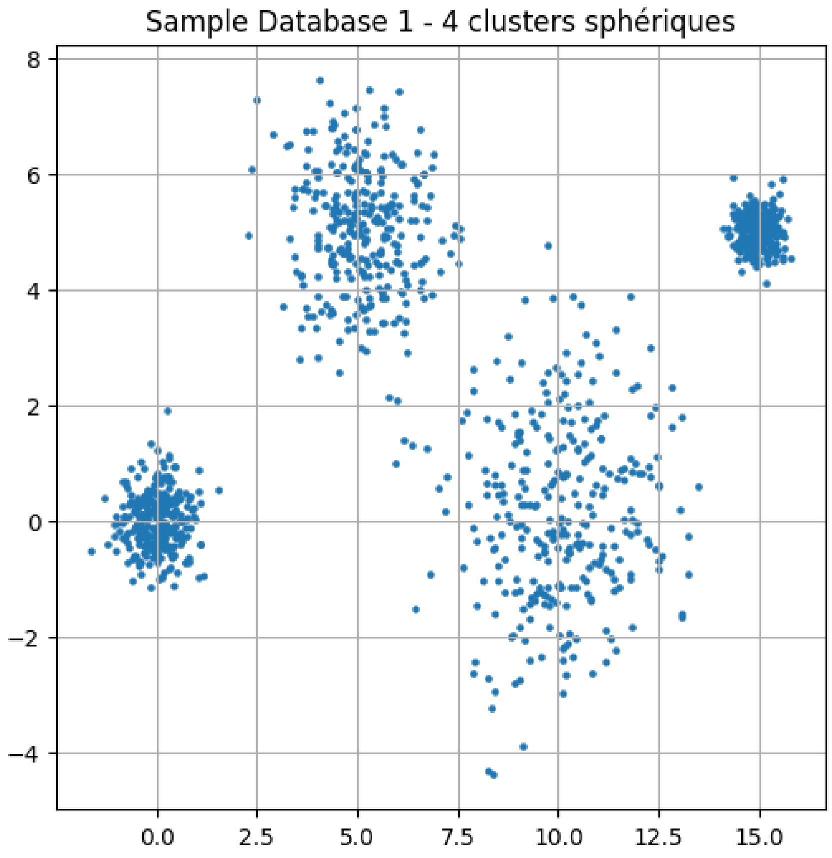
Clustering with DBSCAN

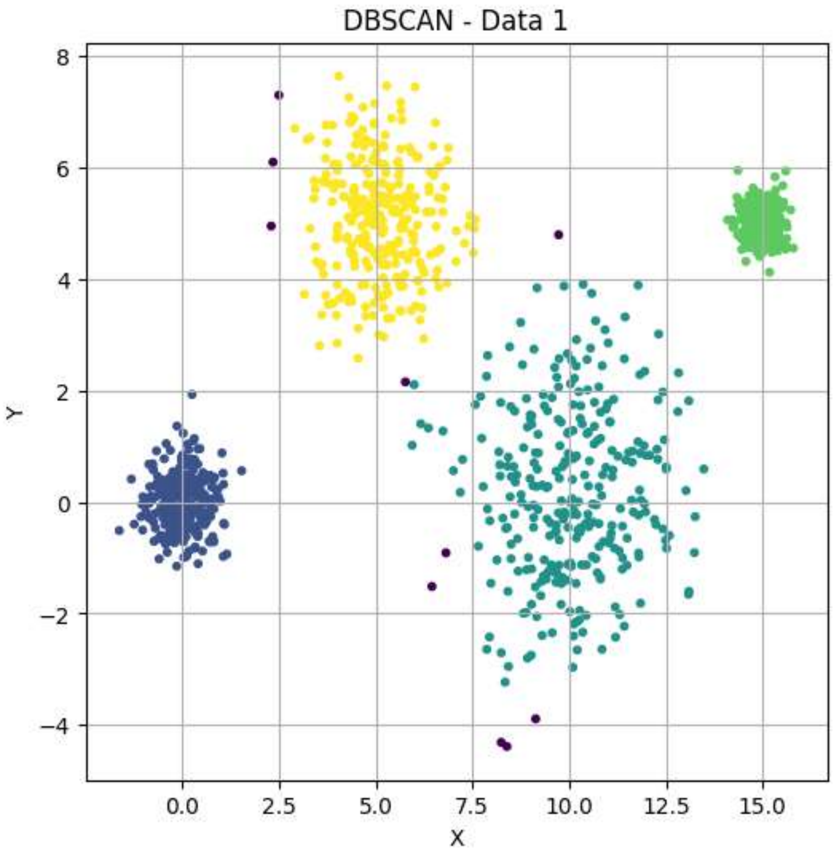
Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu

Yacine ZALFANI - Samuel METIN

Presentation of the article

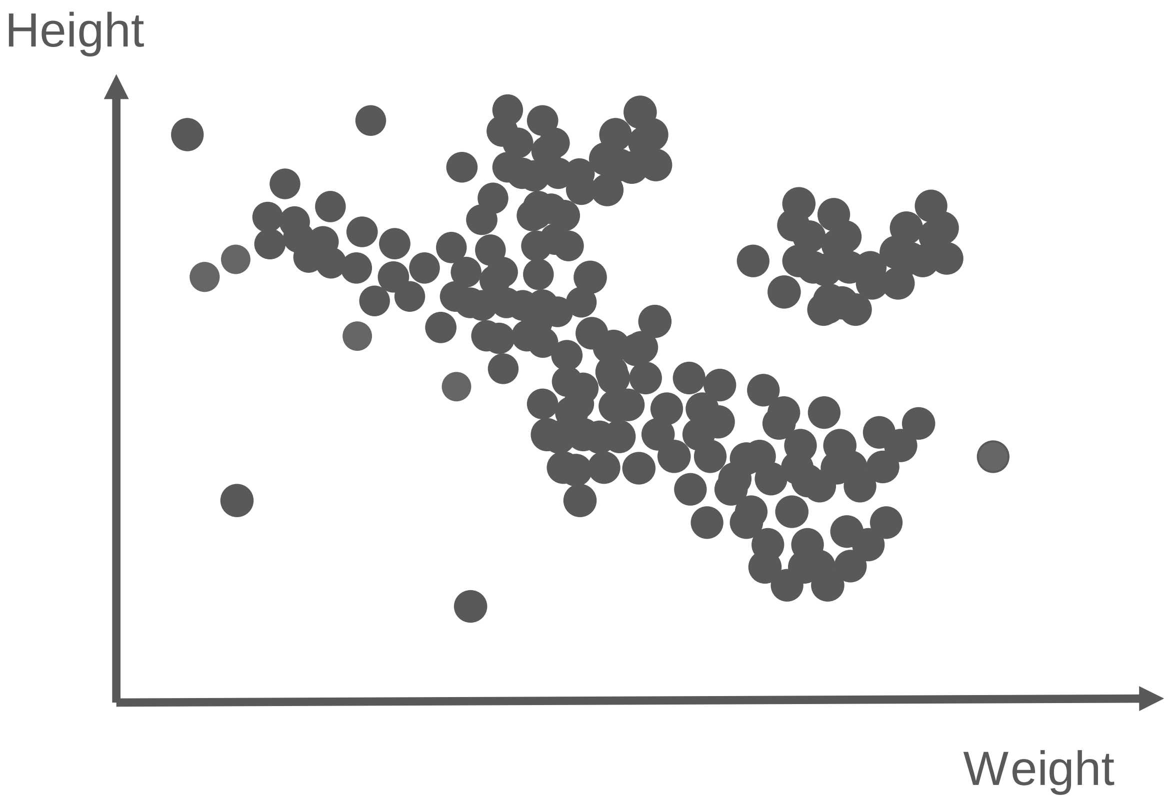


Sample Database 1 - 4 clusters sphériques
8
6
4
2
0
−2
−4
0.0
2.5
5.0
7.5
10.0
12.5
15.0

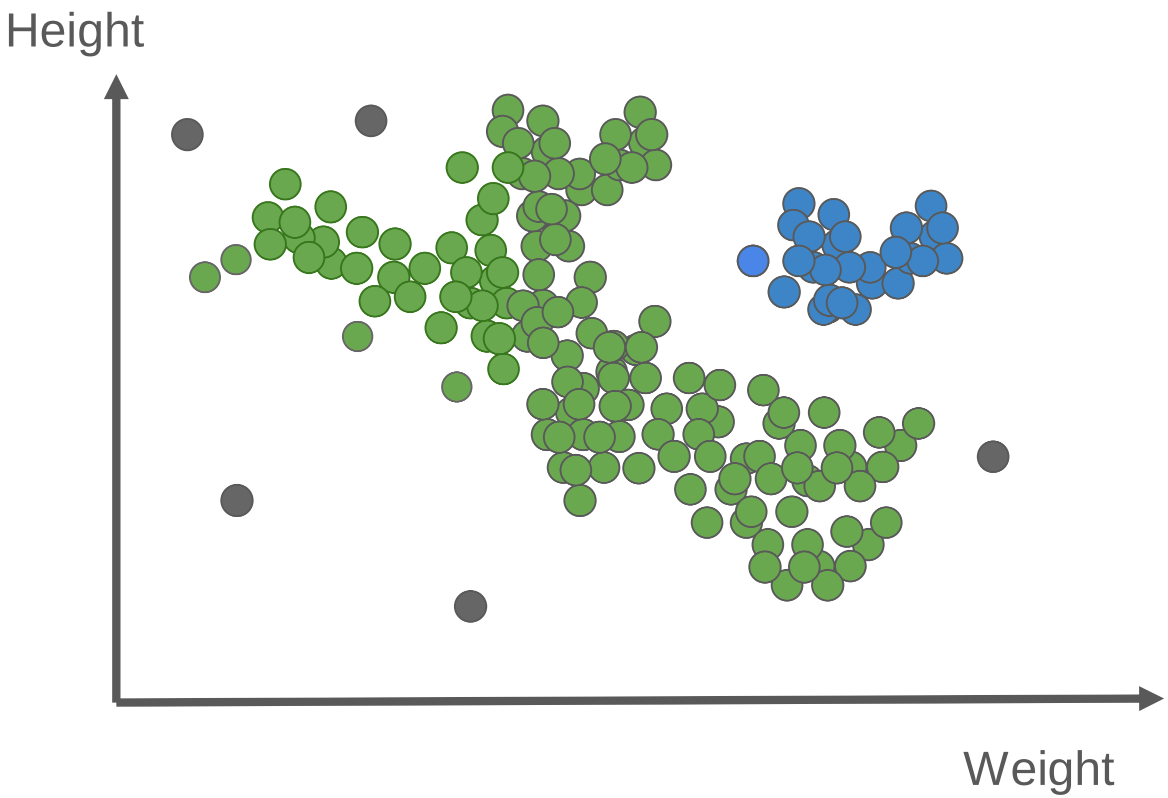


DBSCAN - Data 1
Y
X
8
6
4
2
0
−2
−4
0.0
2.5
5.0
7.5
10.0
12.5
15.0

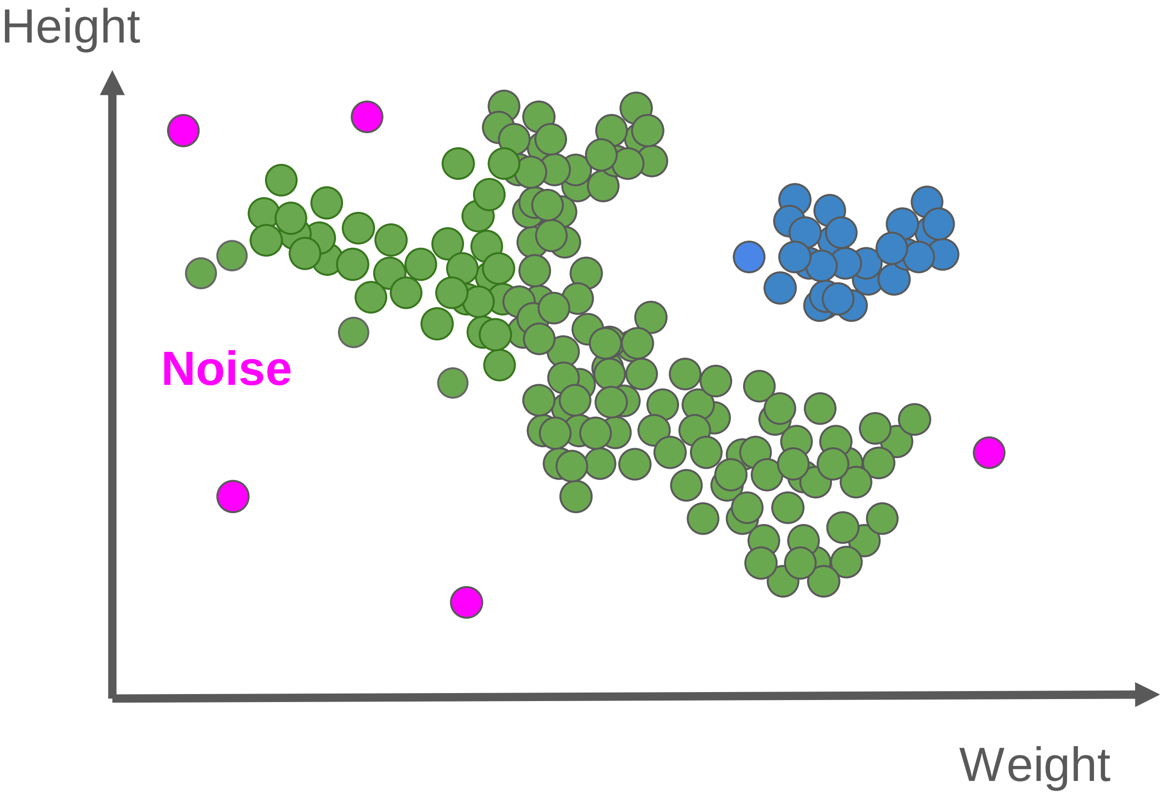
	Weight	Height
Person 1	45	140
Person 2	56	160
Person 3	90	190
…	…	…

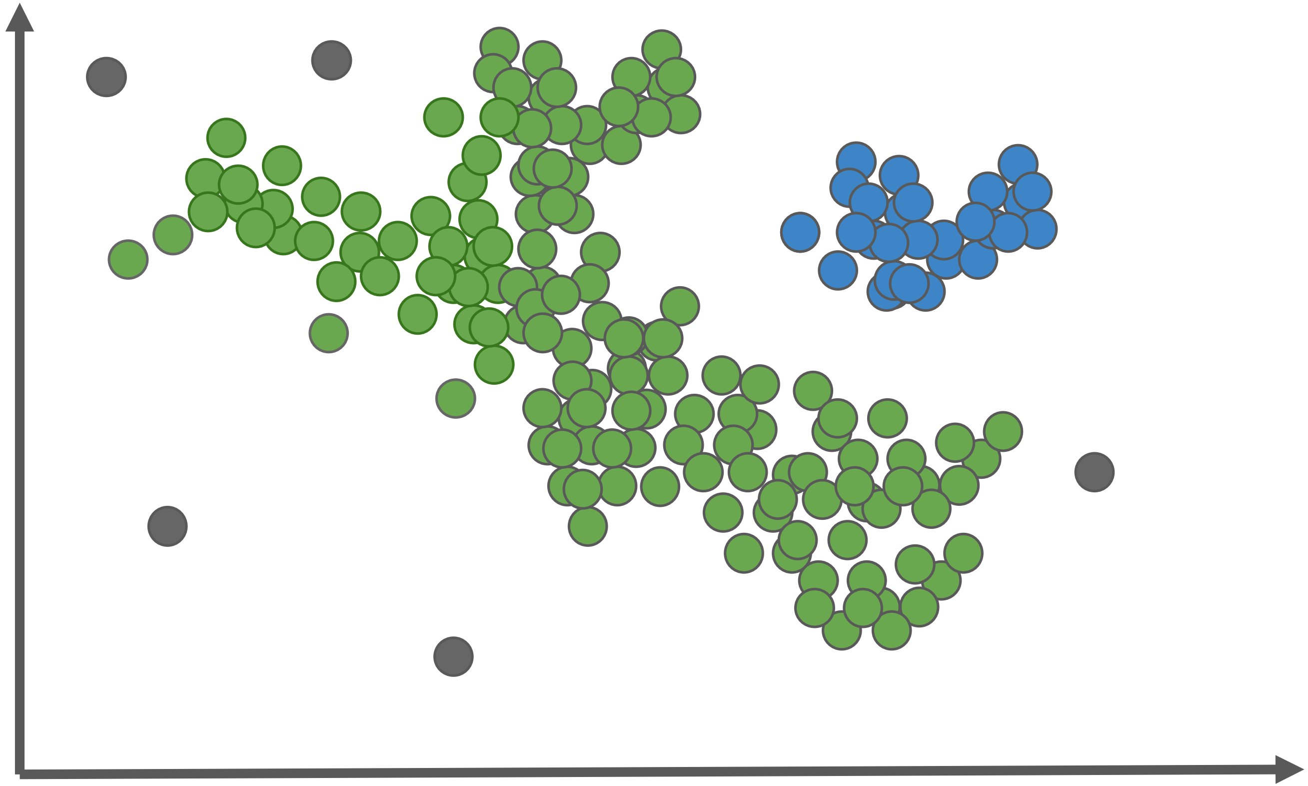


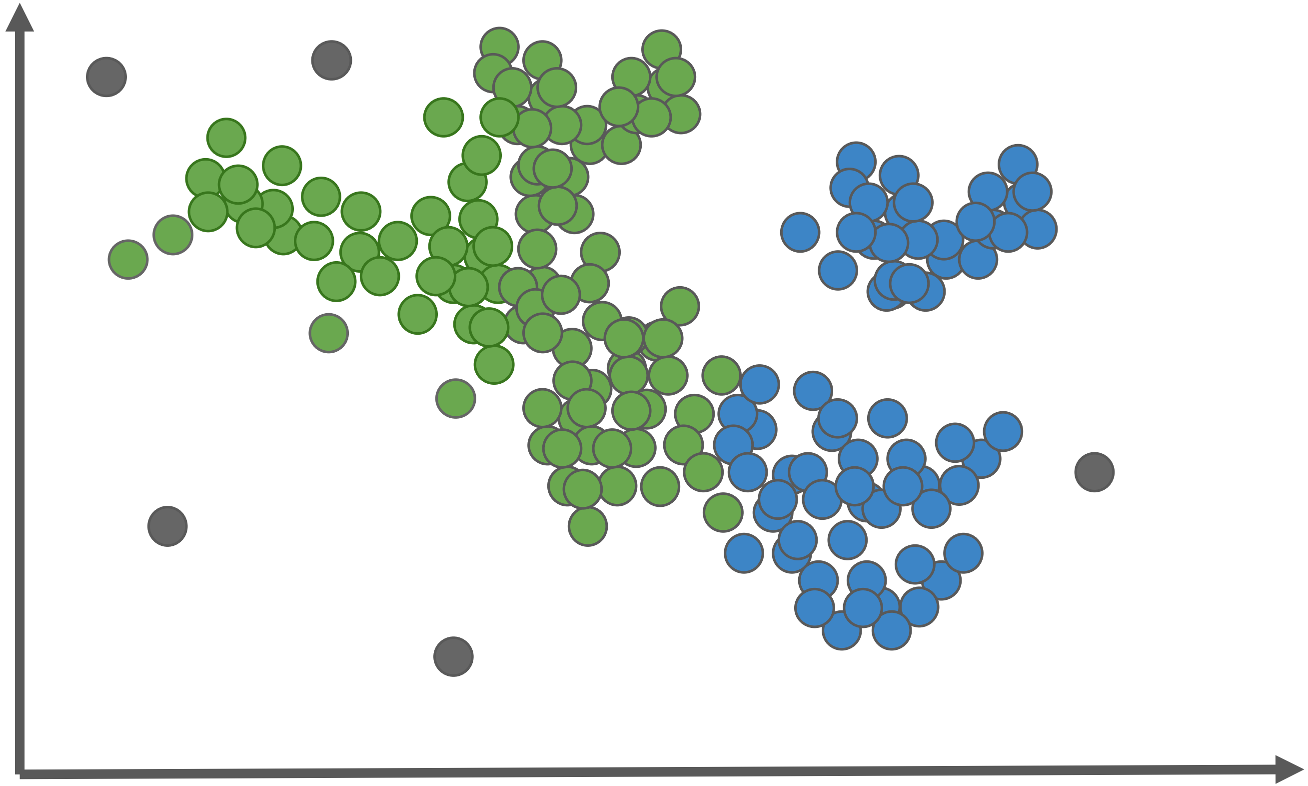
	Weight	Height
Person 1	45	140
Person 2	56	160
Person 3	90	190
…	…	…



	Weight	Height
Person 1	45	140
Person 2	56	160
Person 3	90	190
...	...	...







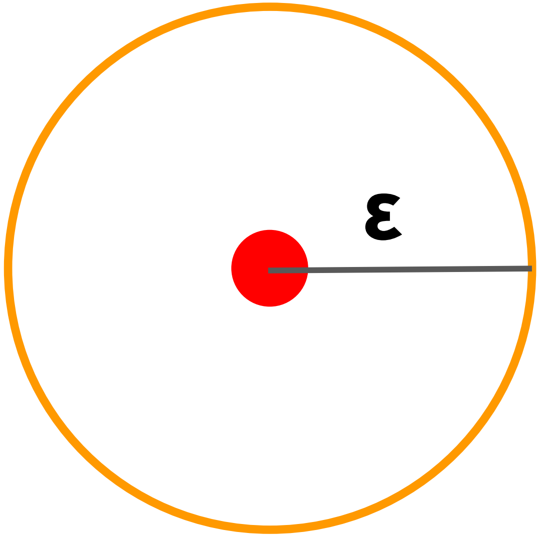
DBSCAN Explanation

1) **Identify all core points : points with at least MinPts neighbours**

In practice : MinPts = 4

1) **Group only core points into clusters**

1) **Add non-core points to the cluster if they are close to a core point**


ε

Choice of **ε**

1) For each point: compute **4-dist(p)** = distance to its 4-th nearest neighbor
2) Sort all points by **4-dist** → sorted k-distance graph

<u>Threshold = first valley in the graph</u>

→ Set `Eps = 4-dist(threshold)`

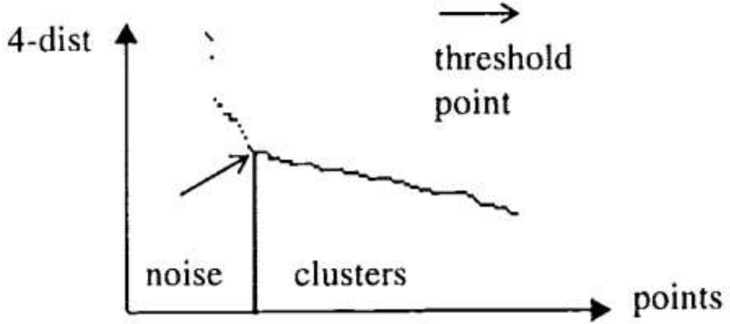
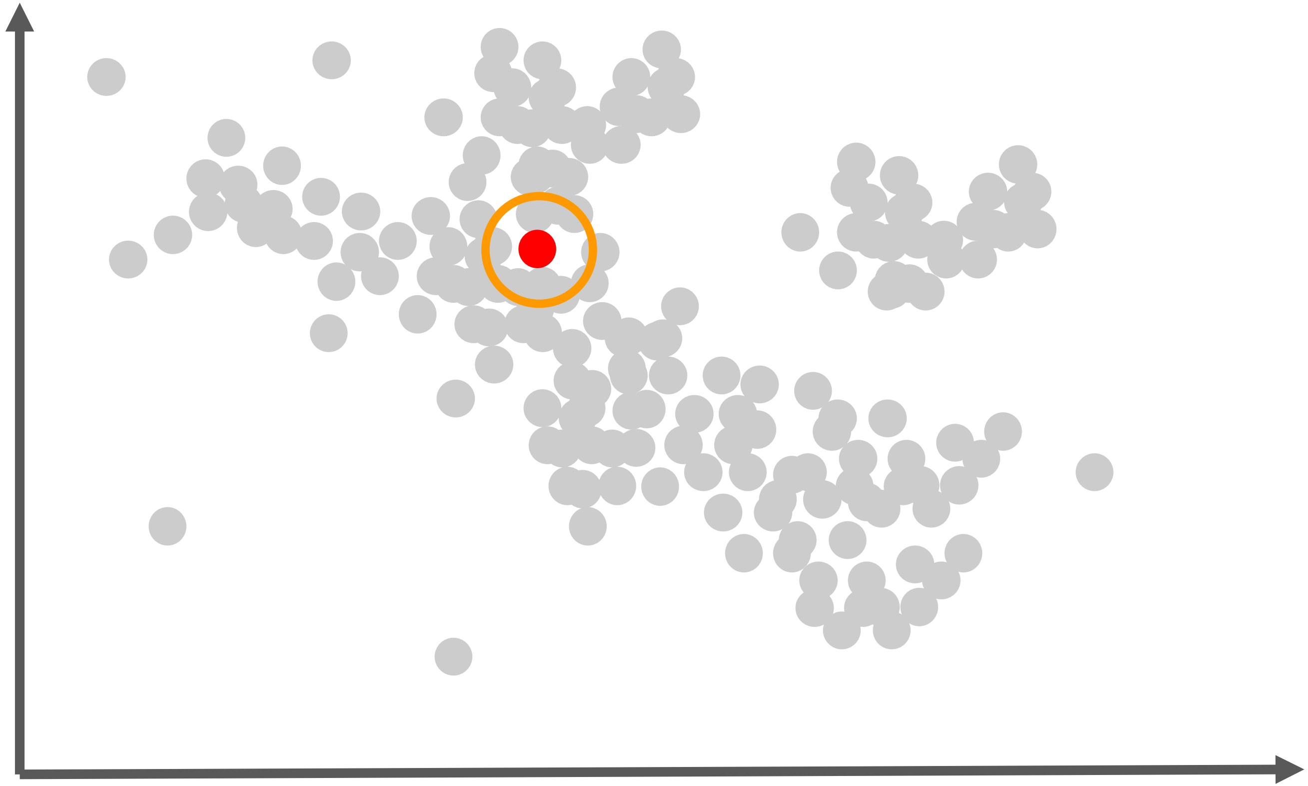


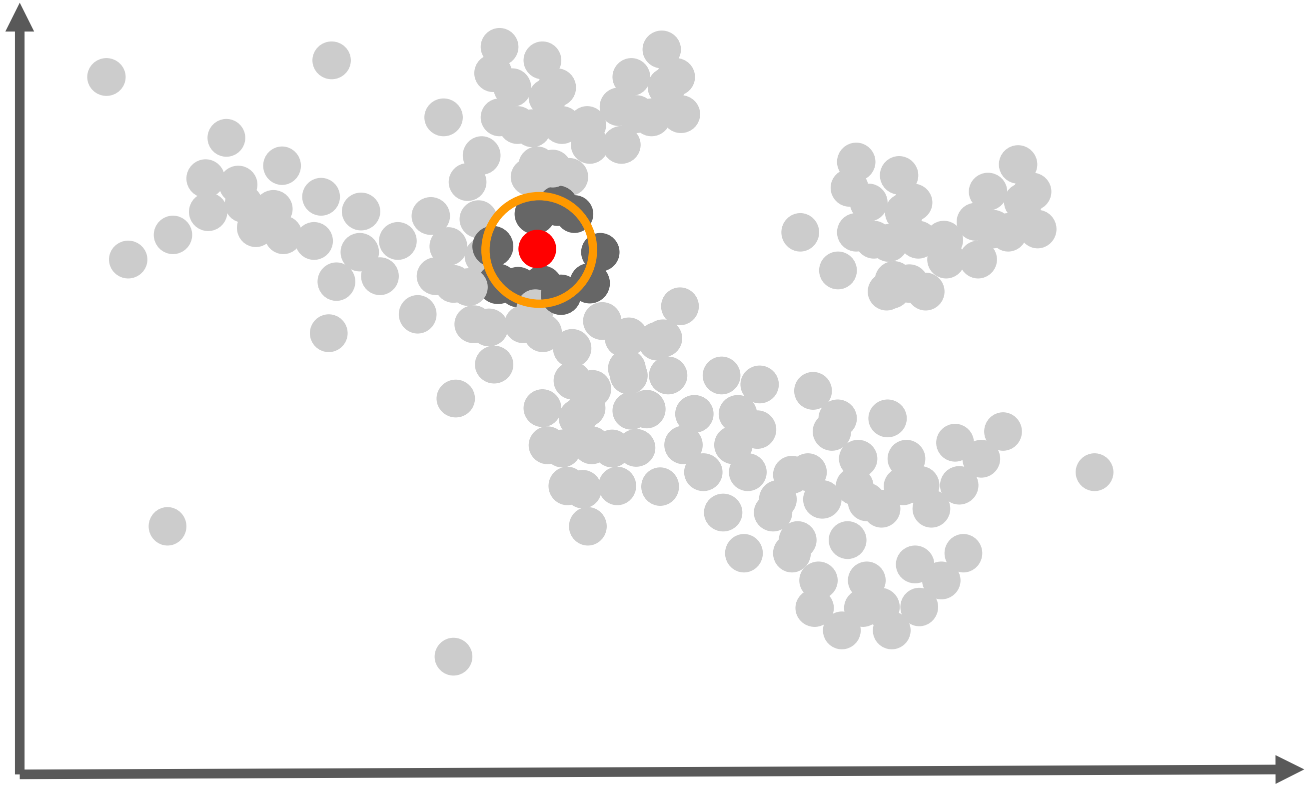
figure 4: sorted 4-dist graph for sample database 3

Points left of the threshold = noise

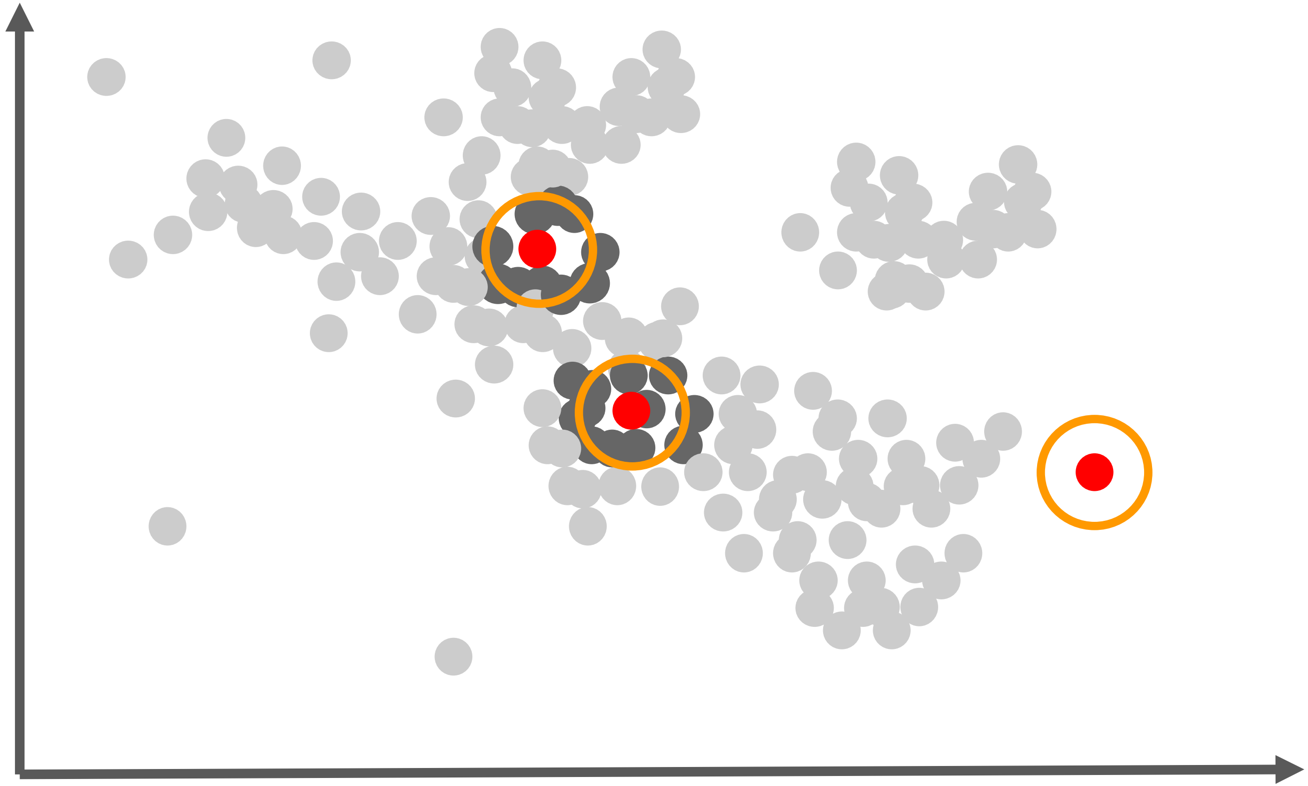
Points right = core points



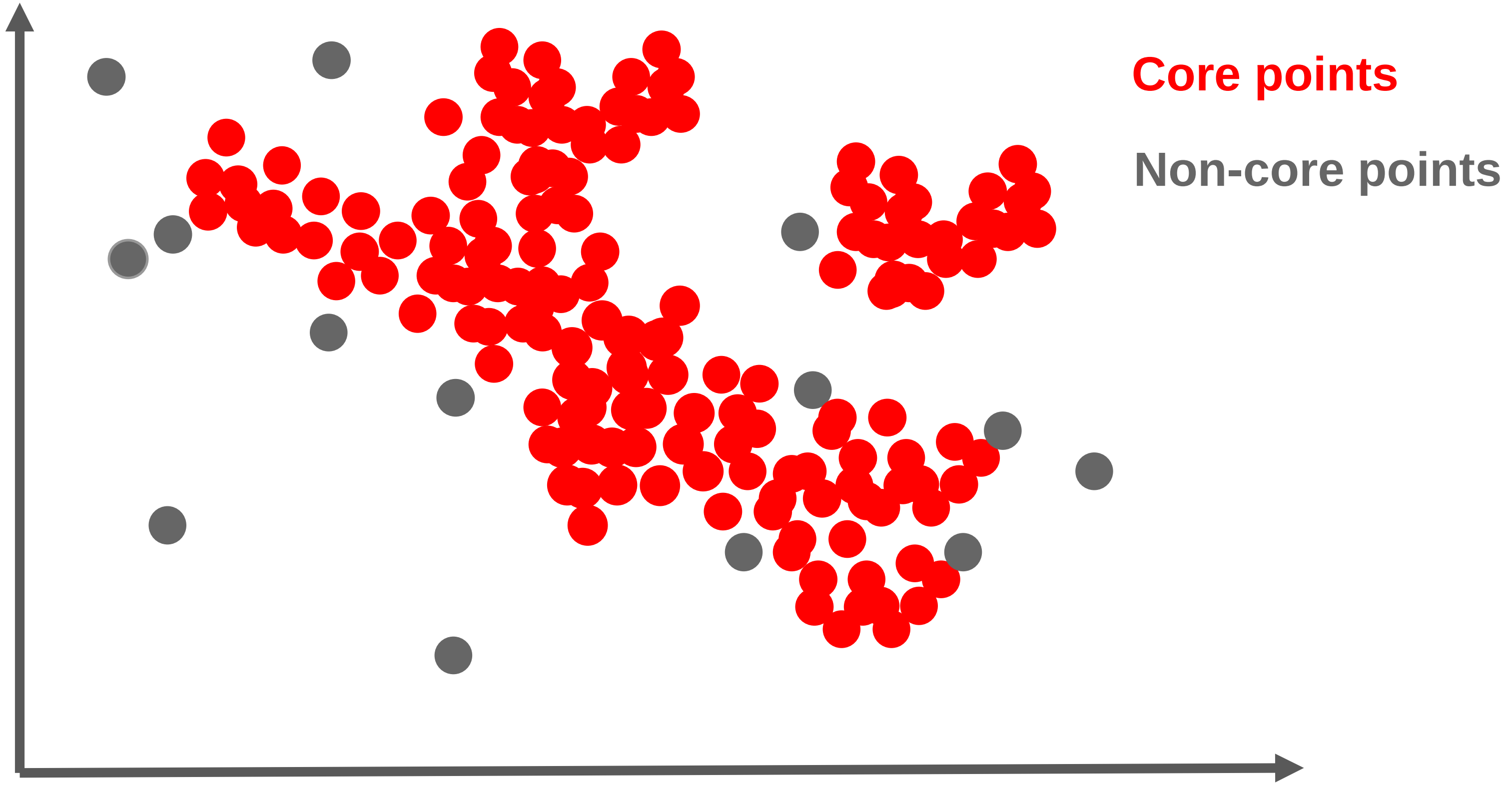
DBSCAN



DBSCAN



DBSCAN



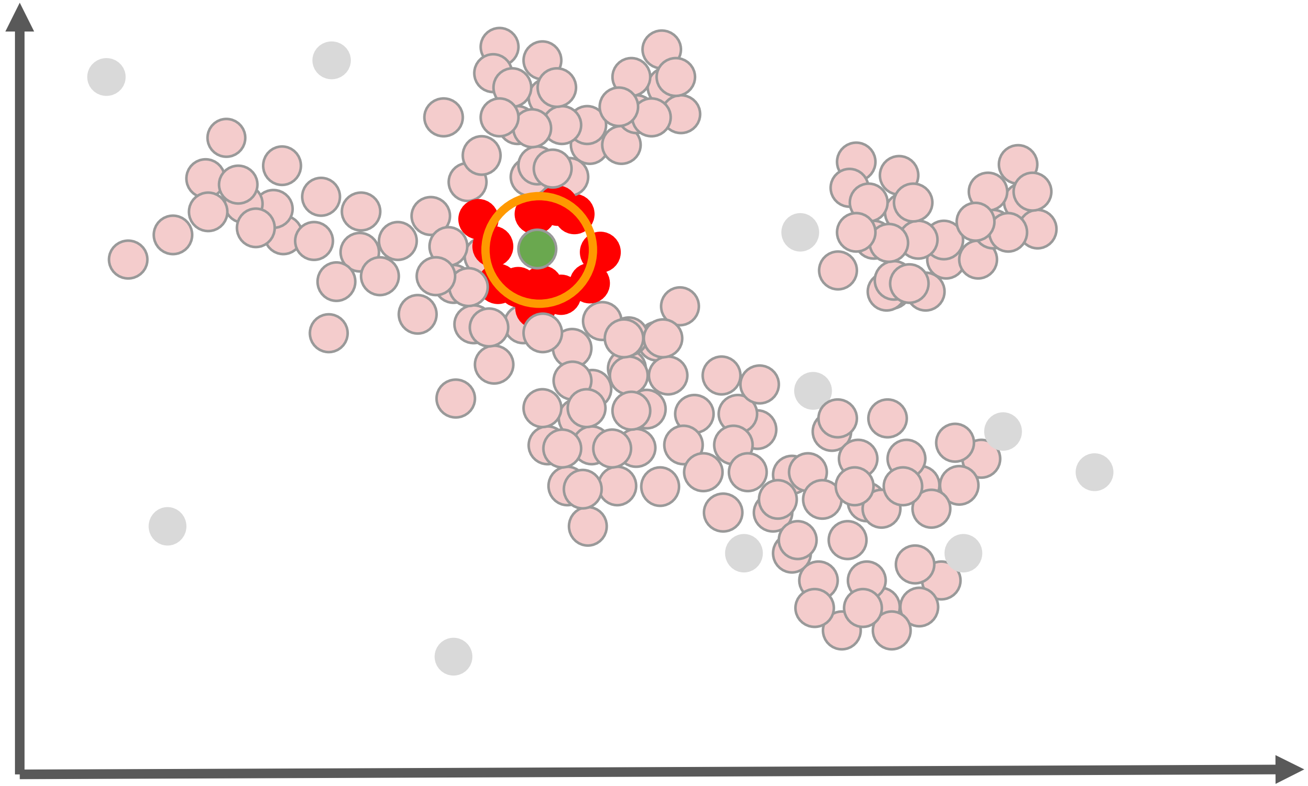
DBSCAN

DBSCAN Explanation

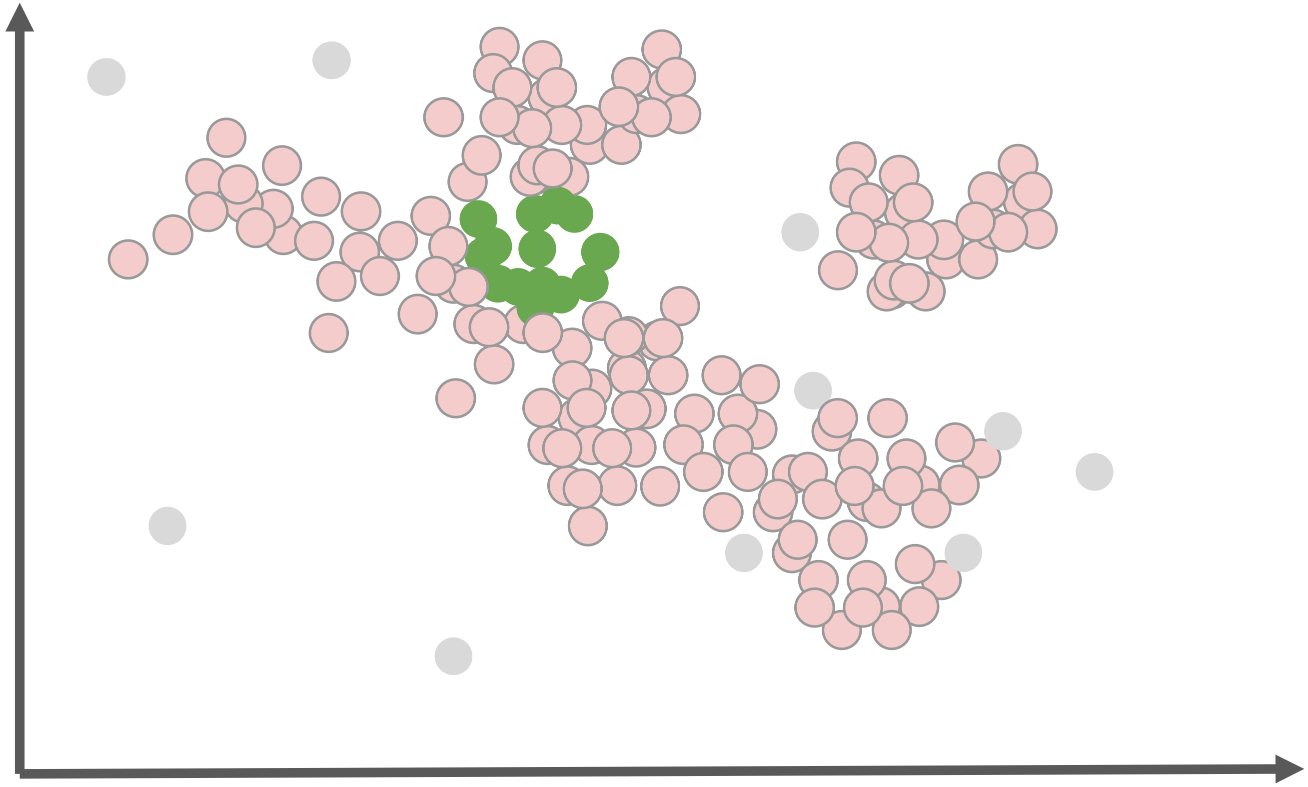
1) **Identify all core points**

1) **Group all core points into clusters**

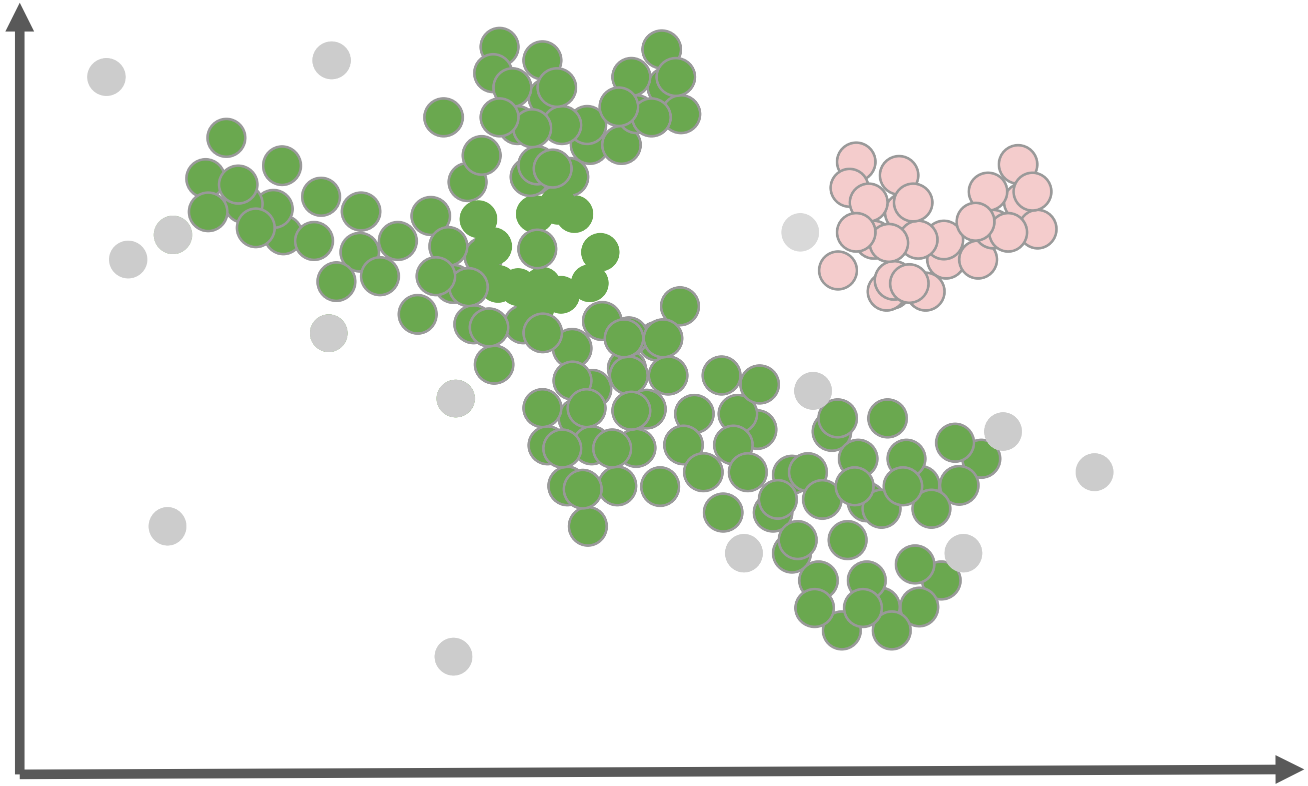
1) **Add non-core points to the cluster if they are close to a core point**



DBSCAN



DBSCAN



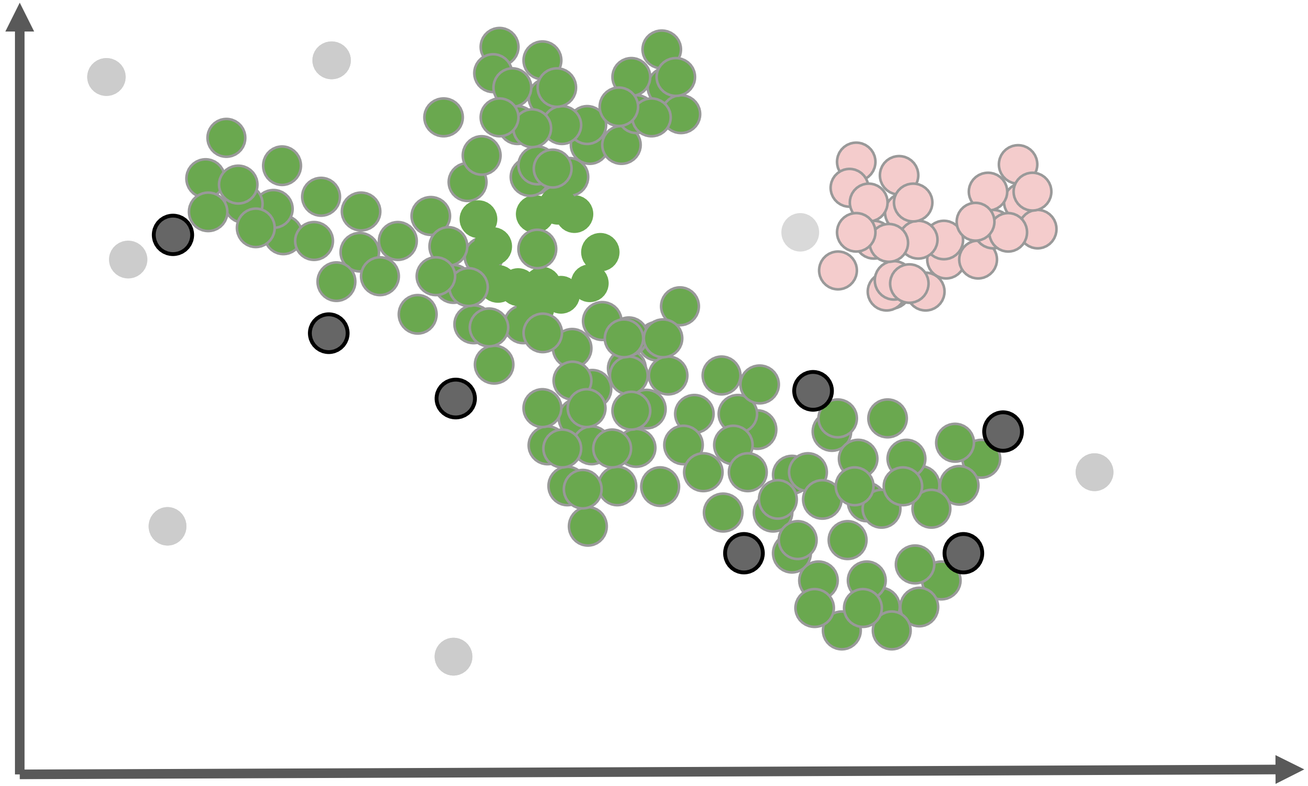
DBSCAN

DBSCAN Explanation

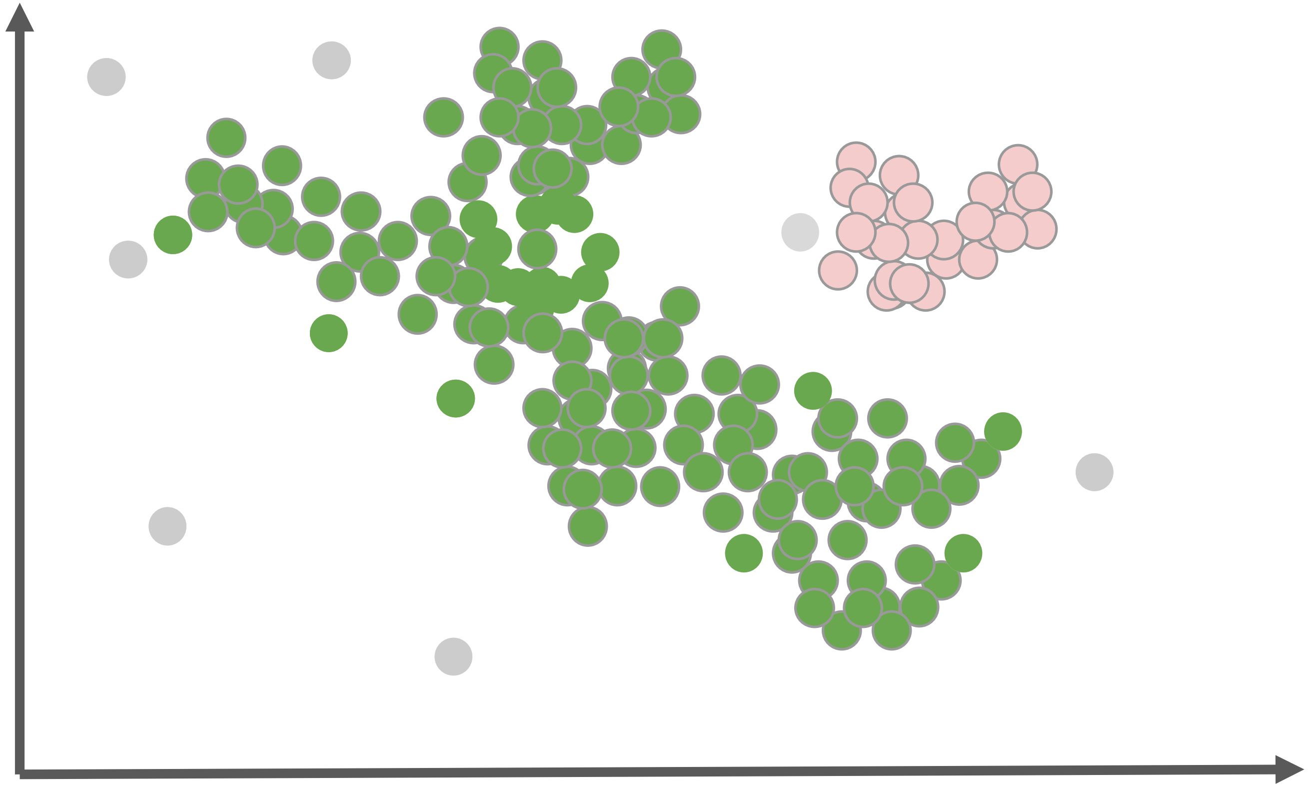
1) **Identify all core points**

1) **Group all core points into clusters**

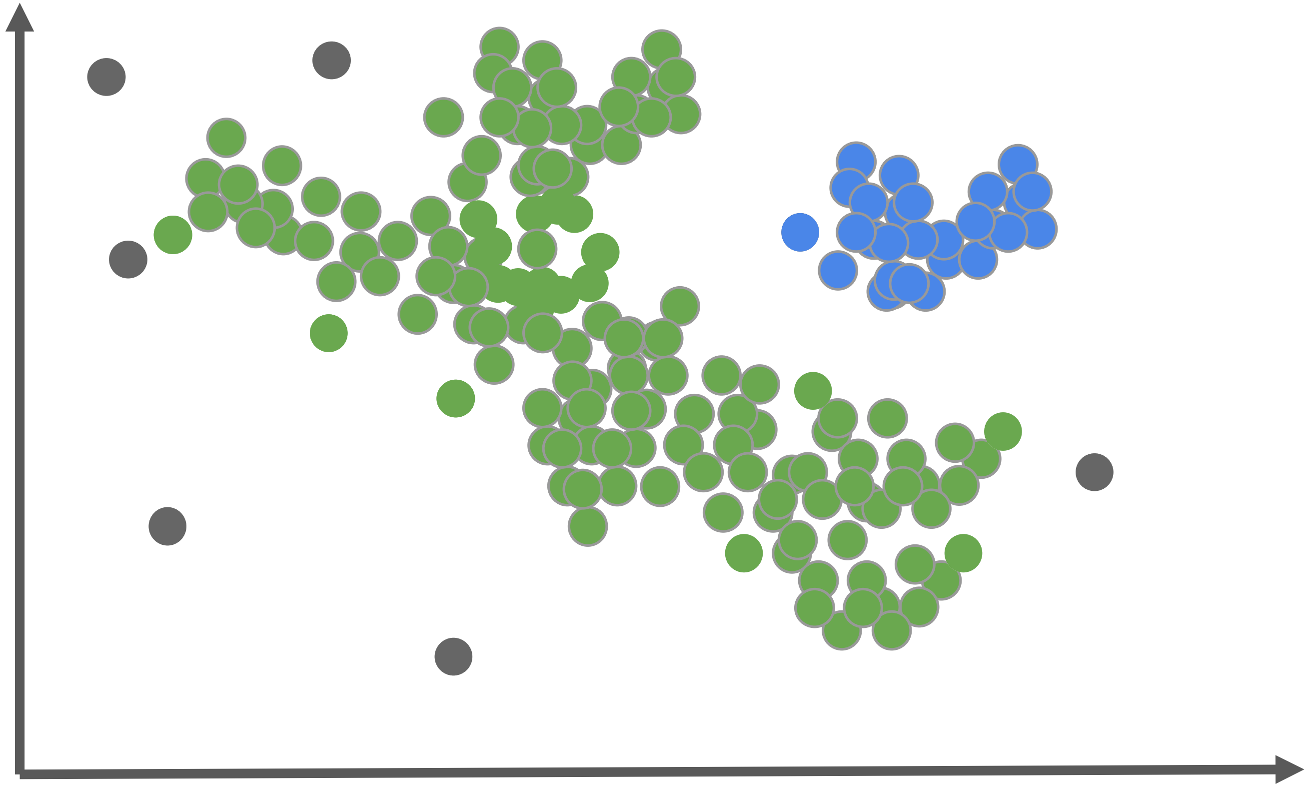
1) **Add non-core points to the cluster if they are close to a core point**



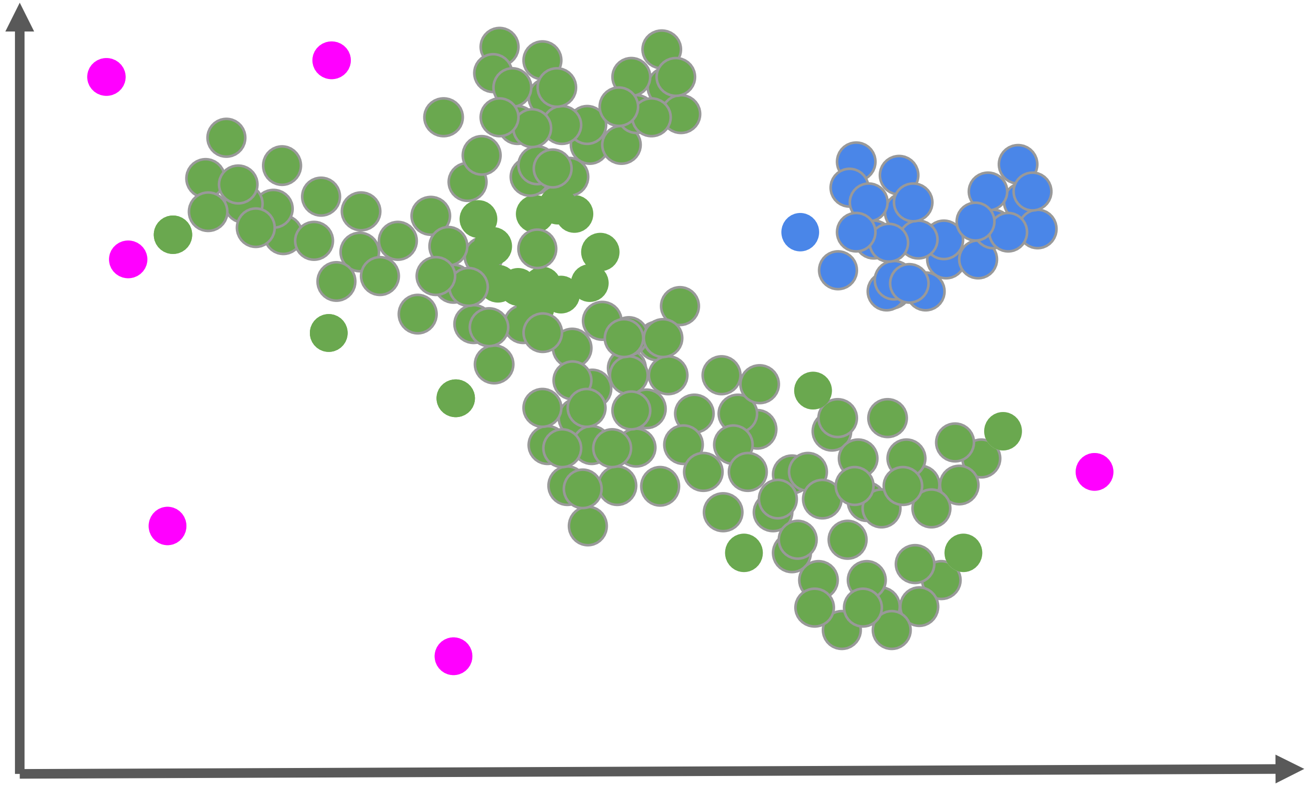
DBSCAN



DBSCAN



DBSCAN



DBSCAN

ADVANTAGES

- Detects clusters of arbitrary shape
- Identifies noise/outliers naturally
- No need to specify number of clusters **k**
- Robust to data order
- Efficient with spatial indexing structures

LIMITATIONS

- Sensitive to choice of parameters : **ε** and **`MinPts`**
- Struggles with cluster of varying densities
- Performance degrades in high dimensions
- Global parameter choice limits flexibility
- No guarantee in high-dimensional spaces

WHEN TO USE DBSCAN

- Spatial databases in 2D or 3D
- Datasets with noise or outliers
- Unknown number of clusters or hard to estimate
- Large-scale datasates (>1000)

LESS APPROPRIATE FOR DBSCAN

- Datasets with clusters of varying density
- Non-point spatial data
- High-dimensional data

Synthetic Databases in the Document

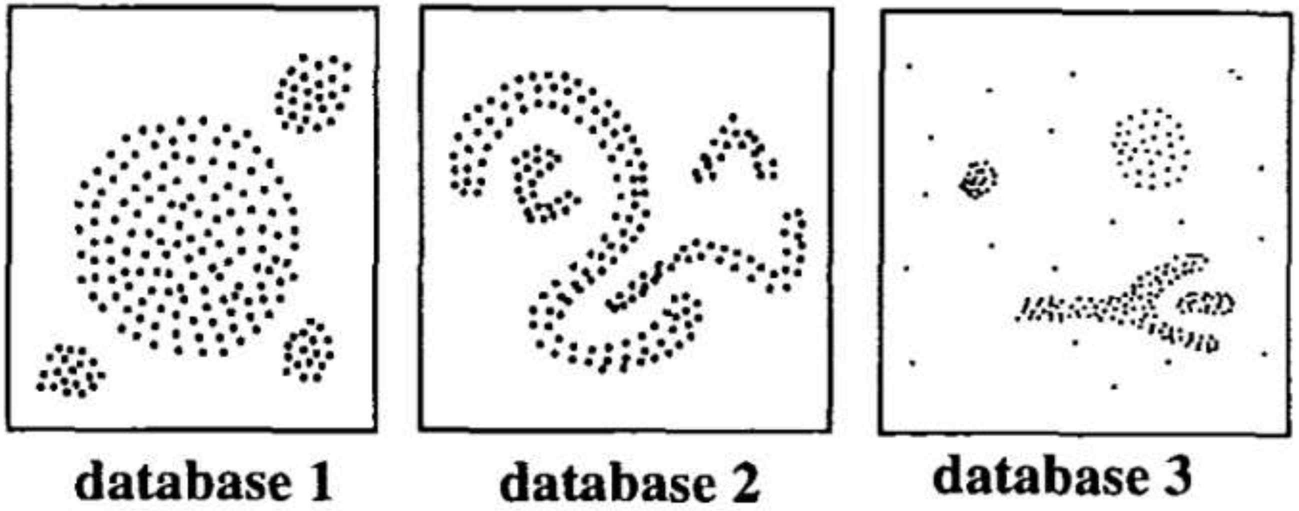


figure 1: Sample databases

Balls of different sizes / Nonconvex shapes / Different shapes of different sizes with noise

Results for DBSCAN in the Document

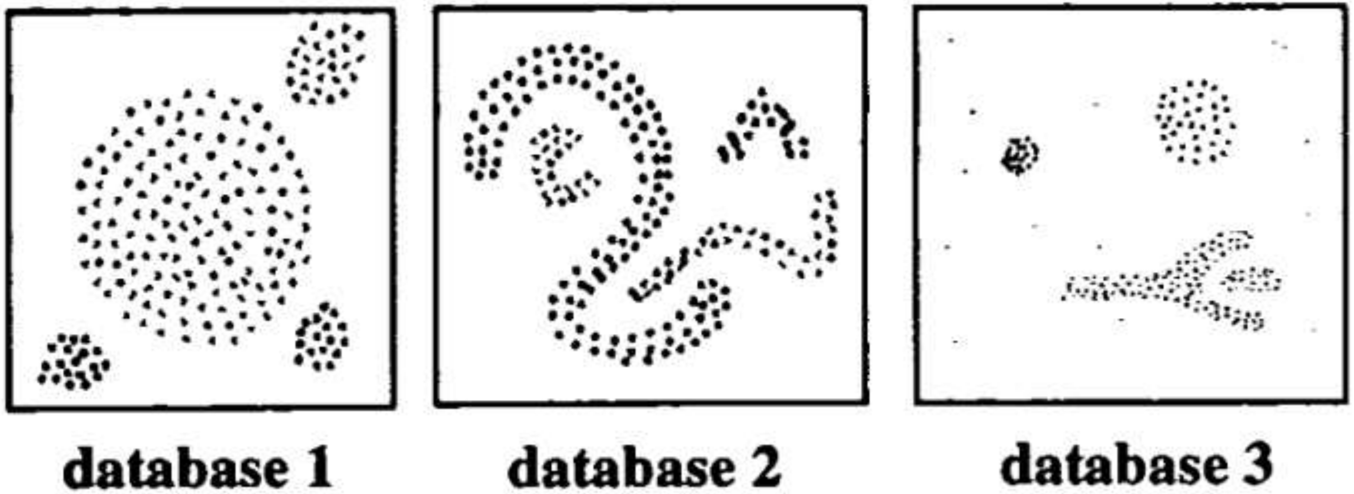
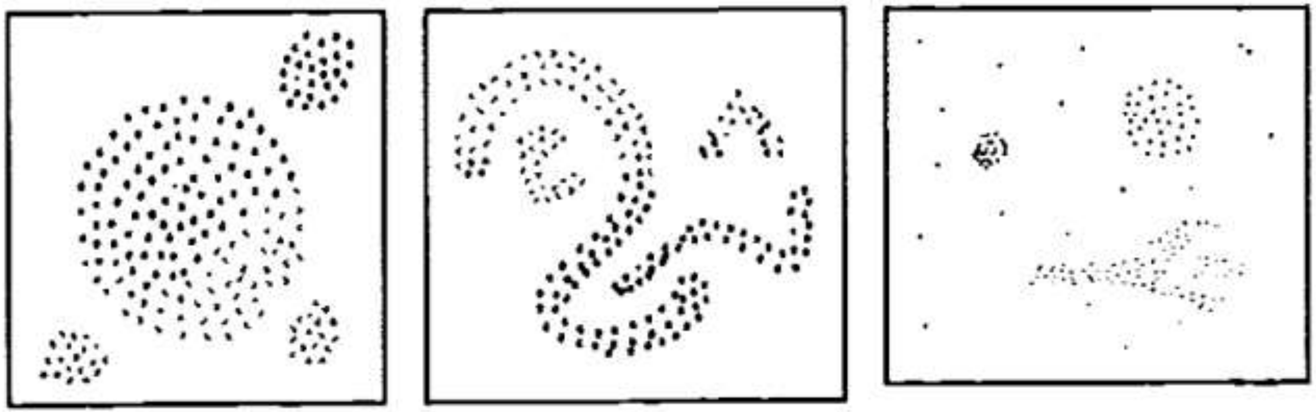


figure 6: Clusterings discovered by DBSCAN

Hard to see but perfect assignment to each cluster

Results for CLARANS in the Document



database 1 database 2 database 3

figure 5: Clusterings discovered by CLARANS

We see a difference of blacks within clusters : they have been split

Run Time in Seconds in the Document

Table 1: run time in seconds

number of points	1252	2503	3910	5213	6256
DBSCAN	3.1	6.7	11.3	16.0	17.8
CLAR-ANS	758	3026	6845	11745	18029
number of points	**7820**	**8937**	**10426**	**12512**	
DBSCAN	24.5	28.2	32.7	41.7	
CLAR-ANS	29826	39265	60540	80638	

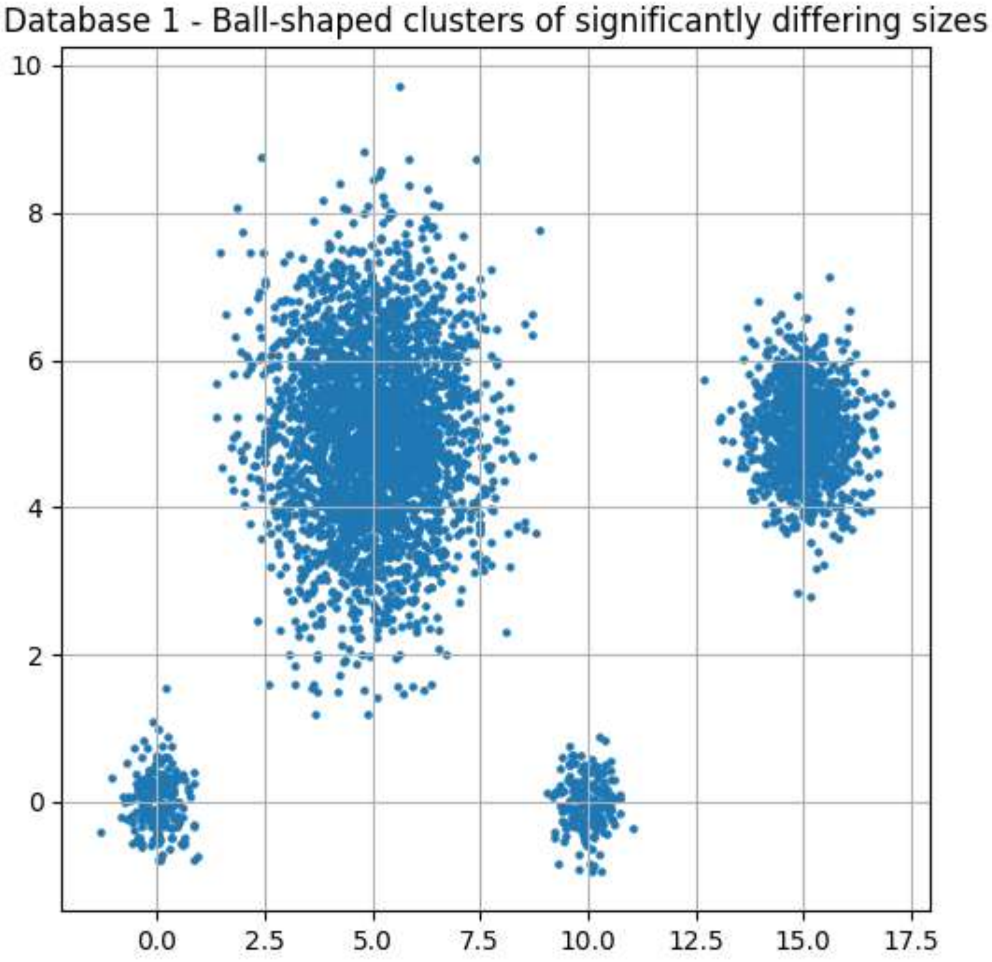
DBSCAN is at least 250 times faster than CLARANS, up to 1900 faster

Summary

Presentation of our experiment

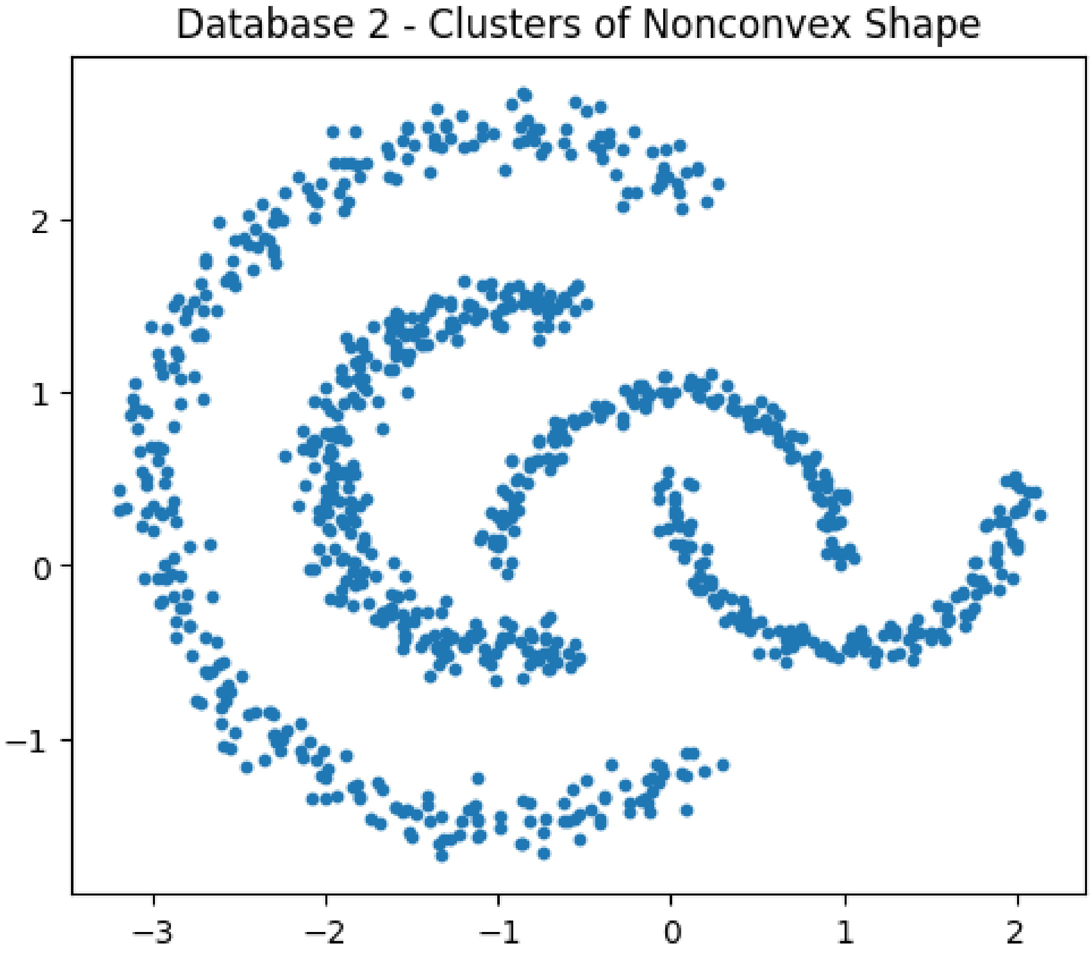
Generation of synthetic databases

- sklearn.datasets.make_blobs()

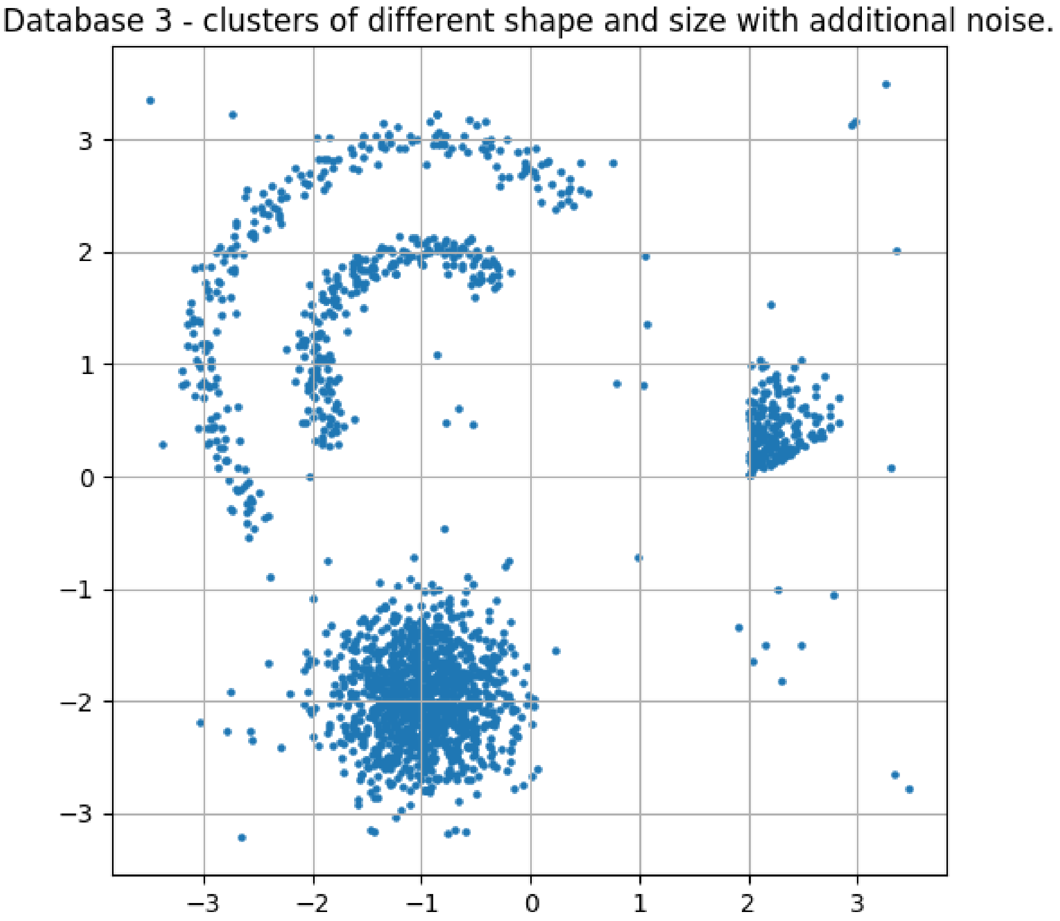


- sklearn.dataset.make_moons()
- sklearn.dataset.make_circles()

  and filtered with f(x) = x**2

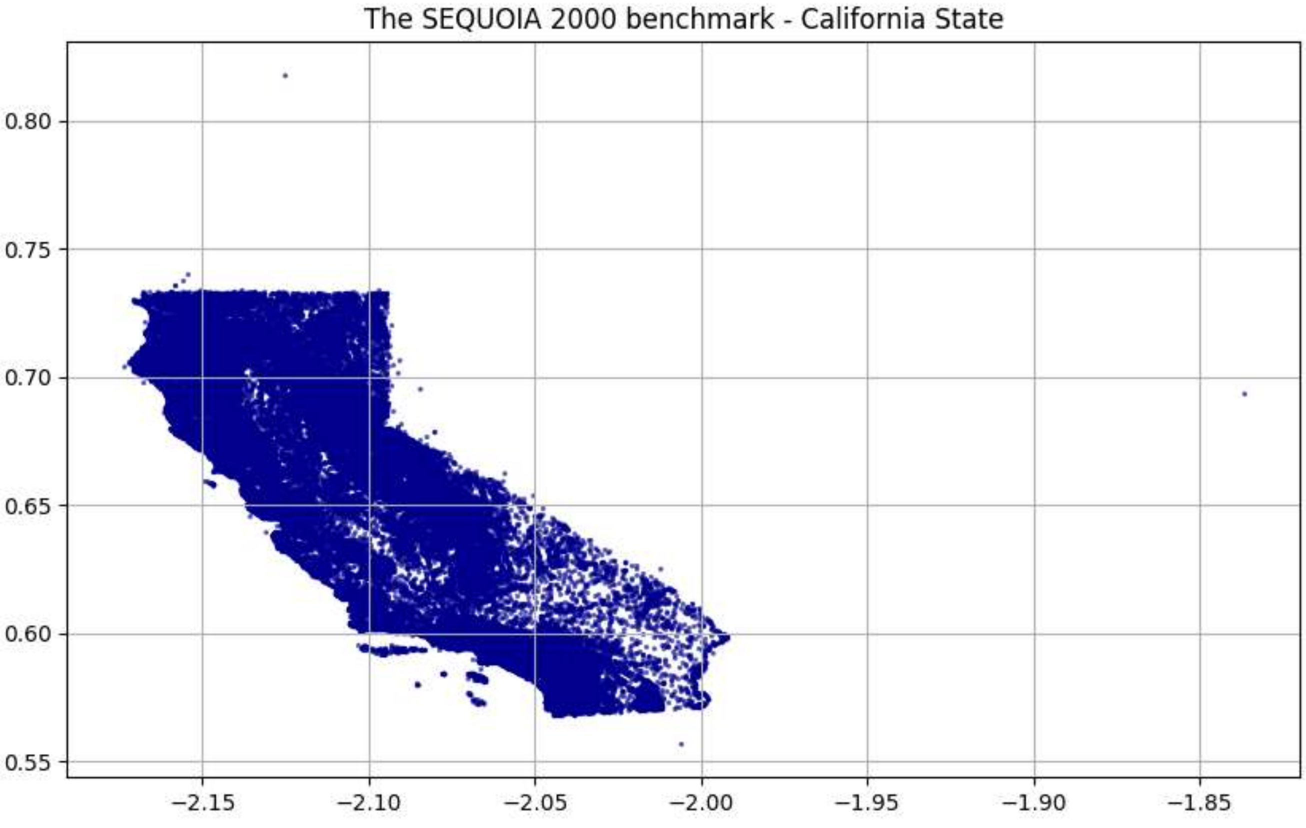


- sklearn.dataset.make_circles() and filtered with f(x) = x
- sklearn.datasets.make_blobs()
- sklearn.datasets.make_blobs() and filtered with f(x) = 0 and f(x) = 2*x : triangle
- np.random.uniform() : noise

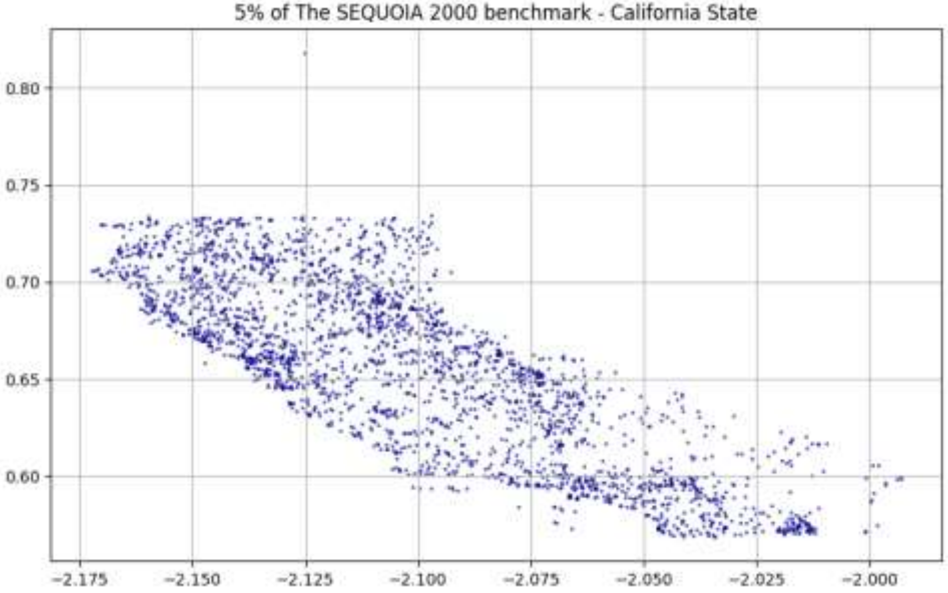


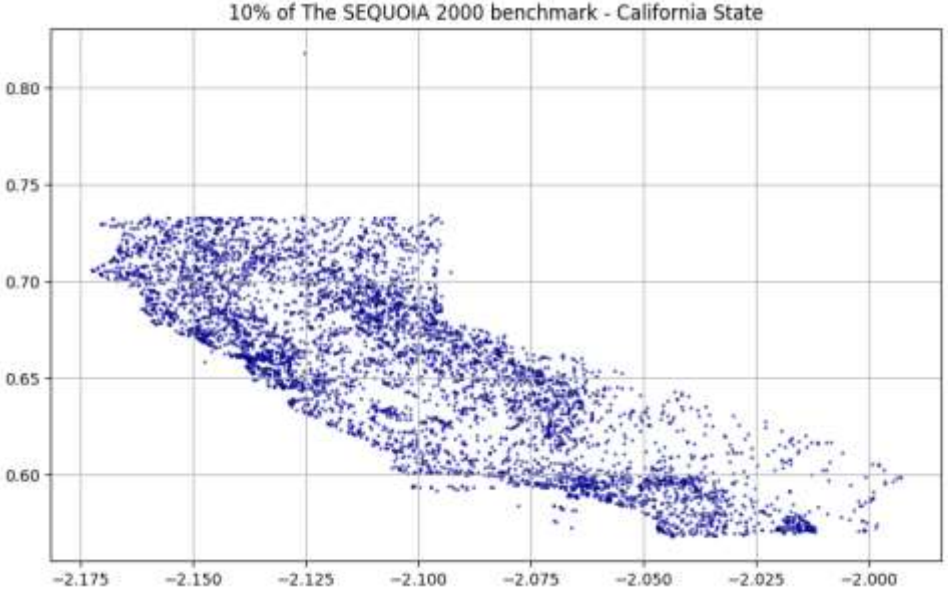
The Spatial Dataset of the Paper

- GPS points from longitude / latitude (degrees) to radians

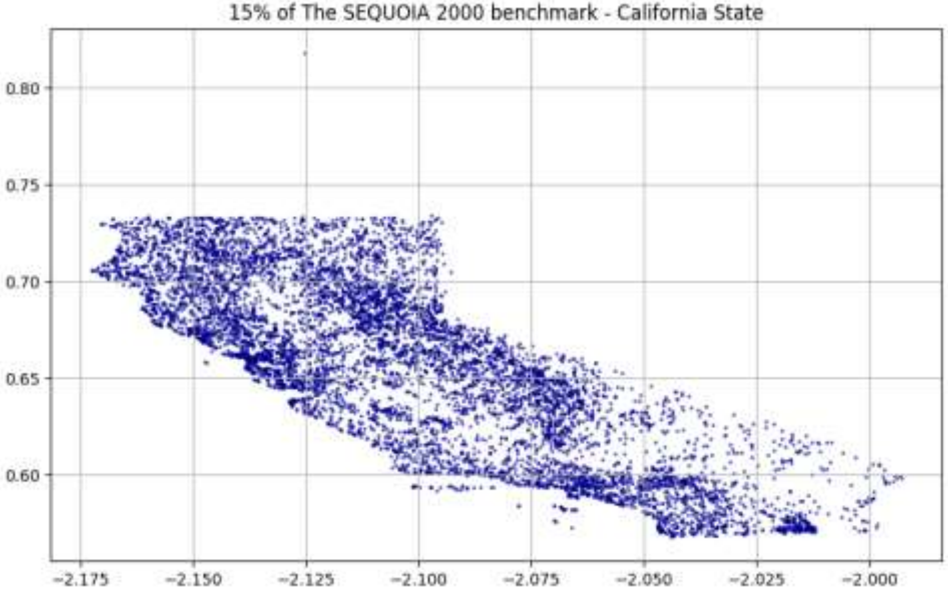


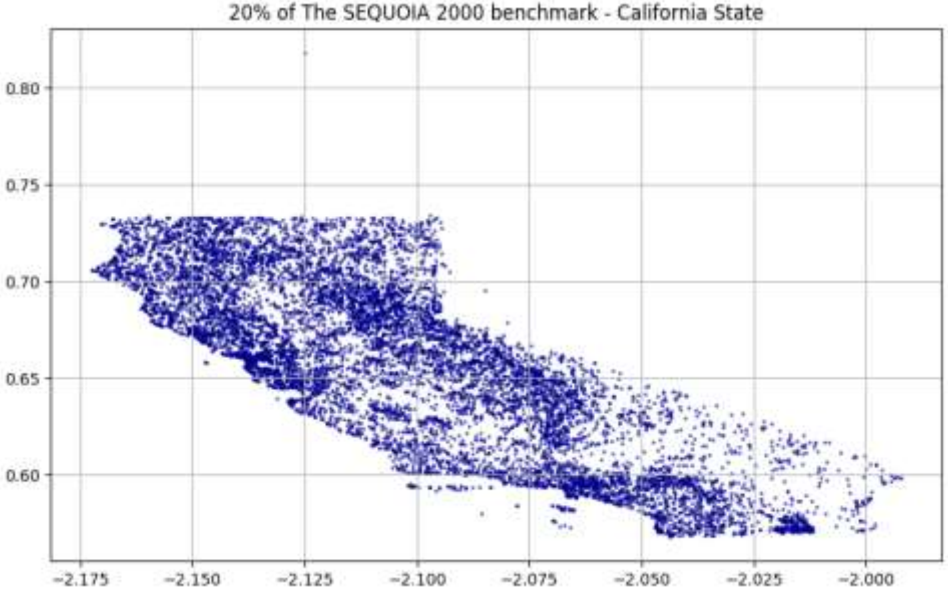
Subsets from 5% to 20% of the dataset





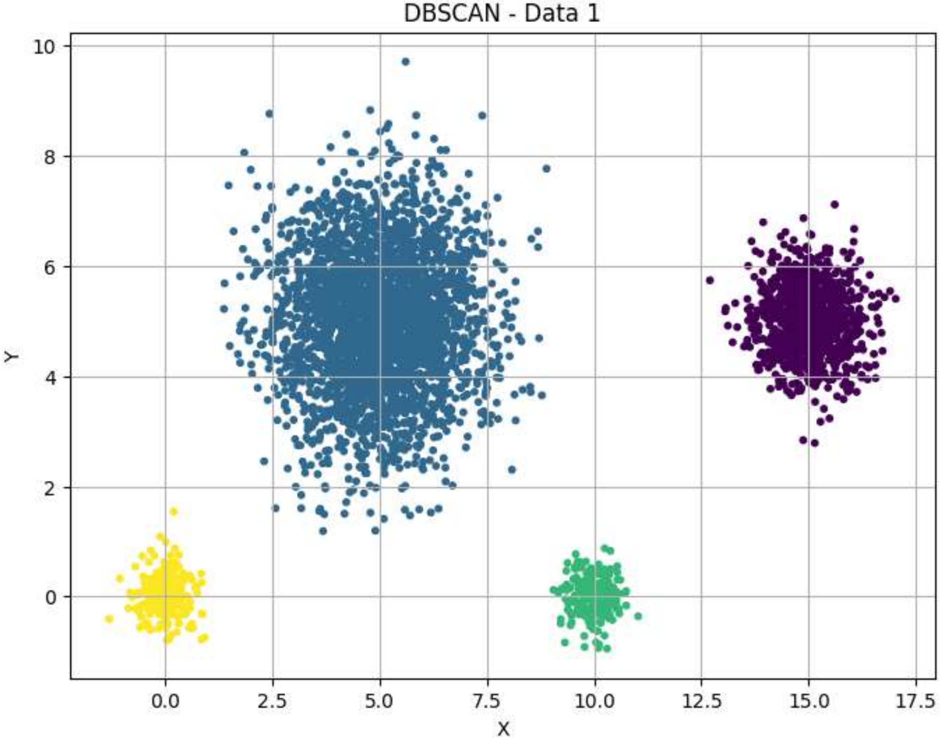
Subsets from 5% to 20% of the dataset

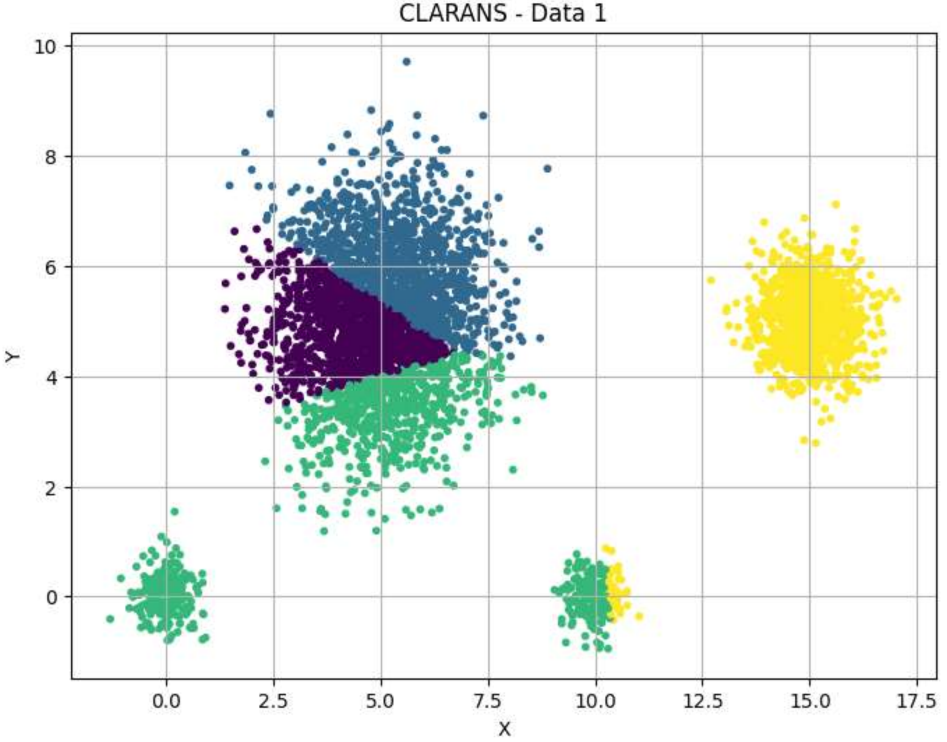




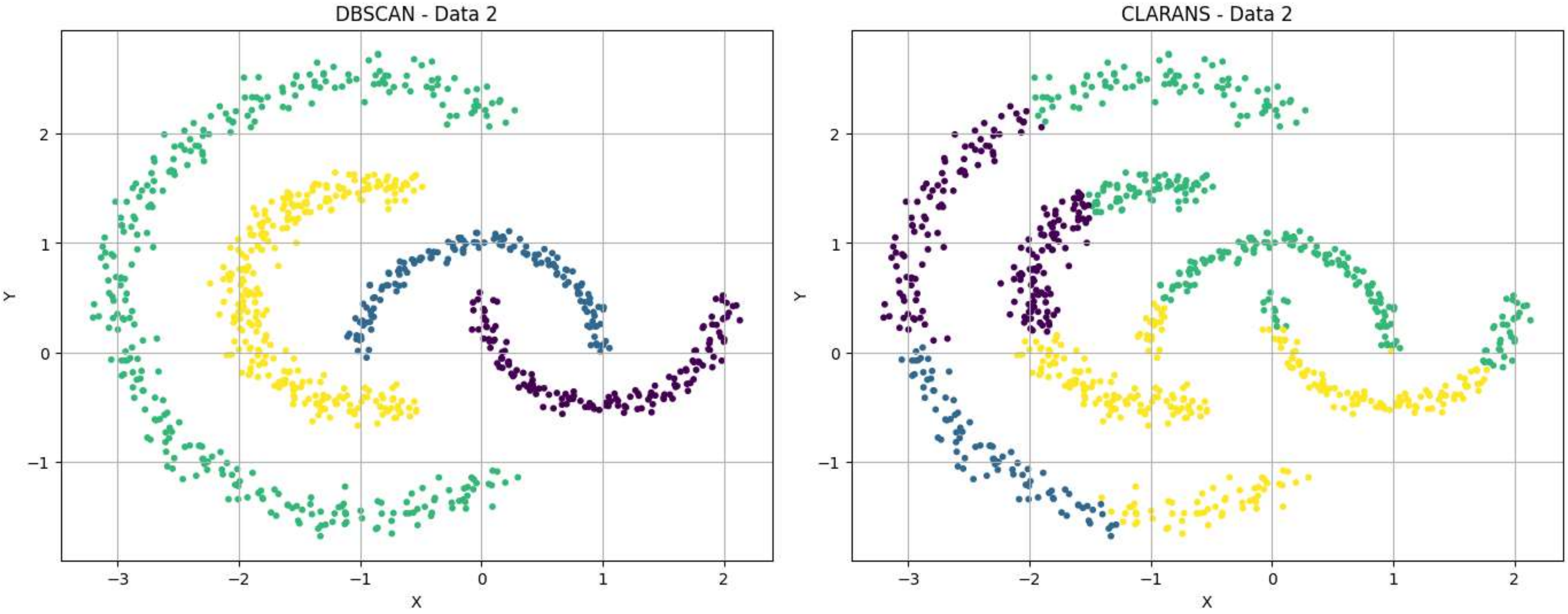
DBSCAN vs CLARANS Results on Synthetic Data

Comparison of the Results : eps=1.2, min=5

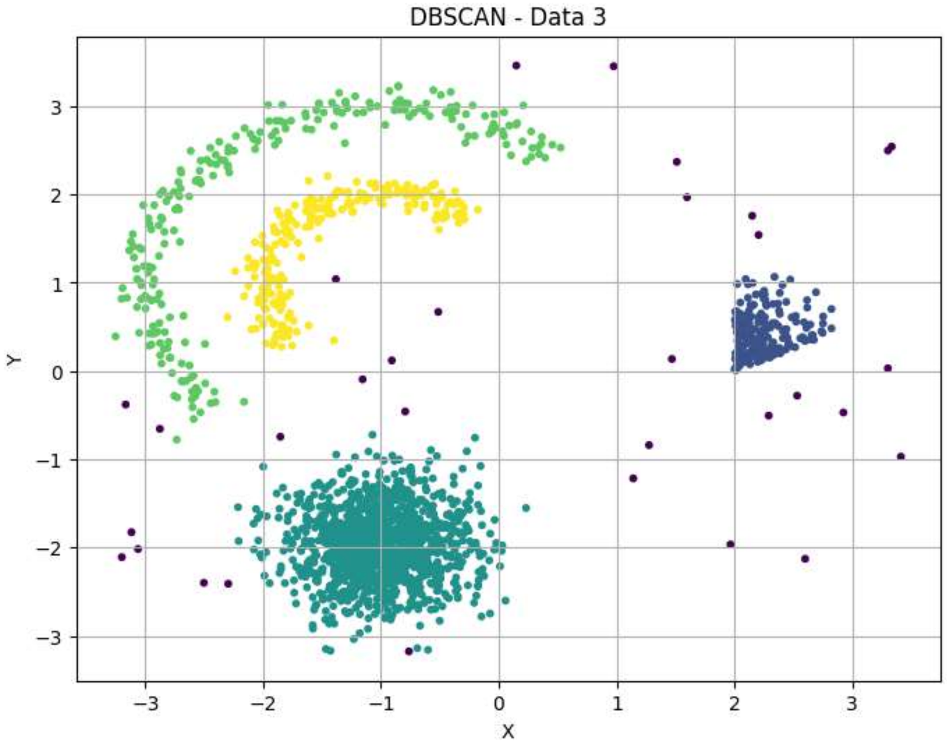


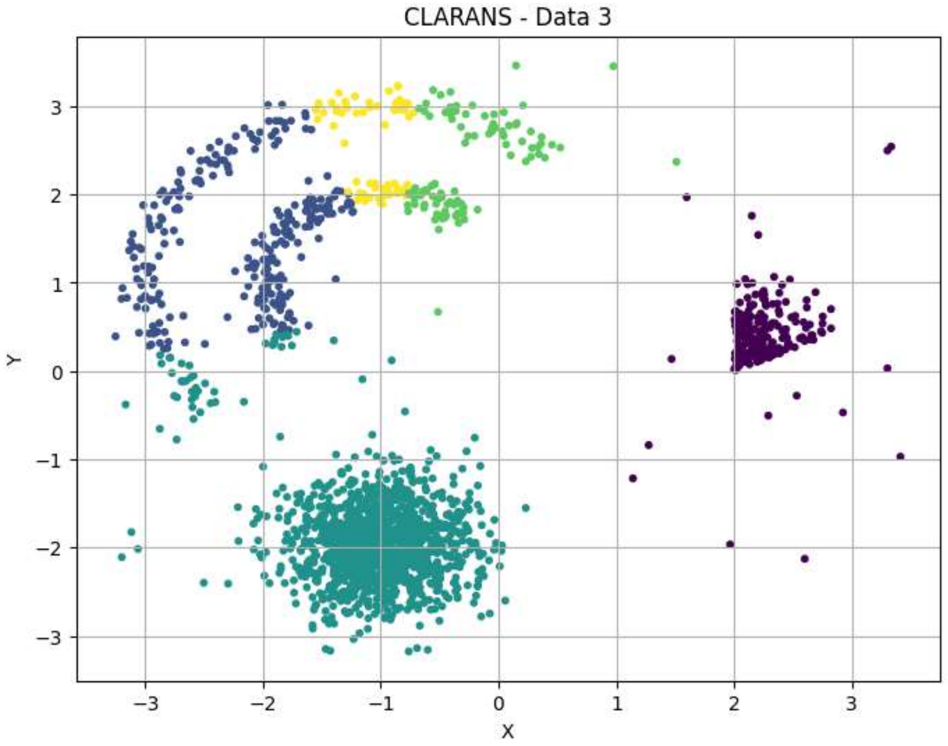


Comparison of the Results : eps=0.25, min=5



Comparison of the Results : eps=0.3, min=5



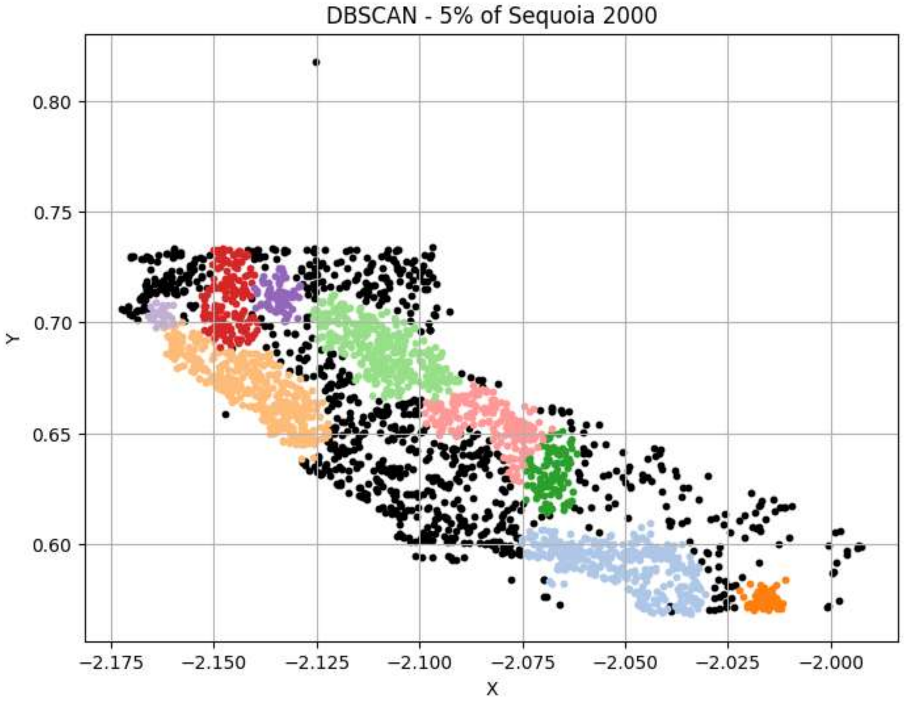


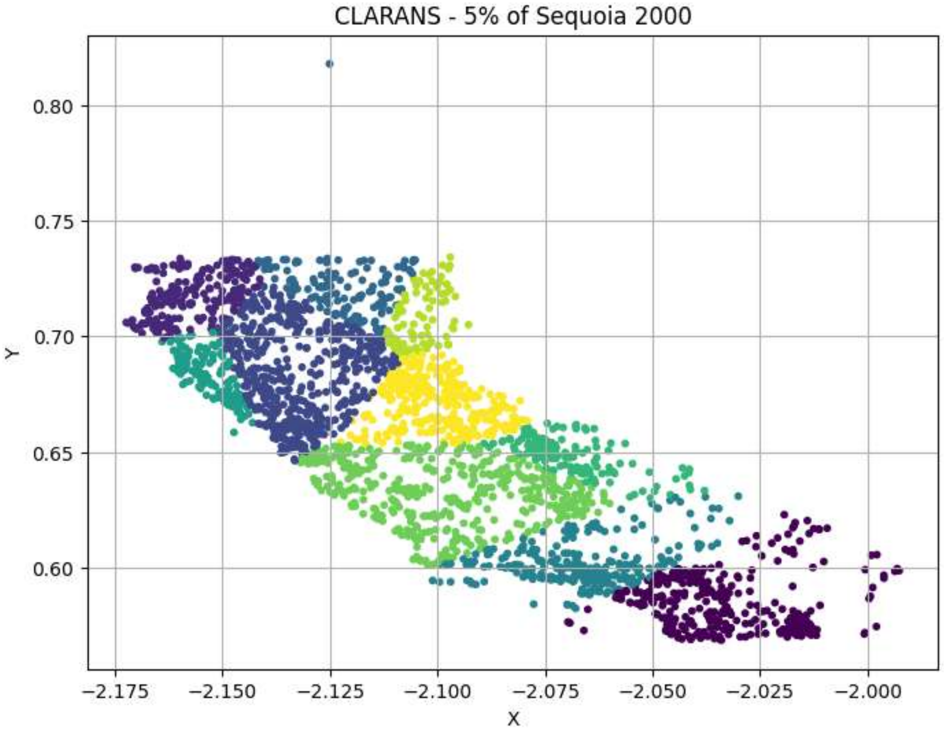
DBSCAN vs CLARANS Results on Real Spatial Data

- The points being more and more dense we increase the minimum of samples to define a core points so we get readable results -> only really dense clusters are localized.

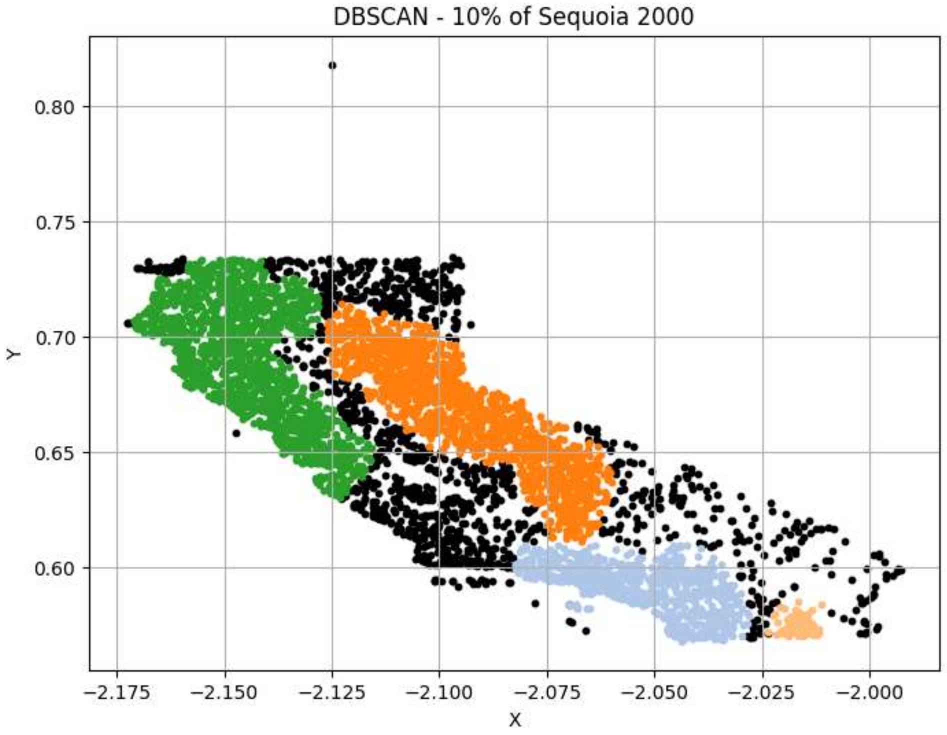
- We always take eps = 25km / 6371km and use Haversine metric to take in account the spherical shape of the Earth since California is a very large state (3rd of the USA, 1300km by 400km). Otherwise diminishing eps would constrain too much the size of clusters.

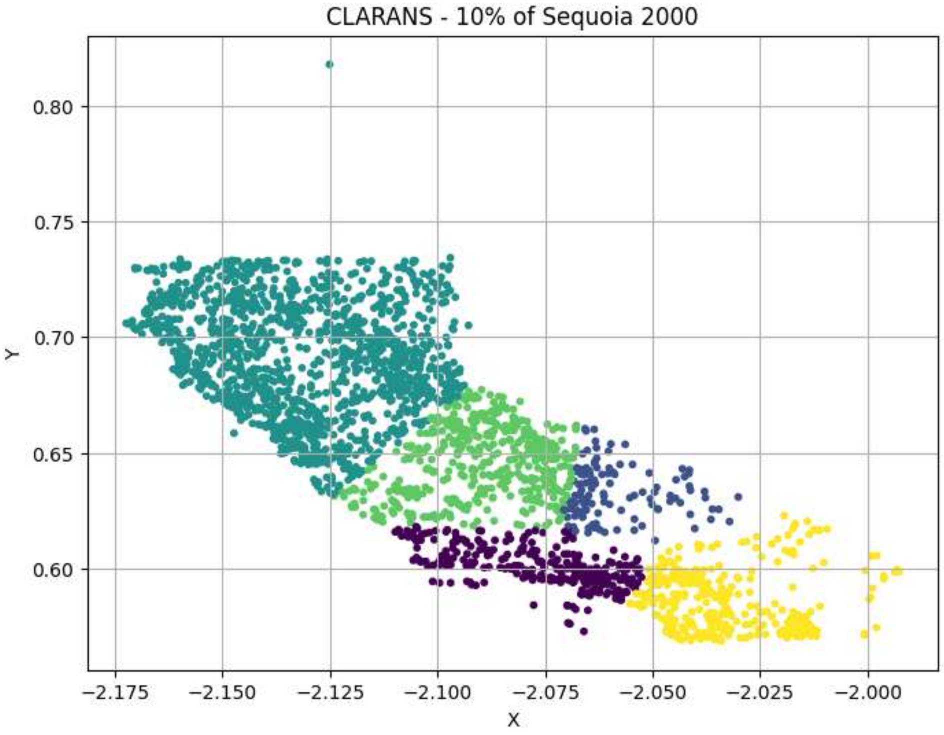
Comparison of the Results : min=30, 9 clusters



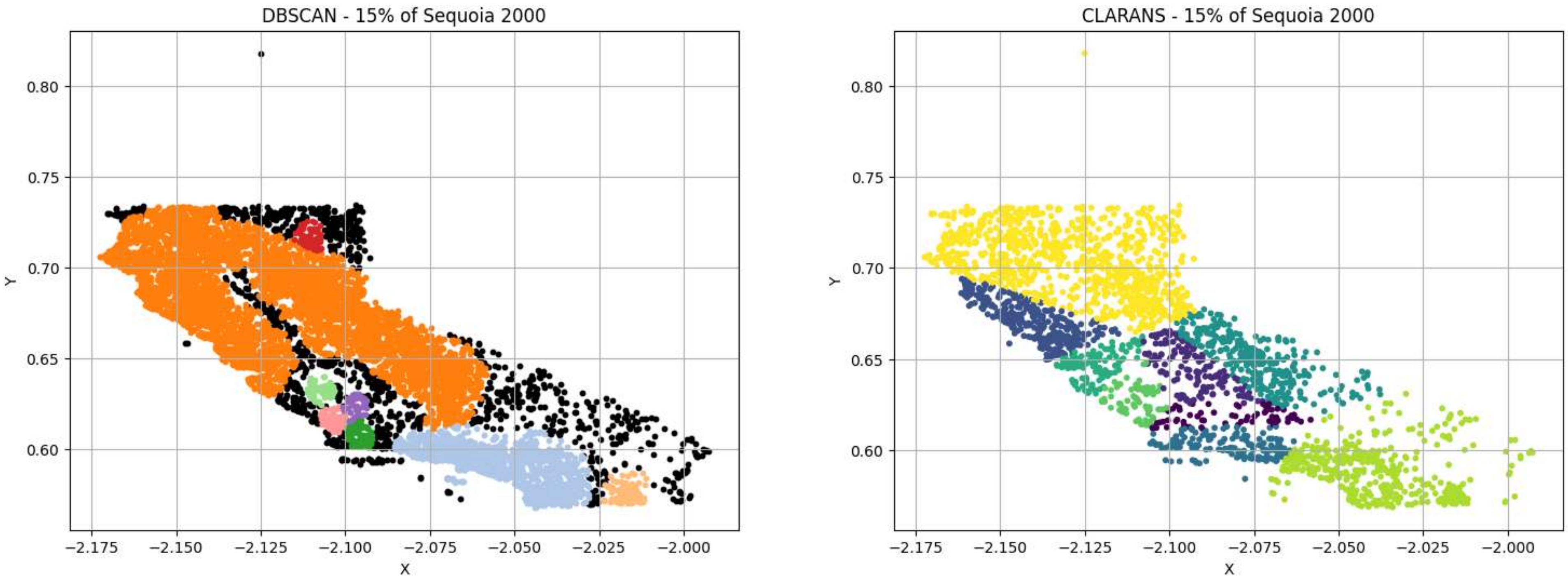


Comparison of the Results : min=50, 4 clusters

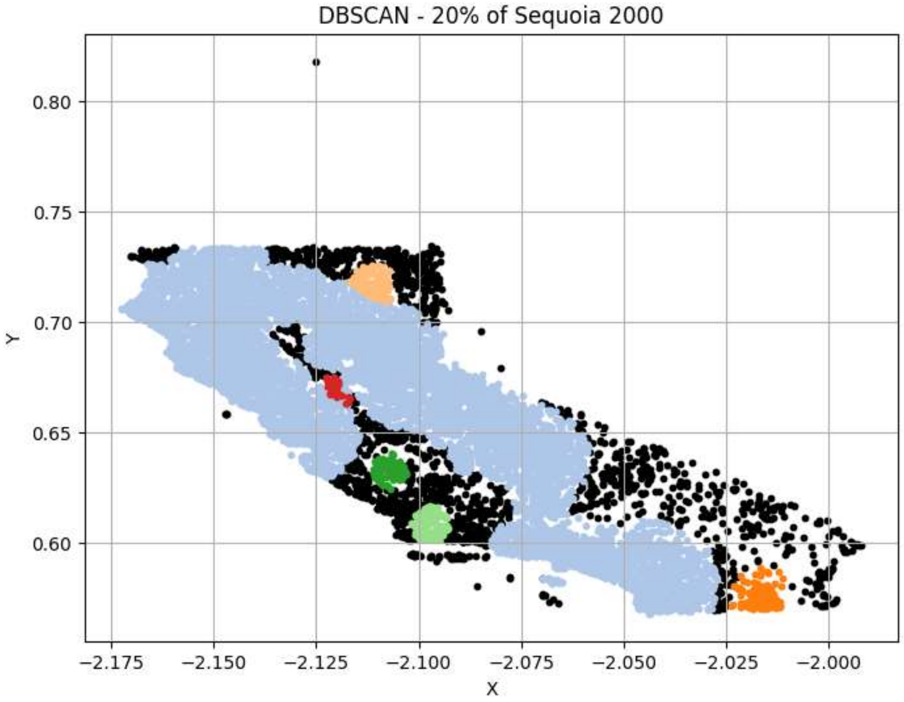


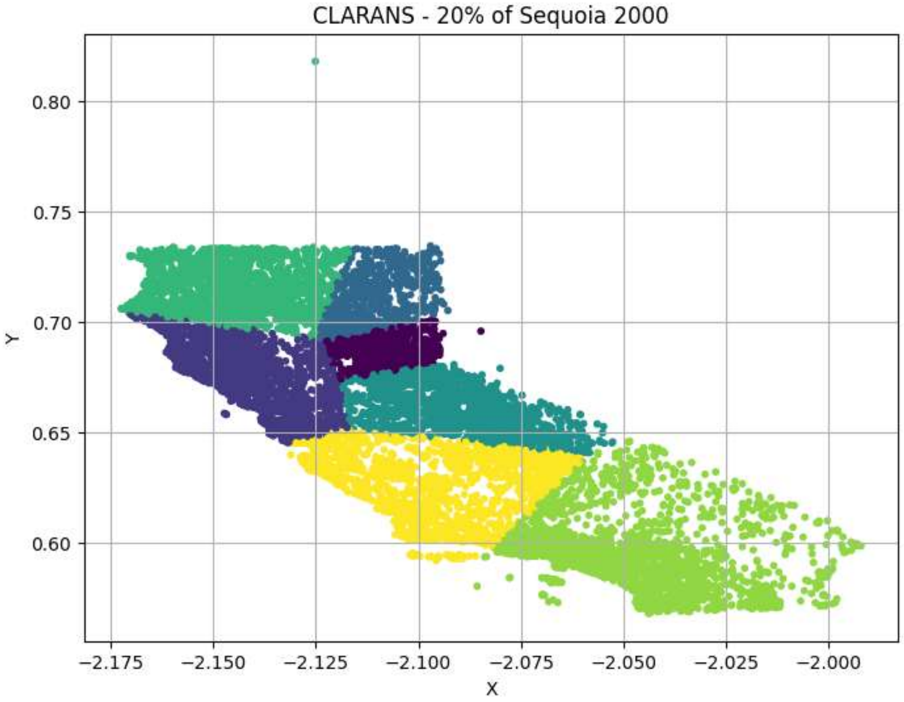


Comparison of the Results : min=65, 8 clusters



Comparison of the Results : min=85, 6 clusters





DBSCAN vs CLARANS Results on Real Spatial Data

- DBSCAN clusters are more graphically more logical. We can see which places have not been registered enough in the GPS database and wich are well present on the map.

- CLARANS looks like a polygonal cutting of the map, with no real logic behind. This doesn’t represent the geographical issues to create the GPS points (less inhabitants, more urban zone, hills, lakes, …)

Comparison of the Run Time (in seconds)

	Data1 : 4400	Data2 : 1200	Data3 : 3150	Data4_5 : 2634	Data4_10 : 5268	Data4_15 : 7901	Data_20 : 10535
DBSCAN	0.0725	0.00495	0.0139	0.0380	0.0934	0.154	0.242
CLARANS	20.4	13.2	17.0	1.986	36.2	2.563	20.45
CLARANS / DBSCAN	281.4	2667	1223	52.26	387.6	16.64	84.5

Conclusion

DBSCAN is also way more efficient in time than CLARANS with a factor between 10 and 2600 (at least 100 says the paper), and by clustering well complicated shapes and large datasets.

Our results confirm in the same measure the performance of DBSCAN on CLARANS described in the paper.