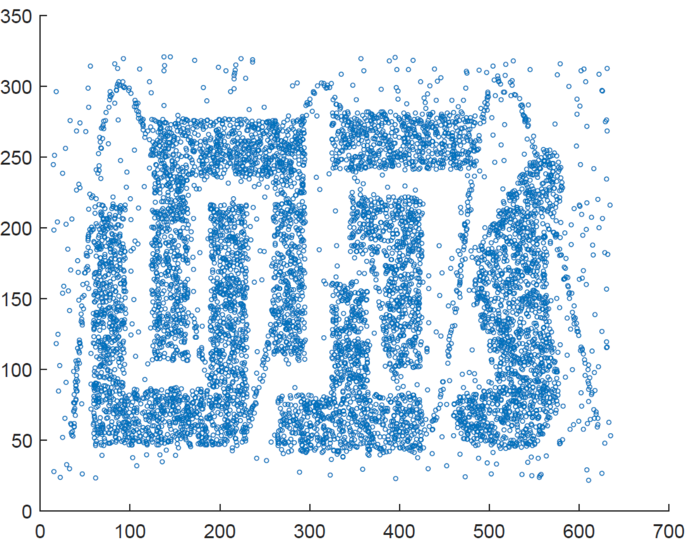


Macro-cluster obtained by merging three micro-clusters MC_1, MC_2, MC_3

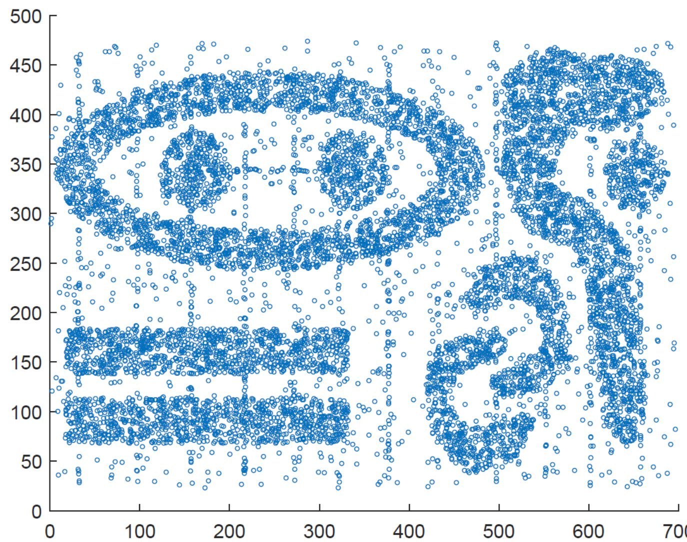
We assume that the maturity threshold of a micro-cluster is equal to 3

- If the micro-cluster of the recently processed data point reaches the **maturity threshold**, it is considered for the re-clustering phase
- The list of micro-clusters within the **macro-magnetic attraction field** but currently belonging to another macro-cluster is found
- All the micro-clusters found in this region become members of the data point macro-cluster
- These members will be **re-clustered** to be part of the updated micro-cluster's macro-cluster.

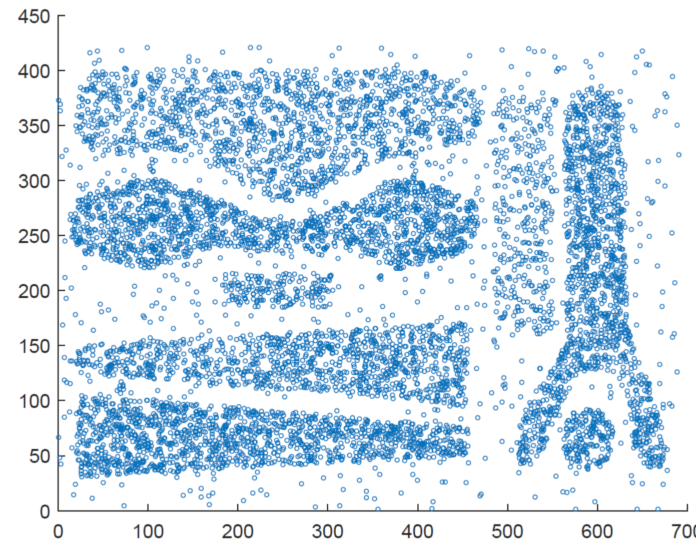
Data Sets for Clustering Evaluation



(a) Dataset DS1

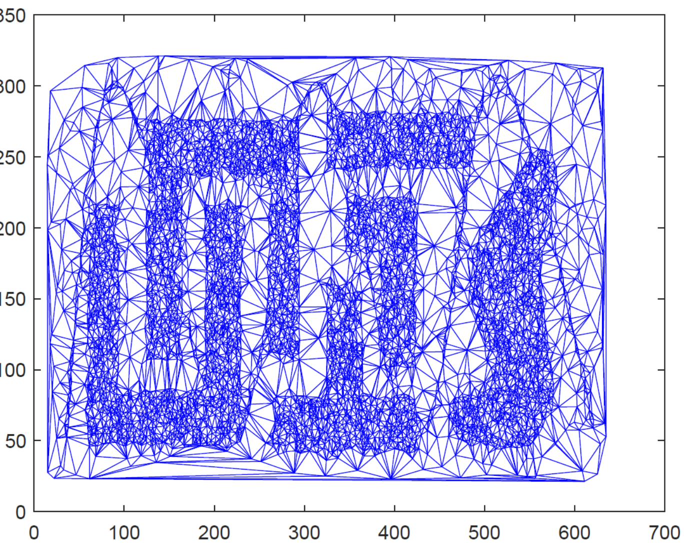


(b) Dataset DS2

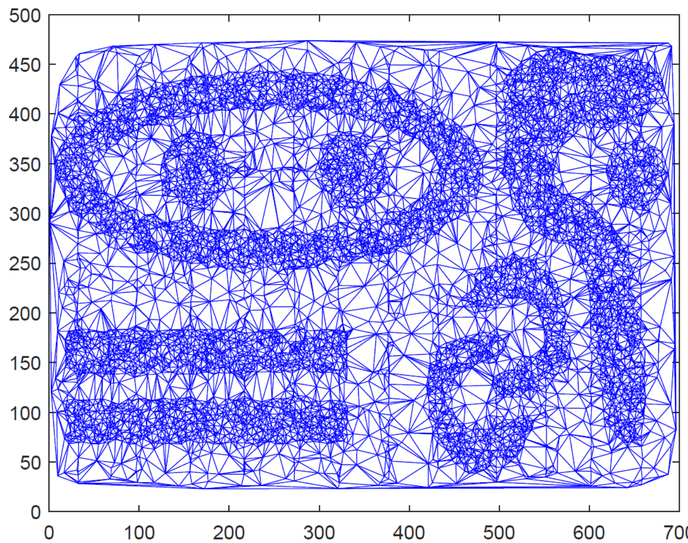


(c) Dataset DS3

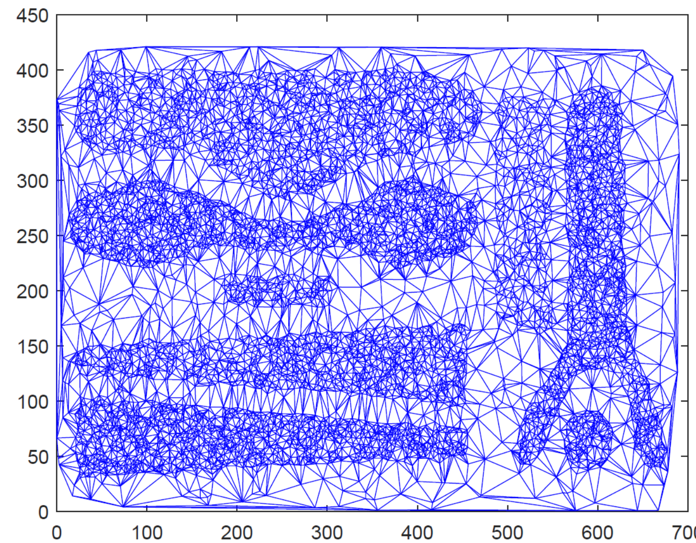
Triangulation



(d) Delaunay triangulation of DS1

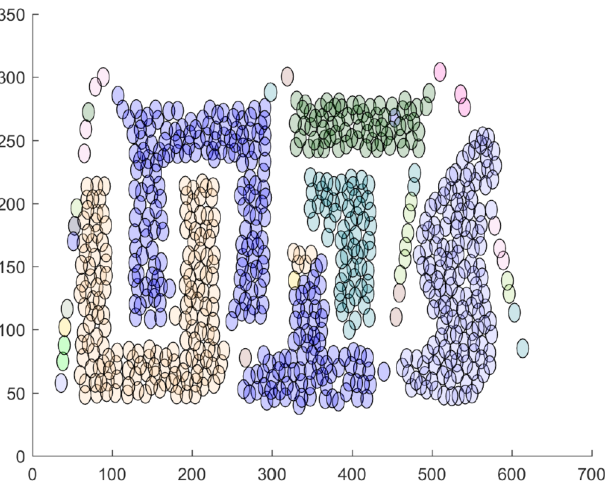


(e) Delaunay triangulation of DS2

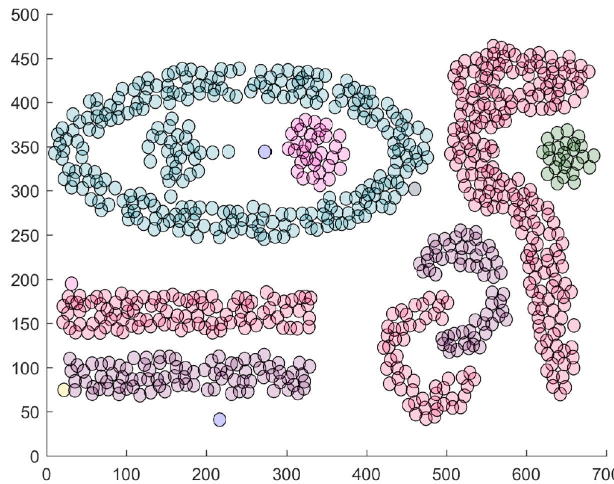


(f) Delaunay triangulation of DS3

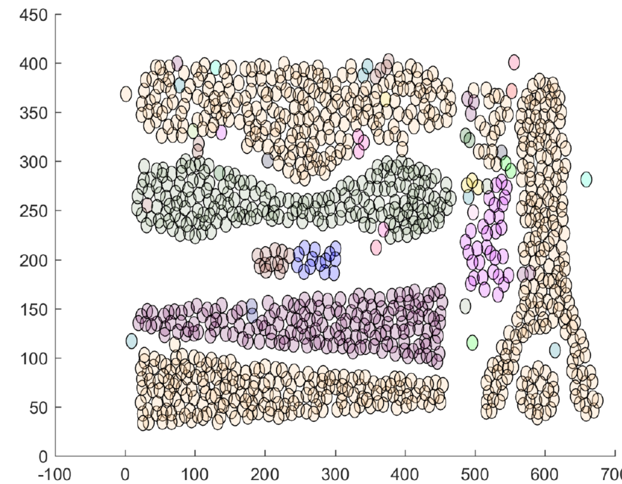
Micro-clusters



(g) Created micro-clusters of DS1

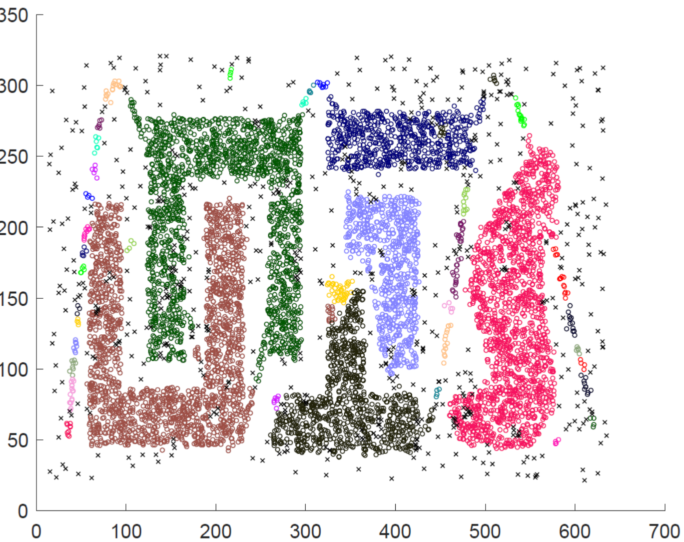


(h) Created micro-clusters of DS2

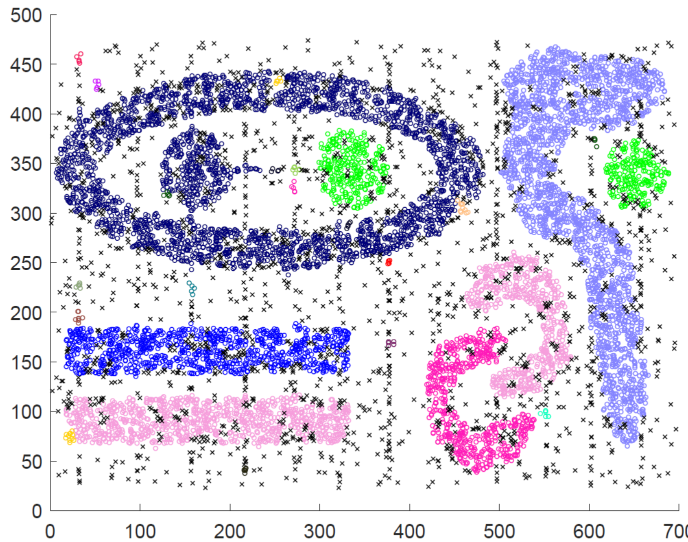


(i) Created micro-clusters of DS3

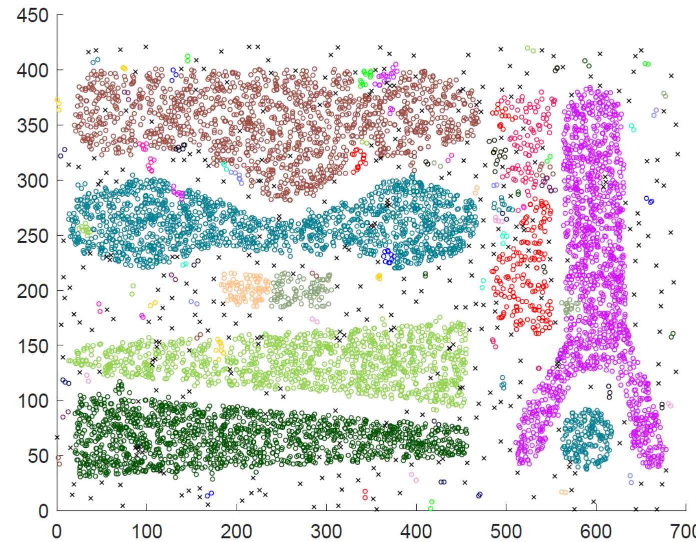
Macro-clusters



(j) DS1 Placed into macro-clusters

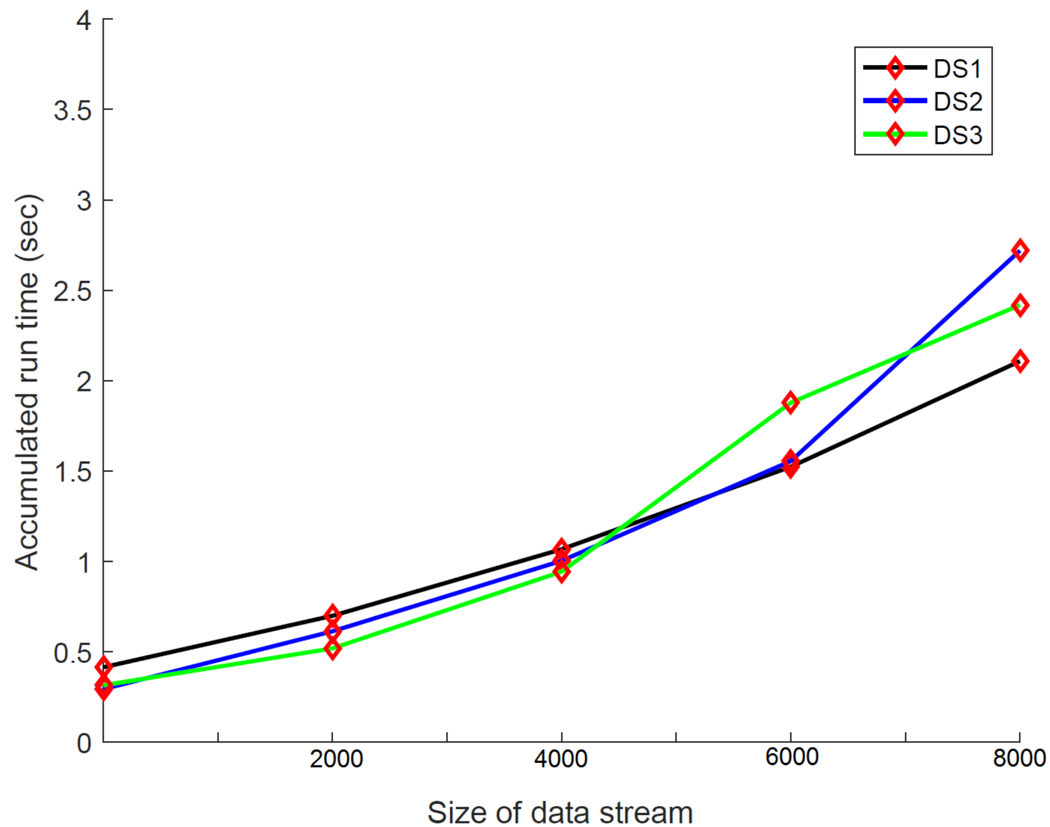


(k) DS2 Placed into macro-clusters

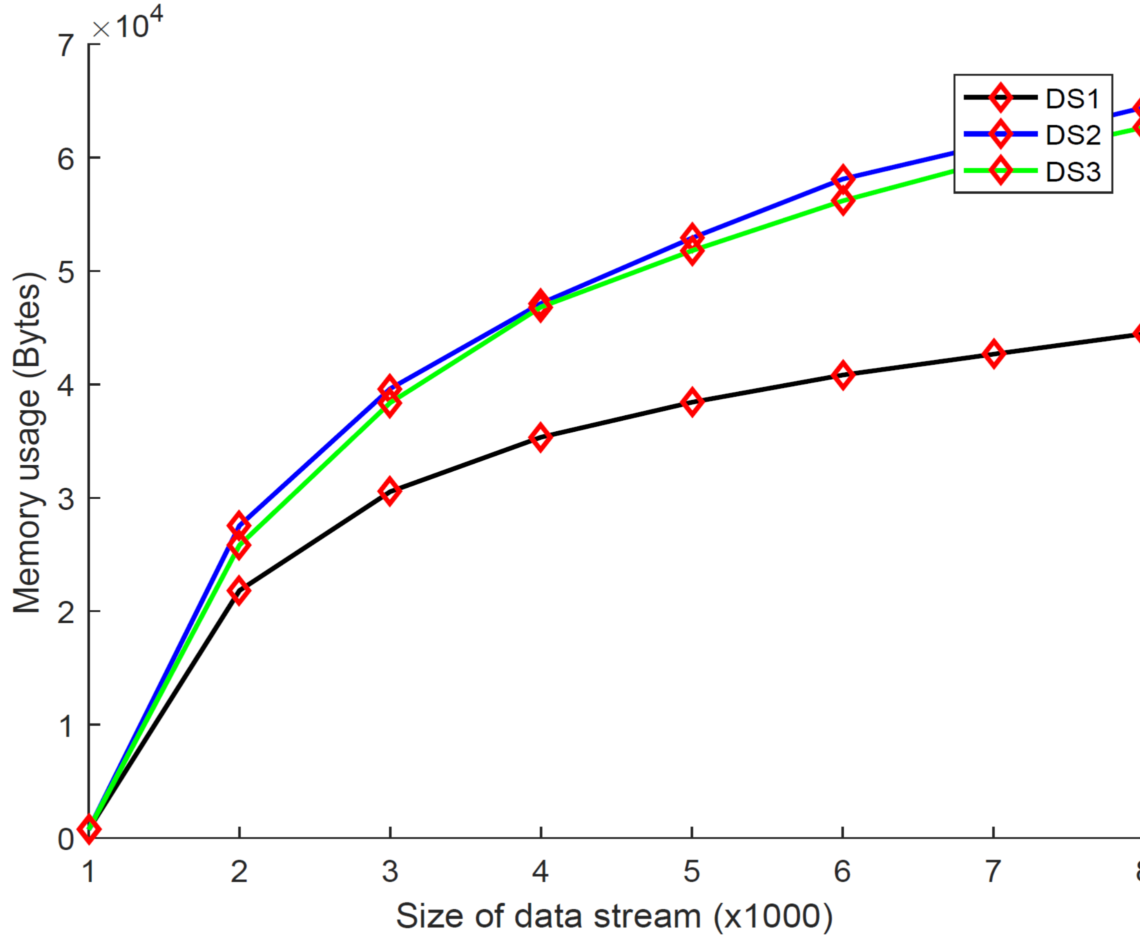


(l) DS3 Placed into macro-clusters

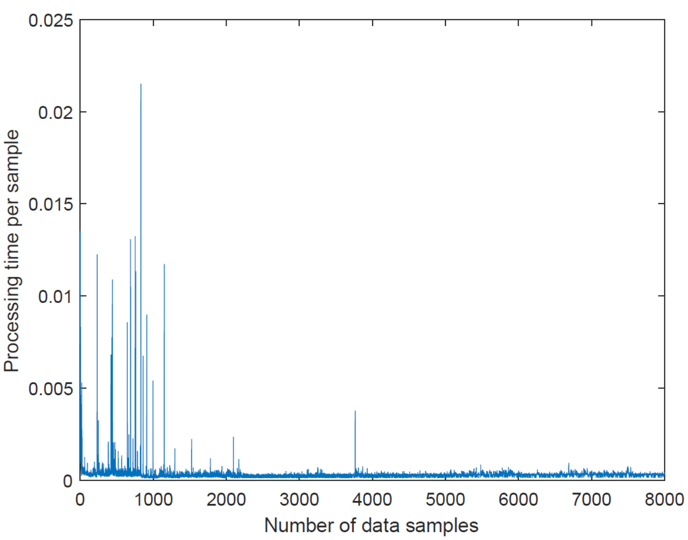
Scalability



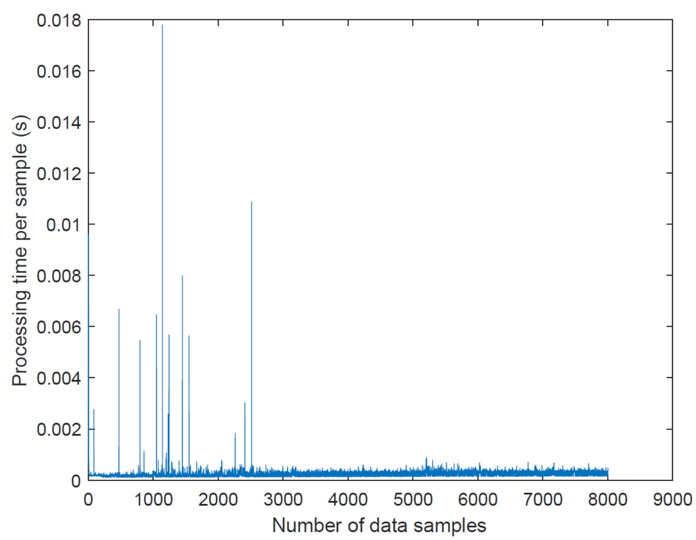
Memory Usage



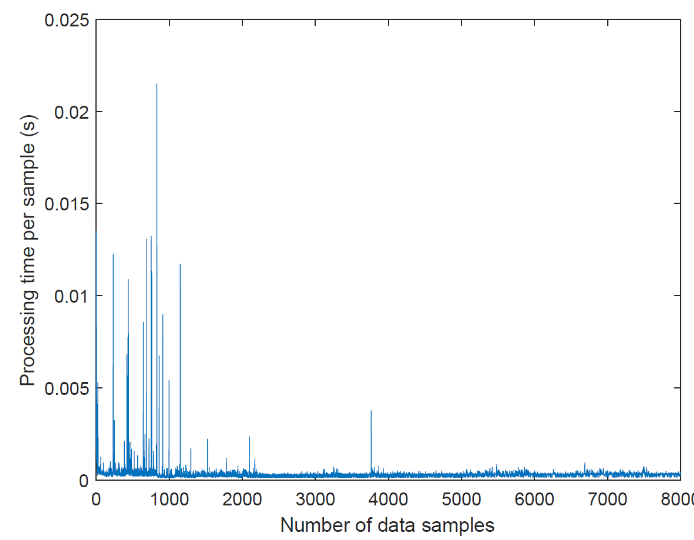
Processing Time



(a) DS1



(b) DS2



(c) DS3

Conclusion

- Processing streaming data in real-time environments requires new techniques in data mining
- Traditional offline methods appropriate only for resident data stored in large data repositories and consequently cannot address the problem of a continuous supply of data
- We propose a fully online and efficient incremental Delaunay triangulation-based data stream clustering al-gorithm
  - Time and memory optimization