EDAMI Project – First Report

Implementation of the OPTICS algorithm

Daniel Bannister - daniel.bannister.stud@pw.edu.pl

1) Task and algorithm description

The task consists in the implementation of the OPTICS algorithm as described in the paper *"OPTICS: Ordering Points To Identify the Clustering Structure"* [1]. The OPTICS algorithm does not explicitly perform clustering of a dataset. It rather organizes the dataset to represent its density-based clustering structure. This allows access to information at all clustering levels and simplifies the analysis of this information. It also requires less memory resources.

In fact, a density-based clustering structure is a set of parameters that help describe data as a set of clusters, i.e. they describe how we can group data together within a dataset. These parameters are related to the key notions of neighborhood and core object:

- A core object is an object that has more neighbors than a defined minimum
- An object A is a neighbor of object B if the distance between A and B is less than a defined distance ε

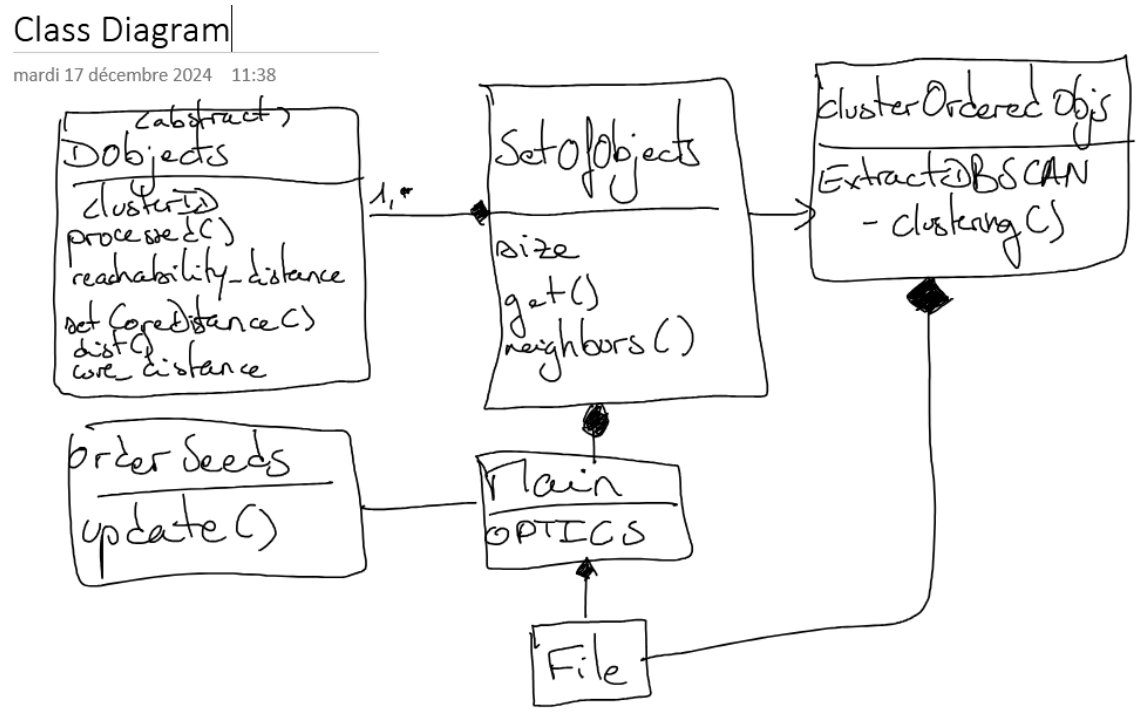
The parameters can be either global or local and can take an infinite number of values. A slight change in parameters induces a huge difference in the number of clusters detected. The OPTICS algorithm helps overcome this problem by ordering the database, rather than giving the density-based cluster parameters. The clustering structure is thus implicit.

The theory the algorithm is based on is detailed in the paper. Since the algorithm is already described in pseudo-code (figures 5 to 8), what is left to do is the actual writing of the code.

2) Implementation technology

The OPTICS algorithm can be applied to any type of dataset, as long as it is provided a definition of "distance" between objects. Furthermore, the pseudo-code suggests the use of methods and attributes. That's the reason why I decided to use the Java programming language to implement the algorithm. This Object-Oriented-Programming language is adapted to the implementation of notions such as abstraction,

polymorphism and inheritance that will be used in this project. Most notably, we will be able to extend the algorithm to any datatype by manipulating an abstract “DObject” class. This class will derive from the standard “Object” class [3], with additional attributes and methods. The following class graph illustrates the links between all the classes that will be used in this project:



3) Datasets and experiments

We will mostly work with the Iris dataset [2]. The data consists of 150 instances of irises, that are described with 5 attributes: 4 integers and a species (Iris Setosa, Iris Versicolour, or Iris Virginica). Classification involves assigning data into predefined categories based on specific attributes, whereas clustering groups data into clusters based on similarities without predefined labels. In addition to standard evaluation metrics of clustering algorithms, we can thus evaluate the performances of the algorithm by examining how many irises of the same species have been clustered together. We can use other datasets to prove the consistency of our algorithm with different datatypes.

Other performance metrics include:

- Silhouette Score
- Davies-Bouldin Index
- Calinski-Harabasz Index (Variance Ratio Criterion)
- Adjusted Rand Index (ARI)
- Mutual Information (MI)
- Steps to Evaluate Clustering Using Sklearn

[4]

Finally, we can compare our implementation and the results it gives us to the Python implementation given in the sickit-learn library [5]

4) Bibliography

1 - Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander, *"OPTICS: Ordering Points To Identify the Clustering Structure"*, Proceedings of the 1999 ACM SIGMOD international conference on Management of data (SIGMOD '99)

2 - https://archive.ics.uci.edu/dataset/53/iris

3 - https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html

4 - https://www.geeksforgeeks.org/clustering-metrics/

5 - https://scikit-learn.org/dev/modules/generated/sklearn.cluster.OPTICS.html