CLUEstering: a high-performance density-based clustering library for scientific computing

Presented by: Simone Balducci

Supervisor: Daniele Bonacorsi
Co-supervisors: Francesco Giacomini, INFN-CNAF
Felice Pantaleo, CERN
Wahid Redjeb, CERN

Alma Mater Studiorum – Università di Bologna

20 September 2024

Goals of the thesis

The goals of this thesis are to:

- Develop a general-purpose clustering library based on the CLUE algorithm

Goals of the thesis

The goals of this thesis are to:

- Develop a general-purpose clustering library based on the CLUE algorithm
- Achieve performance portability using the Alpaka library

Goals of the thesis

The goals of this thesis are to:

- Develop a general-purpose clustering library based on the CLUE algorithm
- Achieve performance portability using the Alpaka library
- Benchmark the library and assess the quality of the results

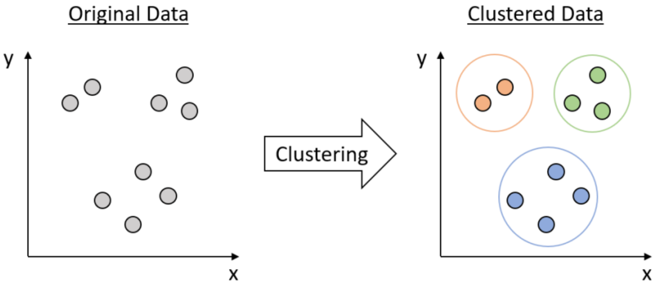
Goals of the thesis

The goals of this thesis are to:

- Develop a general-purpose clustering library based on the CLUE algorithm
- Achieve performance portability using the Alpaka library
- Benchmark the library and assess the quality of the results
- Apply the library to several problems regarding different branches of science, in order to show its generality

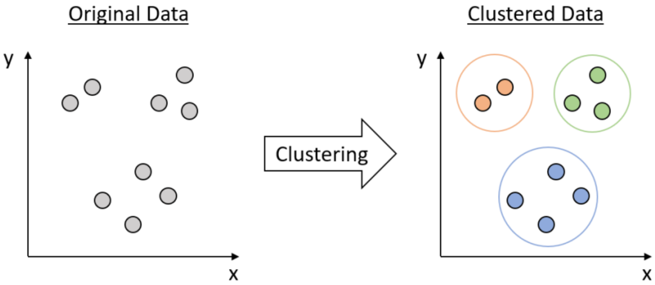
Clustering

- Clustering is used to group data based on some measure of proximity or similarity



Clustering

- Clustering is used to group data based on some measure of proximity or similarity



- It's a widely used technique because it allows to reconstruct classes of objects when there is no truth information available

Density-based clustering

Density-based clustering finds clusters by indentifying regions with **high density** of points.

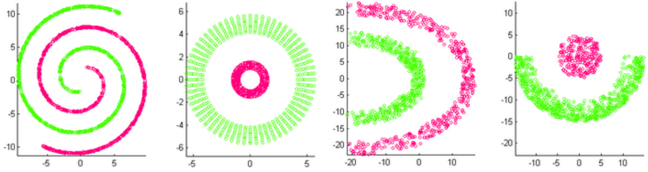
Density-based clustering

Density-based clustering finds clusters by indentifying regions with **high density** of points.
This presents several advantages:

Density-based clustering

Density-based clustering finds clusters by indentifying regions with **high density** of points.
This presents several advantages:

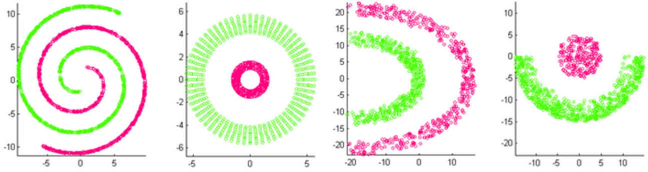
- it works well with clusters of irregular shapes



Density-based clustering

Density-based clustering finds clusters by indentifying regions with **high density** of points.
This presents several advantages:

- it works well with clusters of irregular shapes

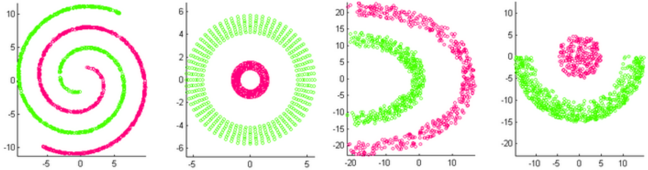


- it doesn't require a-priori knowledge of the number of clusters

Density-based clustering

Density-based clustering finds clusters by indentifying regions with **high density** of points.
This presents several advantages:

- it works well with clusters of irregular shapes



- it doesn't require a-priori knowledge of the number of clusters
- it works well with data full of noise and outliers

The importance of weighted clustering

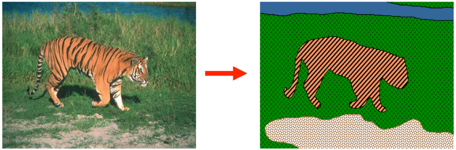
- In weighted clustering each point is assigned a weight
- The clusters are constructed taking the weights into account

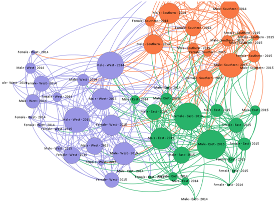
The importance of weighted clustering

- In weighted clustering each point is assigned a weight
- The clusters are constructed taking the weights into account
- Weights can be used to represent: signal measures, prior knowledge

The importance of weighted clustering

- In weighted clustering each point is assigned a weight
- The clusters are constructed taking the weights into account
- Weights can be used to represent: signal measures, prior knowledge
- Widely used in many applications: customer segmentation, image segmentation, social network analysis, anomaly detection, biological analysis





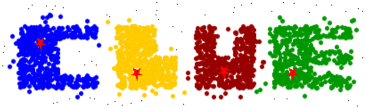
There is a need for weighted density-based solutions

- Most density-based algorithms don't support weighted clustering out-of-the-box
  - ➙ They need hand-made modifications to the dataset or to the distance matrix

There is a need for weighted density-based solutions

- Most density-based algorithms don't support weighted clustering out-of-the-box
  - ➙ They need hand-made modifications to the dataset or to the distance matrix
- There is a need for an alternative solution that combines the power of **density-based** algorithms with the generality of **weighted** clustering

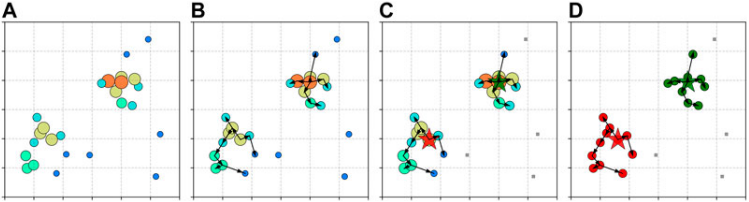
The CLUE algorithm



- CLUE (CLUstering of Energy) is a density-based clustering algorithm used in the CMS experiment at LHC
- It was originally designed for the clustering of hits in the calorimeters
- Each point has a weight which is used when calculating the densities
- The weights are the energy deposit measurements of the detector layer sensors

Reference: https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2020.591315/full#B16

Description of the algorithm



a → Computation of the local density for each point

b → Selection of the *nearest highers*

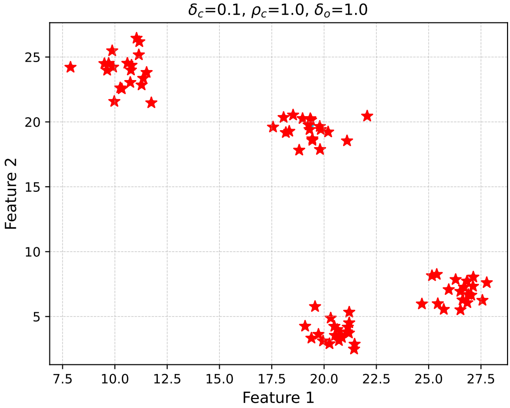
c → Finding clusters and outliers

d → Assigning clusters

The parameters of CLUE

3 parameters:

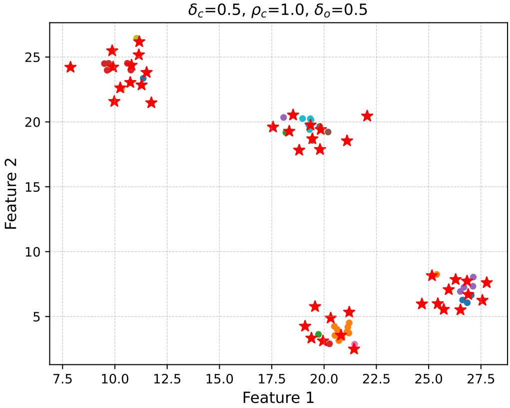
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

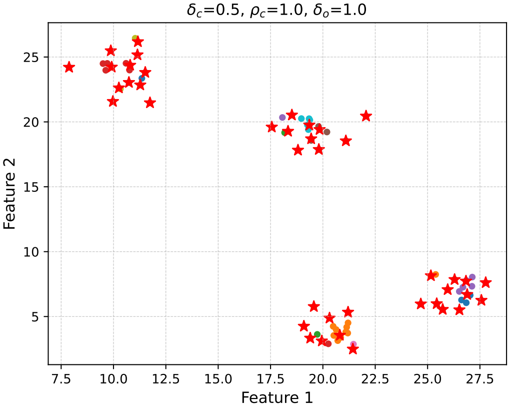
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

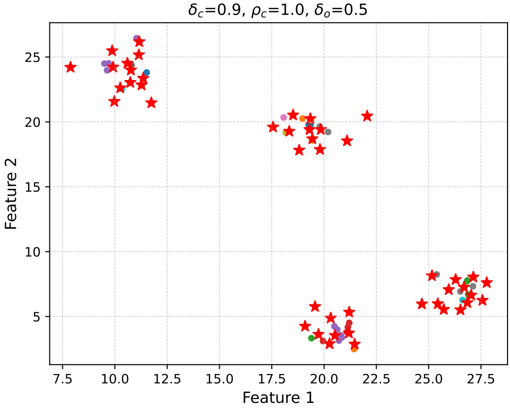
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

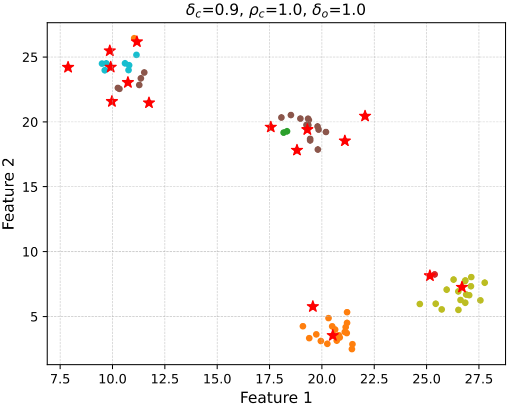
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

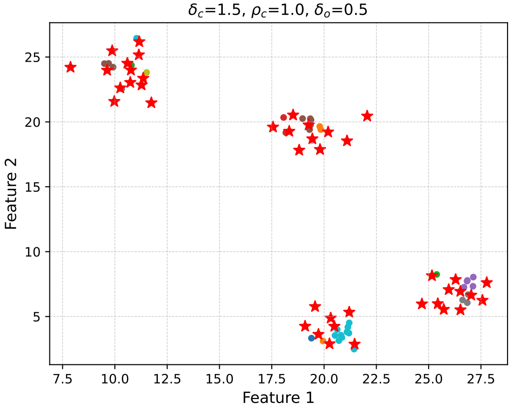
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

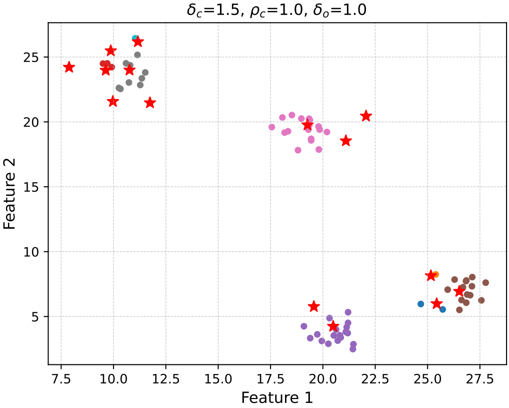
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

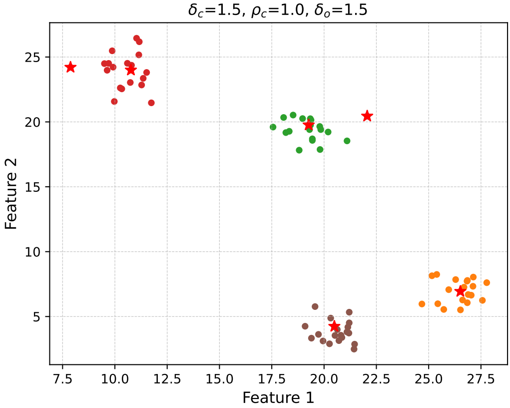
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

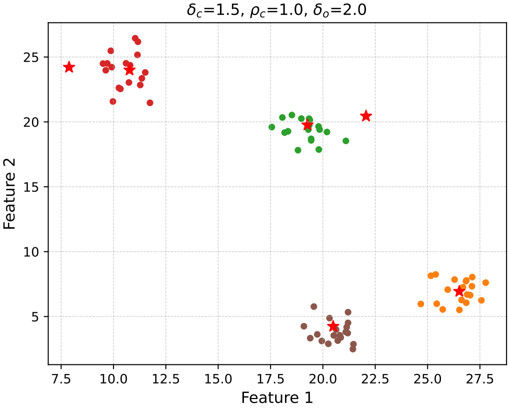
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

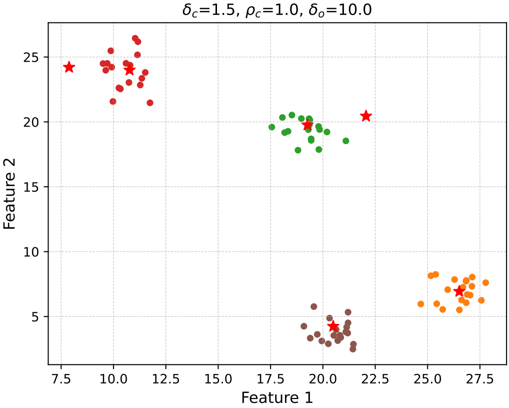
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

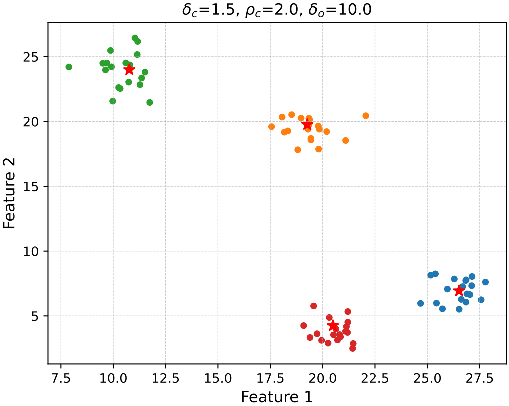
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

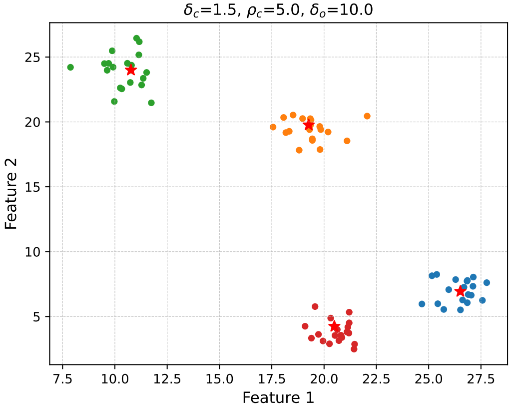
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

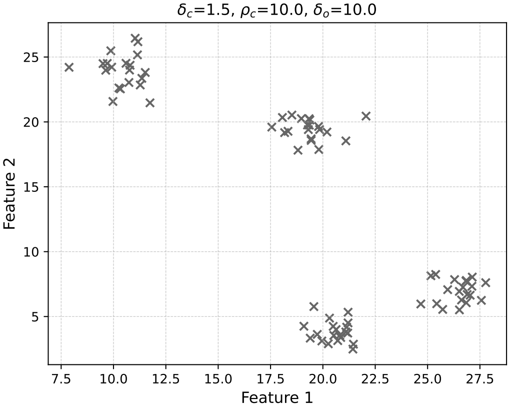
- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



The parameters of CLUE

3 parameters:

- δ_c, size of query range for computation of local density
- ρ_c, density cut-off for promotion to cluster seed
- δ_o, size of query range for cluster extension



From CLUE ...

- CLUE was specifically tailored to work in the CMS detector
- 2-dimensional clustering for each of the layers
- Could not be used on a general dataset

From CLUE ... to CLUEstering

- CLUEstering provides a generalization of CLUE
  - ➙ It's a general-purpose library
  - ➙ Applicable to any number of dimensions

From CLUE ... to CLUEstering

- CLUEstering provides a generalization of CLUE
  - ➙ It's a general-purpose library
  - ➙ Applicable to any number of dimensions
- Provides a Python interface to the C++ backend, which makes it easily usable by the machine learning community

What can CLUEstering be used for?

- The implementation of CLUEstering makes it very general

What can CLUEstering be used for?

- The implementation of CLUEstering makes it very general
- It can be applied to a large variety of problems that use clustering
  - ➙ in particular density-based clustering

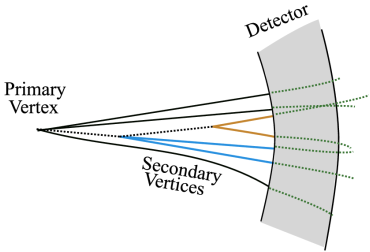
What can CLUEstering be used for?

- The implementation of CLUEstering makes it very general
- It can be applied to a large variety of problems that use clustering
  - ➙ in particular density-based clustering
- The main requirement is that data provides numerical coordinates

What can CLUEstering be used for?

- The implementation of CLUEstering makes it very general
- It can be applied to a large variety of problems that use clustering
  - ➞ in particular density-based clustering
- The main requirement is that data provides numerical coordinates
- Two examples:
  - ➞ vertex reconstruction in particle physics
  - ➞ star detection in astronomy

Example 1: Vertex reconstruction (vertexing)



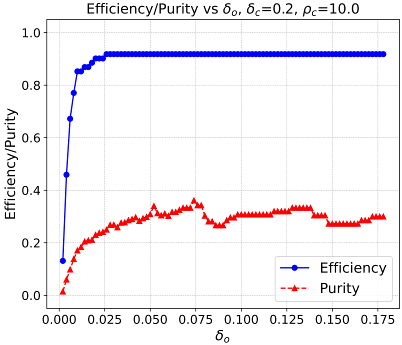
- Vertexing is the reconstruction of the interaction points (vertices) of the particle tracks
- The reconstructed vertices (recos) are compared to simulated vertices (sims)
- There is a match if recos and sims share at least 40% of the points (tracks)
- The relationship between recos and sims is a Many-To-Many

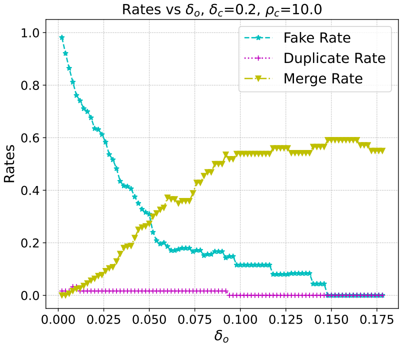
Useful definitions in vertexing

- **Efficiency**: fraction of sims associated to at least one reco
- **pure:** a reco where less than 20% of the points are noise
- **duplicate:** a sim associated to more than 1 reco
- **merged:** a reco associated to more than 1 sim
- **fake:** a reco associated to 0 sims

CLUEstering for vertexing

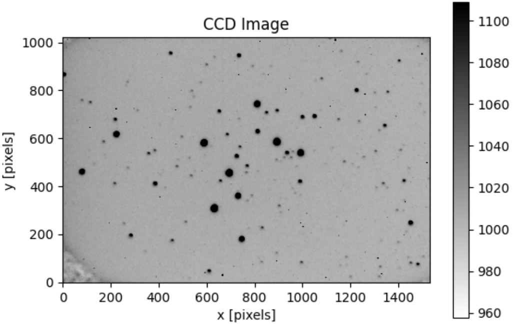
Simulation from an official sample of the CMS tracker community.
Clustering done using the z coordinate of the tracks and p_T as weight





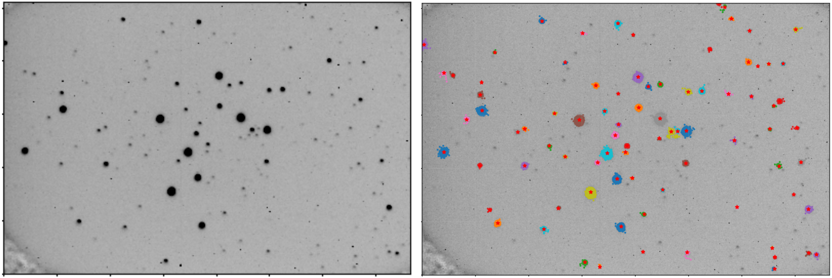
Reminder: δ_o indicates the size of query region used for cluster extension

Example 2: Star detection in astronomy



- Modern telescopes use CCDs (charge-coupled devices) to convert impinging photons into electrons
- Each pixel contains the number of electrons

Comparison of the PSF (Point Spread Function) image and the stars detected by CLUEstering



The advent of big data

- The amount of data being produced keeps increasing in every branch of science
- Software needs to continually improve in order to handle the increasing volume of data

The software portability challenge

- Nowadays there are many different types of processors available
- Heterogeneous computing platforms are becoming increasingly popular for demanding tasks

The software portability challenge

- Nowadays there are many different types of processors available
- Heterogeneous computing platforms are becoming increasingly popular for demanding tasks

- To support several platforms, often many different code-bases have to be developed and maintained
- We want to write software in a way that works on **many** possible platforms while achieving **near-native performance** for each one

The software portability challenge

- Nowadays there are many different types of processors available
- Heterogeneous computing platforms are becoming increasingly popular for demanding tasks

- To support several platforms, often many different code-bases have to be developed and maintained
- We want to write software in a way that works on **many** possible platforms while achieving **near-native performance** for each one
  - → Performance portability libraries

Software performance portability

- Performance portability libraries allow to:
  write code once → compile backends separately → run on different backends

Software performance portability

- Performance portability libraries allow to:
  write code once → compile backends separately → run on different backends
- There are many options currently available or under development

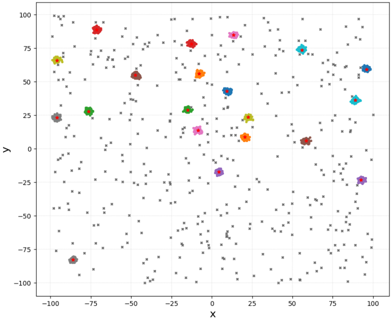
Performance portability in CLUEstering

- CLUE is a highly-parallel algorithm
- It's designed to work well on heterogeneous platforms
- The backend of CLUEstering is implemented with Alpaka
- Users can run the clustering on any backend with a single command

alpaka

https://github.com/alpaka-group/alpaka

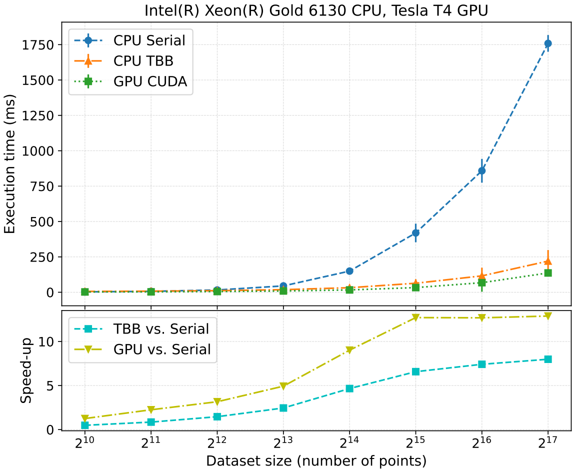
A representative dataset



- This dataset is representative of a common clustering ploblem
- The clusters are surrounded by noise, which mimics physical data
- CLUEstering reconstructs all the clusters correctly and the results are not affected by the noise

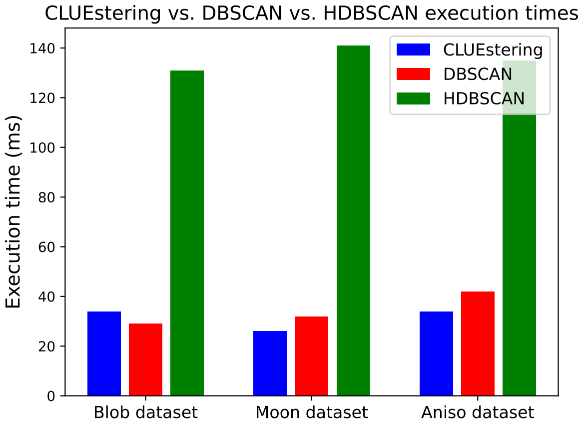
Scaling with dataset size

Parallel accelerators provide a 10× speed-up with respect to serial execution.



Comparison with other algorithms

Finally, how does CLUEstering compare with two of the most popular density-based algorithms against typical benchmarking datasets?



Note: The lower the better.

Conclusions

- CLUEstering is a **density-based weighted** clustering library

Conclusions

- CLUEstering is a **density-based weighted** clustering library
- It can be applied to almost any clustering problem

Conclusions

- CLUEstering is a **density-based weighted** clustering library
- It can be applied to almost any clustering problem
- Its use of heterogeneous platforms and its performance portability make it stand out from most other clustering libraries

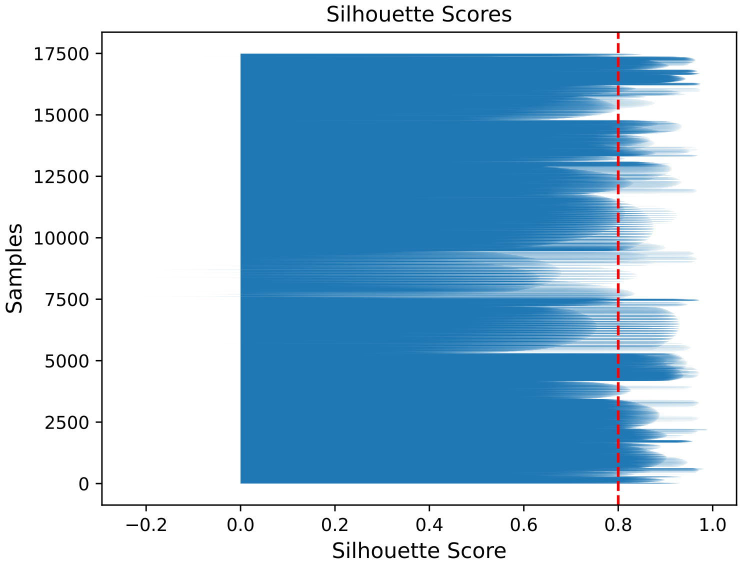
Conclusions

- CLUEstering is a **density-based weighted** clustering library
- It can be applied to almost any clustering problem
- Its use of heterogeneous platforms and its performance portability make it stand out from most other clustering libraries
- It's open source and available on github
  `https://github.com/cms-patatrack/CLUEstering`

Conclusions

- CLUEstering is a **density-based weighted** clustering library
- It can be applied to almost any clustering problem
- Its use of heterogeneous platforms and its performance portability make it stand out from most other clustering libraries
- It's open source and available on github `https://github.com/cms-patatrack/CLUEstering`
- Can be easily installed with a simple `pip install` command

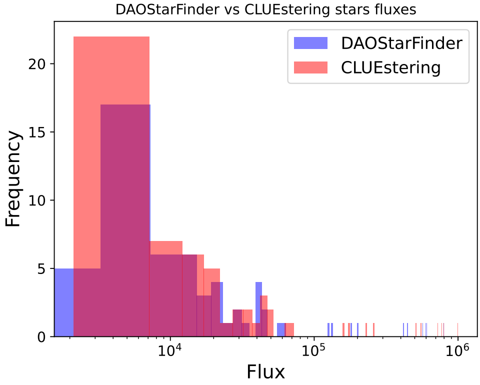
CLUEstering for star detection: silhouette scores



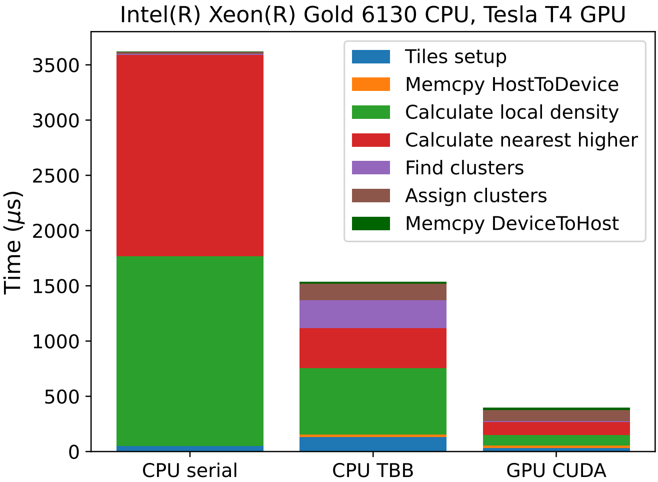
CLUEstering for star detection: comparison of fluxes

Compare the fluxes obtained using CLUEstering with DAOStarFinder and aperture photometry.

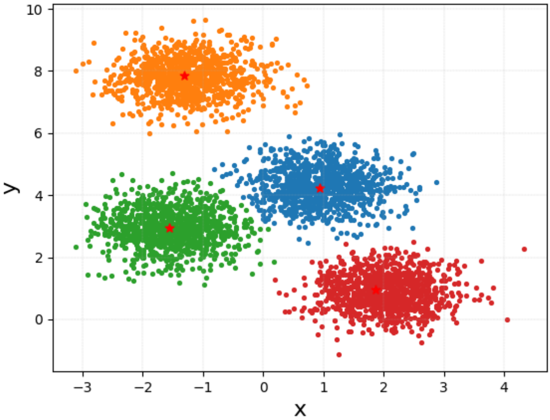
Execution time: $59 \pm 2\,ms$ for CLUE and $262 \pm 15\,ms$ for DAOStarFinder



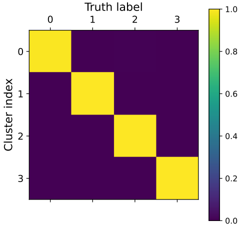
Profiling results

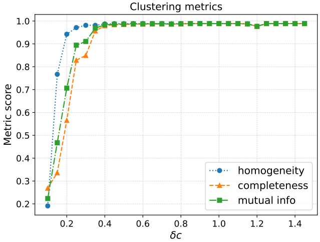


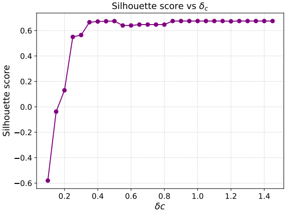
Blob datasets

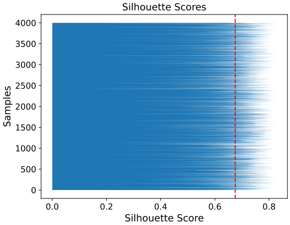


Results on the blob dataset

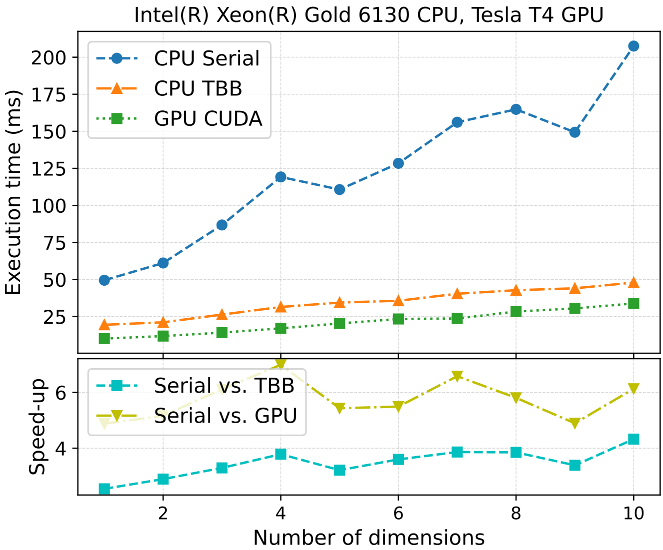




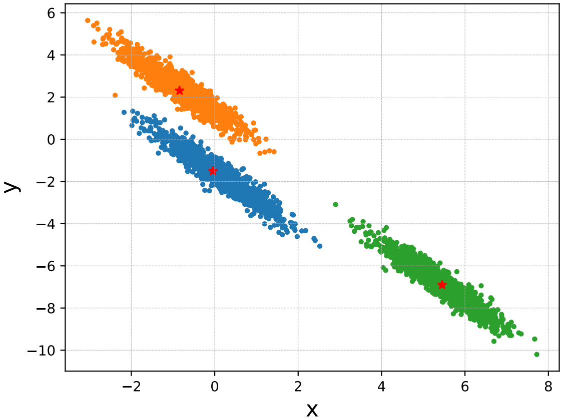




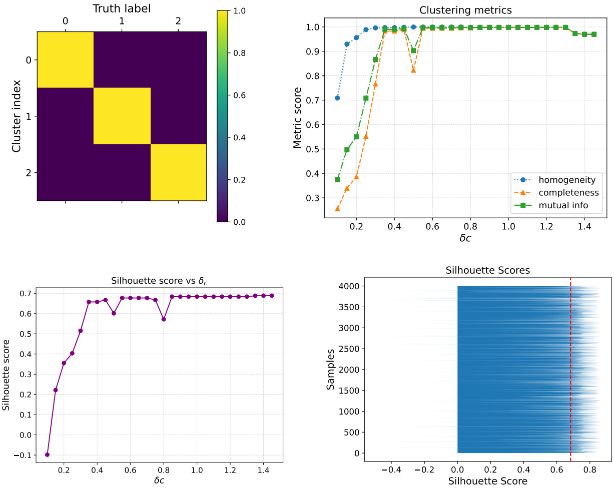
Scaling with number of dimensions



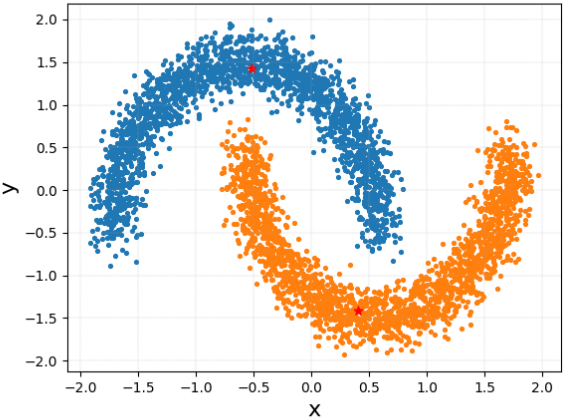
Aniso dataset



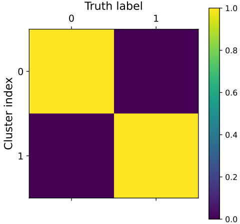
Results on the aniso dataset

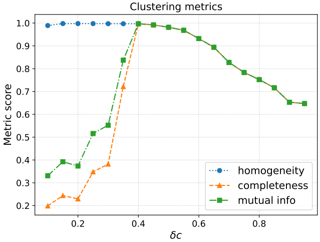


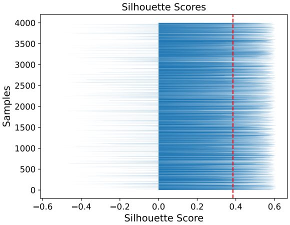
Moon dataset

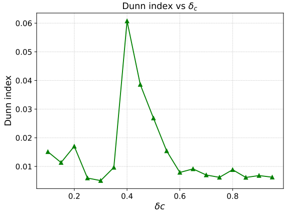


Results on the moon dataset









Backup: CLUEstering for vertexing (cont.)

Clustering done using the z coordinate of the tracks and p_T as weight