One batch greedy medoids: A new heuristic to the k-medoids problem on large scale datasets

Masters internship report

Nicolás Enrique Cecchi

Internship carried out from 01/05/2023 to 31/07/2023

Internship tutors : Mme. MOUGEOT, Mathilde and M. DE MATHELIN, Antoine
Referent teacher : Mme. MOUGEOT, Mathilde

Educational institution : Université Paris-Saclay / ENSIIE - M1 Applied Mathematics
Internship host : Centre Borelli - 4 Av. des Sciences, 91190 Gif-sur-Yvette

Abstract

During the three months internship period the k-medoids problem was studied in depth. A literature review was performed to understand the historical evolution of this algorithm, its origins, why it came to be and the existing proposed solutions, up to the state of the art. At this point, a new heuristic was proposed to tackle the main problems of many of the algorithms that are memory requirements and run-time when run on large scale datasets. After performing many experiments, we could verify that the proposed solution is performant and competitive.

Key words: clustering, k-medoids, partitioning, heuristic

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

This is an internship report of the work performed between 02/05/2023 and 31/07/2023 in the Centre Borelli, at the École Normale Supérieure - Paris Saclay. The Centre Borelli is a research unit of the CNRS that brings together specialist from applied mathematics, computer science and neuroscience. The work of the laboratory is tightly related to applications to the biomedical field and industrial transfer.

Clustering is the task of finding, among a set of unlabeled data, the way of grouping the data into groups (clusters) who are internally as similar as possible and externally as dissimilar as possible. Many clustering paradigms exists, partitioning is the one where data-space is divided into regions that define the clusters. These regions are usually defined with respect to a reference point that is called the centroid of the cluster. In this project we focus on finding those representative points among the observed dataset, in that case they are referred to as by the name of medoid.

Nowadays, the acquisition and storage of data is cheaper than never before. Datasets' sizes have been growing and are expected to continue doing so, given the value that can be extracted from them. To keep up with this bigger datasets, practitioners need, not only more and more-powerful hardware, but also efficient algorithms. The objective of the work is to propose a new algorithm to solve the k-medoids problem, overcoming the limitations of current available algorithms: run-time and memory requirements on large scale datasets. To do so, an understanding of the history and state-of-the-art is necessary. Furthermore, we provide an open source Python implementation of the algorithm, as well as an extensive benchmark to compare with existing alternatives.

In section 2 we present the Centre Borelli, the host institution, its history, its present work and its composition. In section 3 we discuss the medoids clustering problem and formally present it as a clustering problem and a facility location problem. Then, in section 4 we discuss the algorithms included for benchmarking, and introduce our contribution. Following, section 5 explains the methodology used for the work and in section 6 we present the results. Finally, section 7 and 8 are dedicated to the conclusions and future work, respectively.

2 Centre Borelli

In this section is introduced the Centre Borelli, the host institution for all the work done in this internship. First, in section 2.1, is presented the general organizational structure. In section 2.2 we introduce the main facts about the research done in Centre Borelli.

2.1 Organizational structure

The Centre Borelli is a multidisciplinary joint research unit (UMR 9010) that brings together researchers from the fields of mathematics, computer science and neuroscience, notably involved in interfaces with the biomedical field and industrial transfer. It is located on many sites among the ENS Paris-Saclay, the Université Paris Cité , the Army training hospital HIA Bégin and HIA Percy. It was born in 2019 as the merger of two CNRS Units: CMLA (Centre de Mathématiques et de Leurs Applications), from ENS Cachan (now ENS Paris Saclay), and Cognac-G (Cognition and Action Group), from Université Paris-Descartes.

The laboratory is affiliated to five institutions that provide financial, human and logistic resources: ENS Paris-Saclay, CNRS, Université Paris-Cité, Inserm and Service de Santé des Armées.

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

Figure 1: Logos of the institutions that conform the Centre Borelli. (a)ENS Paris-Saclay (b) Université Paris Cité (c) INSERM (d) CNRS (e) SSA

Centre Borelli develops fundamental knowledge in neuroscience and mathematics, scientific methods in modeling, simulation and processing of complex data, clinical approaches and software tools. The research is interdisciplinary and global: including the theoretical bases, in situ observations, tangible developments in the form of industrial code, internet platforms or mobile applications, and always prioritizing the canons of reproducible science.

2.2 Research at Centre Borelli

The Centre Borelli has over one hundred members, including about fifty permanent researchers and clinicians, and hosts about one hundred doctoral and post-doctoral students involved in three research areas.

2.2.1 Research axis

The research of the laboratory focuses on three main axis:

1. **Mathematical foundations and algorithms for artificial perception and learning**, covering image, video and signal processing and analysis, machine learning and network science.

2. **Modeling, mathematical analysis and simulation of complex physical, natural and biological phenomena**, focused on developing and analyzing numerical models of complex systems. The unit includes mathematicians, numerical analysts, statisticians, biologists, mechanics and physicists.

3. **Integrative and behavioral neurosciences in humans and animals**, which aims to formalize the quantitative study of human behavior in normal, altered or pathological situations. This unit is composed of neurophysiologists, clinicians, ergonomists, psychiatrists and statisticians.

This internship was performed in the context of the first of these axis, reason for which we delve deeper into it.

2.2.2 Mathematical foundations and algorithms for artificial perception and learning

This group works with image, video and signal processing and analysis, machine learning and network science.

Its works have a pragmatic approach with a strong theoretical basis and regularly contributes to the design and improvement of observations instruments, processing pipelines for image, signal and videos. The group is also pioneer in the publication of algorithms with a strong reproducibility.

The covered topics include:

- **Processing and analysis of images and videos:** denoising, deblurring, digital forensics, pattern recognition, event or anomaly detection.
- **Signal processing, trajectories, signals on graphs:** filtering, segmentation, change-point detection, representation learning, sparse coding, multi-modality, pattern recognition. Mainly focused on industrial and biomedical fields.
- **Graphs and networks:** diffusion processes on graphs, resource allocation algorithms for control, signals on graphs. Applied mainly to epidemiology, marketing, biology and operations research.
- **Machine learning:** Global optimization, federated learning, transfer learning and domain adaptation, scoring and ranking algorithms.

The group is subdivided into three divisions:

1. Machine learning and massive data analysis,
2. Image processing,
3. Artificial intelligence for data science and cybersecurity

This work was carried out in the *Machine learning and massive data analysis* division. This division deals with inference, predictive modeling and sequential optimization from complex data such as time series, functional data, network data. Its coordinator is M. Nicolas Vayatis, and the scientific leaders are Mme. Mathilde Mougeot and M. Laurent Oudre. At the time of this writing it hosts 4 post-docs researchers, 17 doctoral students and 3 internship students.

The work was done in collaboration with M. Antoine De Mathelin, final year PhD. student, and Mme. Mathilde Mougeot, his doctoral advisor, scientific leader at Centre Borelli and professor at ENSIIE and ENS Paris-Saclay.

3 K-Medoids

The k-medoids problem can be formally defined as:

Definition 3.1 (k-medoids problem)**.** *Given N data points $\chi = \{x_1, \ldots, x_N\}$, a dissimilarity function $d(\cdot, \cdot)$ and the set $\mathbb{M}_k$ of all possible subsets $\mathcal{M}$ of size k from χ. Find the subset $\mathcal{M}^* \in \mathbb{M}_k$ which minimizes the sum of dissimilarities between all data points in χ and their closest point in $\mathcal{M}^*$, $\mathcal{L}(\mathcal{M}^*)$:*

$$\mathcal{M}^* = \arg\min_{\mathcal{M} \in \mathbb{M}_k} \mathcal{L}(\mathcal{M}) \tag{1}$$

$$\textit{Where } \mathcal{L}(\mathcal{M}) = \sum_{i=1}^{N} \min_{m_j \in \mathcal{M}} d(m_j, x_i) \tag{2}$$

Informally, it can be summarized as the problem of finding, among N instances, the subset $\mathcal{M}^*$ of size $k \in \mathbb{N}, k \leq N$ such that the sum of the dissimilarities of non-selected instances to its closest selected-instance is minimized.

Since the number of data points is always finite, so is the number of possible partitions, and the global minima could be found by exhaustive search methods. However, this is only true in theory, since finding the globally optimal partition in known to be NP-hard [ÄK06].

It is worth at this stage to emphasize that, despite referring (as is commonly done) to k-medoids as an algorithm, it should probably be called, in general, a *k-medoid problem* given that it tries to find k representative objects that serve as cluster centroids to which data points are assigned if a given dissimilarity is minimal. This is also true for other problems, such as k-means, that will be discussed in section 3.1. Many different algorithms exist for finding solutions to these problems, for a more detailed account, we refer the reader to Bocks's historical review [Boc07].

The k-medoids problem has been mainly studied by two scientific communities. On the one hand, in the field of operational research, it is studied in the context of the facility location and it is sometimes referred to as the *p-medians* problem [Dre95; TFL83; DM15]. On the other hand, the machine learning community studies it as a clustering problem [KKS14; PJ09; RRR04]. This generated the coexistence of some variations on the vocabulary and manners of formalizing, analyzing and generating variants. For example, in more general contexts, the points in the optimal solution $\mathcal{M}^*$ may receive names such as centroids [XI05], prototypes [Tan+05; XI05; XT15] or centrotypes [KAU87; KR90b].

In this internship we adopt the machine learning perspective and, in the following, we refer to them as *medoids* to distinguish them from the classical notion of median [KR90a] and to emphasize the problem we solve.

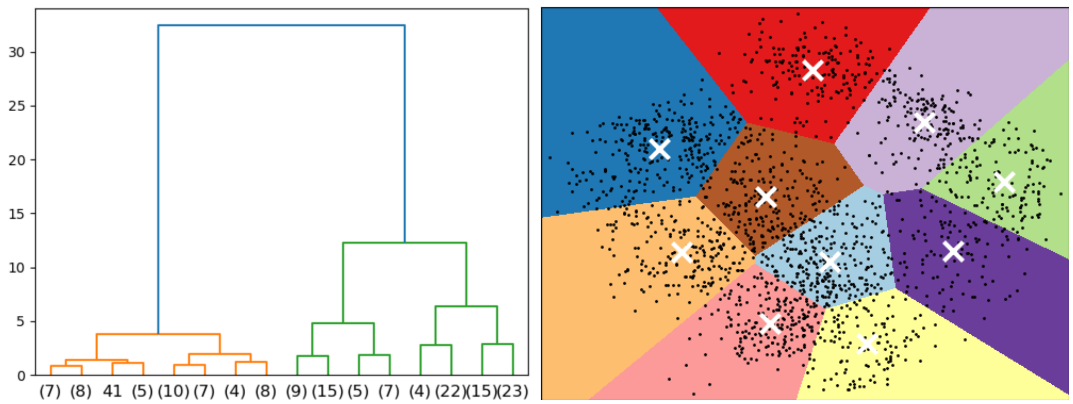
In section 3.1 we make an introduction to the problem of clustering. Section 3.2 is devoted to present the k-medoids problem from the machine learning perspective. The discussion of the facility location problem and its relation the k-medoids problem are left to Annexes B.1

and B.2, respectively.

3.1 Clustering

Clustering is an unsupervised learning problem [Bis06; Mur12] in which objects are grouped together in such a way that those within each cluster are more closely related to one another than objects assigned to different clusters [HFT17]. As pointed out by [XT15] and Everitt 1993, as cited by [XI05], there is no total agreement on what constitutes a cluster. Also Everitt's 1993, as cited by [GMW07], has stated that *"if using a term such as cluster produces an answer of value to the investigators, then it is all that is required"*. Somewhat in the middle ground we find that [Tan+05] says that *"cluster analysis divides data into groups (clusters) that are meaningful, useful, or both"*. This difficulty to define a cluster in a precise manner is the cause for the existence of the great number of clustering algorithms [Est02].

Many clustering techniques exist, traditionally it has been emphasized the importance of distinguishing between hierarchical and non-hierarchical clustering methods [MPK83], while more recent work have identified as many as nine [XT15] or more [JM18] different groups of clustering algorithms. Figure 2 below shows on the left a typical dendrogram representing hierarchical clustering. In this diagram, the root (top) represents a node with all the data-points grouped together. As we descend through the branches of the dendrogram, points are assigned to either to the left or the right one, up until each datapoint is the only one on its node. Algorithms that work from bottom up are called *agglomerative* and those that work from top to bottom are called *divisive*. On the right, we see the case of a non-hierarchical clustering, in particular, the image shows the partitioning (see Section 3.2, below) of a 2-dimensional space into 10 clusters. Each region of the plane is defined by one centroid (white cross) that is the most representative point of the cluster, according to a given measure.



(a) Hierarchical clustering example. (b) Partition based clustering example.

Figure 2: Examples of hierarchical and partition-based clustering [Ped+11].

3.2 K-medoids as a clustering problem

Studying the k-medoids problem from a clustering perspective, we say that it is part of the family of partition-based algorithms. It makes a division of the space where data objects belong to non-overlapping subsets such that each data object is in exactly one of them [Tan+05]. In k-medoids and related problems, the partition is found by selecting k points which serve as reference for classifying the other points.

One popular algorithmic strategy when solving these problems is using an iterative relocation algorithm that minimizes a given criterion by iteratively relocating data points between clusters until a (locally) optimal partition is attained, a general case of an iterative relocation algorithm is shown below, in Algorithm 1 [ÄK06]. To work, the algorithm needs as input the number K of clusters, together with an initialization routine and criteria for membership-computation, centroid update and stopping.

Algorithm 1 Pseudo-code of an iterative relocation algorithm

Require: $1 \leq K \leq N, K \in \mathbb{Z}$

1: Initialize K centroids
2: (Re)compute membership for data points with current centroids
3: Update some/all cluster centers/distributions according to new memberships
4: Repeat from step 2 until stopping criteria is met

Probably the most well-known iterative algorithm for clustering is Lloyd-Forgy solution to the k-means problem [Llo82; For65]. We succinctly present both in Algorithm 2 and Definition 3.2, below.

Definition 3.2 (k-means problem)**.** *Given a set of N points $\chi = \{x_1, \ldots, x_N\}$ with $x_i \in \mathbb{R}^d, \forall i = 1, \ldots, N$ and given $\mathbb{C}_k$ the set of all possible subsets of size k in $\mathbb{R}^d$. Find the set $\mathcal{C}^* \in \mathbb{C}_k$ which minimizes the sum of euclidean distances of the points in χ to the closest point in $\mathcal{C}^*$.*

$$\mathcal{C}^* = \arg\min_{\mathcal{C} \in \mathbb{C}_k} \sum_{i=1}^{N} \min_{c^j \in \mathcal{C}} ||x_i - c^j||_2^2 \tag{3}$$

Algorithm 2 Pseudo-code of the Lloyd-Forgy algorithm

1: Initialize the k centroids as k random points from χ
2: **while** Stopping criteria not met **do**
3: Assign each point to its closest medoid
4: Update centroids as the mean of the points in each cluster
5: **end while**
6: **return** Centroids

Recalling the Definition 3.1 of the k-medoids problem, we can see the resemblance. Next, we discuss their main differences, to help the reader get a better understanding of the interest on k-medoids.

To begin with, in k-means the centroid of a cluster is selected as the arithmetic mean of the data points that belong to it, equivalently, the centroid of each cluster is a point in the space that minimizes the average squared euclidean distance to all other points in the cluster. This is not straightforward to generalize to other distance metrics that may be desirable in other contexts, such as l_1 and *cosine* distances in sparse data [Tiw+20]. Furthermore, being an arithmetic mean, the centroids are not in general a point in the dataset and may not be interpretable in many applications, such as digital images or recommendations systems, as

noted by Leskovec, cited by [Tiw+20]. In contrast, in k-medoids the centroids are defined as the points in the dataset that minimize the average dissimilarity to the other points in that cluster. In this definition we can see that both drawbacks of k-means are solved. First, the medoid minimizes the average *dissimilarity*, no assumption is made on the space of the data and any dissimilarity function is suited, even asymmetric dissimilarity measures can be used. Second, the centroid is a point *in the dataset* by definition, meaning that no matter the context, it will always be interpretable. Figure 3, from the scikit-learn-extra package [YTM23], shows different partitions of a 2 dimensional PCA reduction of the digits dataset using k-means and k-medoids with 3 different dissimilarities.

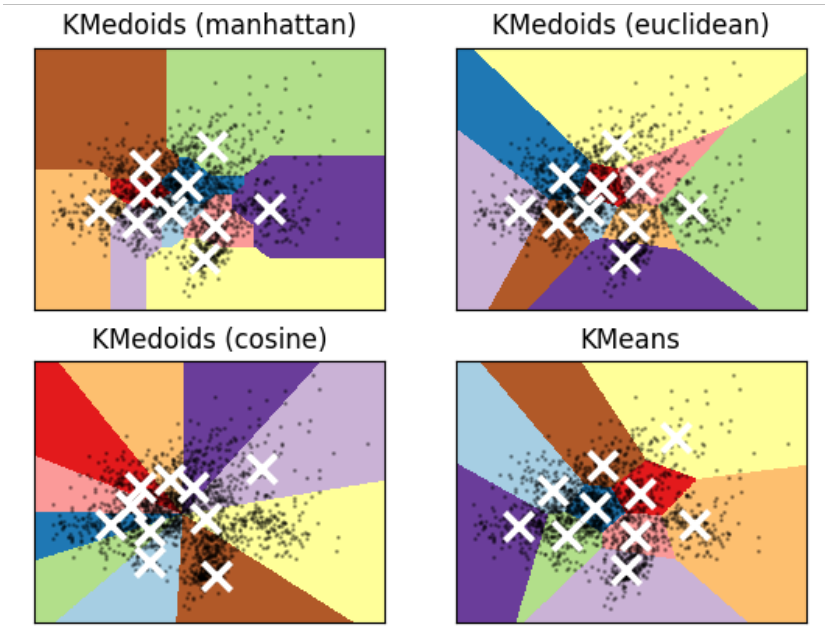


Figure 3: Different partitions of the 2 principal components of the digits dataset computed by: k-medoids with manhattan distance (upper left), k-medoids with euclidean distance (upper right), k-medoids with cosine distance (lower left) and k-means (lower right). Image from [Ped+11].

Figure 4 shows the 10 medoids found both on MNIST handwritten digits dataset [LCB10] and Fashion-MNIST [XRV17] datasets with an algorithm from the apricot-select Python package [SBN20]. While we can see that the selection is not perfect (there are two sixs and no five; and two ankle boots and no sandal, respectively), it does find elements in the dataset that are not only arithmetically but also semantically representative of their classes.

Figure 4: Medoids found for the digist-mnist data set (left) and the fashion-mnist dataset (right) with the apricot-select Python package [SBN20].

4 Algorithms

In this section we present the algorithms that were used in the experiments performed. In section 4.1 we discuss algorithms from the k-medoids Python package, which include PAM and many of its variants, including BanditPAM that introduces an stochastic perspective. Section 4.3 is dedicated to other simpler algorithms that were also used mainly for having reference values. Next, section 4.2 is concerned with the facility location optimization problem as posed in the `apricot-select` package. Finally, in section 4.4 we present our solution, *one batch greedy medoids*.

4.1 PAM and its variants

The `k-medoids` Python package is a wrapper around a Rust package [Sch21], implementing the FasterPAM and FastPAM [SR19; SR21] algorithms along with the baseline k-means-style and PAM [KR90a] algorithms. Furthermore, the (Medoid) Silhouette [Rou87] can be optimized by the FasterMSC [LS22], FastMSC [LS22], PAMMEDSIL and PAMSIL [LPB03] algorithms. All algorithms expect a distance matrix and the number of clusters as input. They hence can be used with arbitrary dissimilarity functions. We only discuss in detail the PAM algorithm, the others are variations of it, therefore only the main differences are addressed. PAM stands for *partitioning around medoids*: it generates a partition of the input space by assigning points to the cluster represented by the closest point selected as a medoid. The algorithm works in two phases: BUILD and SWAP.

The BUILD phase, constitutes an initialization routine. It selects the first k objects that will act as cluster medoids. The first object is selected as the one which has the minimal dissimilarity with respect to every other point. It is the medoid of the entire dataset. The subsequent $k-1$ points are chosen such that, at each step, the selected point is the one that decreases the most the objective function. Once k medoids are selected, the SWAP phase begins. At each iteration, all pairs (m, n), for $m \in \mathcal{M}$ (the set of medoids) and $n \in N$ (the set of non-medoids) are considered for swapping sets. The swap that decreases the most the objective function is performed, until no possible swap decreases the objective function.

Schubert and Rousseeuw [SR19; SR21] modify PAM in a way that allows to prove a speedup factor of $\mathcal{O}(k)$ in the SWAP phase, with respect to PAM by completely eliminating the nested loop of length k, which they call *FastPAM1*. Furthermore, they present *FasterPAM* [SR21] a new variant that eagerly (as soon as it finds an improvement) swaps pairs of medoid/non-medoid, further accelerating runtime. Van der Laan, Pollard and Bryan [LPB03] decide to study variations of PAM to improve its capabilities of finding relatively small clusters in situations where good partitions around medoids clearly exist. Two algorithms are proposed: PAMSIL, which aims to maximize the silhouette, and PAMMEDSIL, which maximizes the *medoid-based silhouette*, a quick approximation of the silhouette.

FastMSC is an accelerated version of PAMMEDSIL clustering, that performs the same swaps as the original PAMMEDSIL (given the same starting conditions), but finds the best swap $\mathcal{O}(k^2)$ times faster. *FasterMSC* accelerates PAMMEDSIL by eagerly performing the swaps, also achieving a $\mathcal{O}(k^2)$ performance improvement.

The last algorithm that is closely related to PAM is BanditPAM [Tiw+20], a randomized algorithm that achieves with high probability the same clustering as State-of-the-art alternatives, requiring significant less memory ressources. It is implemented by its authors in a

separate Python package[1], built in C++ for higher performance.

BanditPAM treats each phase of the PAM algorithm as a statistical estimation problem. It is first shown that each BUILD step and SWAP iteration are equivalent to a best-arm identification problem from the multi-armed bandits. Arms are identified with all the points or all the medoid/non-medoid pairs, in the BUILD and SWAP phases respectively. The best candidate is then estimated via a successive elimination of the medoid candidates with the upper confidence bound algorithm.

4.2 Greedy algorithms

Definition 4.1 (Greedy algorithm)**.** *An algorithm is said to be greedy if at each step it makes the locally optimal choice.*

The `apricot-select` Python package implements many greedy algorithms that solve the k-medoids problem. It has been shown that the quality of the solution found by greedy algorithms cannot be worse than $1 - e^{-1}$ of the optimal value [NWF78].

Below, Table 4.2 presents the algorithms that were used in our experiences.

Algorithm	Comments
Naive greedy	It is the simplest optimizer. It iterates over all non selected examples and chooses from the ground set the one that generates the largest gain, until k elements where selected.
Lazy greedy [Min78]	Also known as accelerated greedy , keeps a priority queue to avoid re-evaluating samples known to provide too little gain.
Two-stage greedy	It combines two optimizers[2], selecting the first n samples with the first one and the remaining $k - n$ with the other.
Approximate lazy greedy [WIB14]	A variation of lazy greedy to reduce the number of function evaluations. Rather than insisting on the finding the element which generates the maximal gain, it selects elements with a gain close to the maximum, up to a given percentage, which controls the level of sub-optimilatity.
Stochastic greedy [Mir+14]	Selects elements which improve the value of the solution approximately the same as greedy, but in a faster manner thanks to the implementation of a subsampling step.
Sample greedy [Mir+14]	Selects the samples by applying lazy-greedy to a subsample of the dataset.
Modular greedy	Approximates the target function by its modular upper-bound, and selects the k objects that would provide the maximal gain in that context.

[1] https://github.com/motiwari/BanditPAM

[2] By default, the first algorithm is the naive greedy optimizer and the second algorithm is the lazy greedy.

4.3 Other clustering algorithms

Other simpler strategies were also used to select the k centroids of the clusters to help benchmarking. These include: *random*: randomly choosing k points from the dataset; *kmeans++ initialization* [AV07]: selecting the k elements as performed in the kmeans++ initialization phase (from now on we will refer to this way of clustering only as *kmeanspp*) and *kcenters*: selecting the first element at random and then by iteratively selecting the point that would minimize the maximal intercluster distance.

4.4 Our contribution: one batch greedy medoids

Most of the previously developed algorithms have the drawback of requiring the computation of a dissimilarity matrix, therefore they have memory requirements of order $\mathcal{O}(N^2)$. This is a critical issue, given that today's datasets have usually many thousands or even millions instances. Consider, as an example, that the 1998's MNIST-digits dataset has 60 000 instances, requires approximately 14.4 GB of memory to store its dissimilarity matrix. This is already a challenge for most consumer-grade computers today. It is therefore crucial to develop algorithms that can efficiently handle large datasets. Few of the previously developed algorithms do not require the storage of this matrix, for instance,*kmeanspp* only requires $\mathcal{O}(KN)$ dissimilarity computations. However, they often generate a poor medoids selection. BanditPAM explicitly proposes to deal with this memory issue by iteratively selecting the dissimilarities to compute, but the time complexity remains important for large datasets.

For these reasons, we propose a new solution that allows one to work with larger datasets and have a good performance estimating medoids, even with home-grade hardware and in a fast manner. We call this algorithm *one batch greedy medoids* (OBGM).

Recall that given a dissimilarity $d(\cdot,\cdot)$ and set $\mathcal{M}$ of k points in the dataset, the objective function of the k-medoids problem can be written as in Equation 4:

$$\mathcal{L}(\mathcal{M}) = \sum_{i=1}^{N} \min_{m_j \in \mathcal{M}} d(m_j, x_i) \tag{4}$$

Then, define the gain G of re-assigning point x from its current medoid y to a new medoid z as:

$$G(x, y, z) = max(0, d(x, y) - d(x, z)) \tag{5}$$

If the dissimilarity between x and y is larger than that between x and z, then the gain is the difference between them. If it is smaller (or equal) then there is no gain.

The key ideas of OBGM are:

1. Taking a random sample of size $B < N$ *reference points* to compute a N by B dissimilarity matrix D. This reduces memory usage from $\mathcal{O}(N^2)$ to $\mathcal{O}(NB)$.
2. Select the first medoid as the point that has minimal average dissimilarity with the reference points.
3. Update D into a matrix of gains $\mathcal{G}$ by applying G.
4. Select new medoid as the point whose average gain is maximal.

5. Update $\mathcal{G}$. The iterative application of G generates sparsity in $\mathcal{G}$, allowing for a more efficient sparse storage.

6. Repeat 4 and 5 until k medoids were selected.

Another try

- Starts with an empty set of medoids.
- Takes a random sample of size $B < N$ (we will call them *reference points*) and computes a N by B dissimilarity matrix D. Reducing the memory requirements from $\mathcal{O}(N^2)$ to $\mathcal{O}(NB)$.
- The first medoid, m_1, is selected as the point in the dataset whose average dissimilarity against the B references is minimal.
- D is updated into a *gain matrix* $\mathcal{G}$, via the application of $G(i, m_1, j)$. For every row i and column j.
- Iteratively selects the next $k-1$ medoids by:
  - Select point whose average gain is maximal
  - Update matrix $\mathcal{G}$. Note that the iterative application of $G()$ generates sparsity, thus allowing to store $\mathcal{G}$ in a more efficient, sparse format.

OBGM begins with an empty set of medoids and greedily add points to this set, up to having the k medoids. At the beginning, the N points in the dataset are potential medoids. Instead of computing the N by N dissimilarity matrix, OBGM takes a random sample of size $B < N$ and computes a N by B dissimilarity matrix, which we call D. This reduces the memory requirements from $\mathcal{O}(N^2)$ to $\mathcal{O}(NB)$. We show with the results of the experiences that a reasonable value for B (about 1 000) is enough for achieving good solutions in reasonable time frames.

The first medoid, m_1, is selected as the point in the dataset whose average dissimilarity against the B references is minimal.

At this stage, all points belong to the cluster of m_1. Then, we can construct a matrix of gains $\mathcal{G}$ by computing element-wise gains G between the row associated to the medoid m_1, D_{m_1}, and every other row in D, as follows:

$$\mathcal{G}_{r,c} = max(0, d(x_r, m_1) - d(x_r, b_c)), r \in \{1, \ldots, N\}, c \in \{1, \ldots, B\} \tag{6}$$

Where x_r denotes the r-th point in the dataset and b_c the c-th point in the batch $\mathcal{B}$. By taking the row-averages of $\mathcal{G}$ we get the average gain of each candidate point and choose the one with the largest average gain. The way in which gains are computed allows us to store the matrix $\mathcal{G}$ in a sparse form, saving memory. Once the jth medoid has been added, updating the gain matrix requires only doing the above computation considering the row corresponding to m_j of $\mathcal{G}$, in this way the computed gain has incorporated the information of all the previous medoids. This is repeated until the k medoids are selected. Algorithm 3

shows the general idea of the algorithm. Keep in mind that it is not the pseudo-code of the actual algorithm, to avoid over-complicated implementation details.

Without getting into over-complicated implementation details, Algorithm 3 shows in a schematic manner how OBGM works.

Algorithm 3 General idea of the OBGM algorithm

Require: X ▷ Dataset of size N
Require: $1 \leq K \leq N, K \in \mathbb{Z}$ ▷ Number of medoids
Require: $1 \leq B \leq N, B \in \mathbb{Z}$ ▷ Random sample size
Require: Dissimilarity function d
1: $\mathcal{M} \leftarrow \{\}$ ▷ Initialize empty set of medoids
2: $\mathcal{B} \leftarrow$ random sample of size B from X ▷ Randomly sample the reference points
3: $\mathcal{G}^0 \leftarrow d(X, \mathcal{B})$ ▷ Initialize gain matrix
4: **for** $i = 1$ to K **do**
5: Select ith medoid m_i
6: $\mathcal{G}^i \leftarrow$ Update($\mathcal{G}^{i-1}$, m_i)
7: **end for**
8: **return** $\mathcal{M}$

5 Methodology

In this section we will discuss the methodology used for comparing the performances of the algorithms. In section 5.1 the datasets are presented. In section 5.2 we present the different configurations used for running the experiments, as well as the limitations imposed by hardware. While in section 5.3 we define the measure we have chosen to compare results, the Pareto front.

5.1 Datasets

Three datasets were selected for performing the tests: MNIST digits[LCB94], CIFAR-10 [Kri09] and HEPMASS [Whi16].

The MNIST digits dataset is a set of 28 by 28 pixel grey-scale images of handwritten digits from 0 to 9. The dataset consists of 60 000 training images and 10 000 testing images. There are an equal number of images of each digit.

CIFAR-10 dataset consists of 60 000 32 by 32 color images, representing 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6 000 images of each class.

Both MNIST and CIFAR-10 were treated as vectors in $\mathbb{R}^N$, with corresponding N,

Finally, the HEPMASS dataset is a dataset for binary classification (signal or background). Each observation consists of 27 normalized features. Instances are balanced both with regards the target variable and 5 possible particle masses.

These datasets were selected given that they are standard for the benchmarking of new algorithms in the machine learning literature.

5.2 Experiment settings

Experiments were performed on a home-grade laptop. The objective was to compute 1, 5, 10, 20, 50, 100, 200, 500 and 1 000 medoids on all datasets, with different algorithms and randomized subsamples.

For each individual configuration, five runs where made with a different random seed, to ensure reproducibility and fairness in comparing the results between algorithms. For the smaller datasets, the subsamples were taken only from the train-partition and with the following scheme: [1 000, 5 000, 10 000, 15 000, 20 000], given the limitation in available RAM, for the cases of RAM-intensive algorithms. All algorithms where used with its default configurations, with exception of BanditPAM whose *swap iter* parameter (number of swap iterations to perform)was set to 0, 2 and 5. For the OBGM, the batch size was also varied, taking values in 100, 300, 1 000 and 3 000.

For HEPMASS, it was decided to try to push the algorithms to their boundaries, and the experiments were run with subsamples of size 100 000, 500 000 and 1 000 000. For this, only kcenter, kmeanspp and OBGM were inluded.

Given the hardware and time limitations, a boundary of 700 seconds was set for the execution of the algorithms. If a given configuration exceeded this mark, it was flagged and subsequent, more complex (greater K) were also flagged to save execution time.

5.3 Soft Pareto front

The Pareto Front is a measure taken in multi-objective optimization problems. Given a set S of configurations and its respective set R of multiple-objective results, one configuration s_i is said to strictly dominate another configuration s_j if its results are strictly better than those of s_j in every optimized dimension. One configuration s_i belongs to the Pareto front if it is not strictly dominated by any other configuration in the set S.

As an example, take Figure 5 where the objective would be to maximize the production of Item 1 and Item 2 and a set of feasible solutions. There exists no unique solution that dominates on both axis. Computing the Pareto front allows the decision maker to focus the decision on a subset of all feasible solutions, instead of considering them all, and evaluating more carefully what the trade-off are between them.

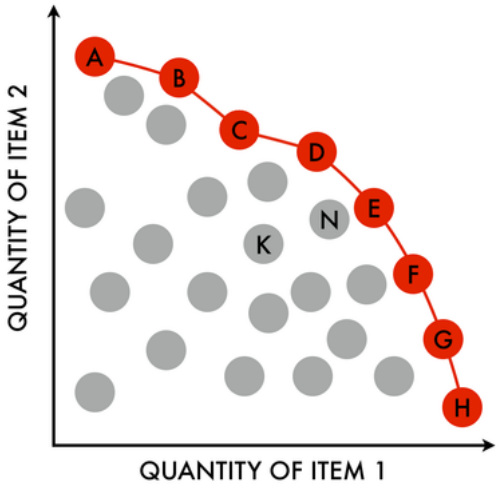


Figure 5: Example of the Pareto Frontier for a two dimensional maximization objective. The points in red are not Pareto dominated by any of the points in grey.

In the case of our experiments, we would like to minimize both the execution time and the objective function. We then count how many times and algorithm was part of the front.

It is important to highlight that certain configurations become meaningless, such as the estimation of the 1-medoid with a random-algorithm, given that it will always dominate in time. Also the cases where the subsample is of size 1 000 and we search for 1 000 clusters, given that each point in the subsample will become its own centroid and a perfect score.

As a matter of fact, we use a variation that we refer to as *soft Pareto front*. With this variation we relax the strict dominance to a margin of 1% on each axis, to avoid cases of a point being excluded from the front because it is partially worse than another one in one axis, despite being much better on the other.

6 Experimental results

In this section we present and discuss the results obtained after running all the experiences. Tables 1, 2 and 3 show the consolidated results for CIFAR, MNIST and HEPMASS datasets, respectively. With each dataset, for each unique pair (sample size, number of clusters) soft Pareto front were computed and the total number of appearances of each algorithm were counted.

6.1 CIFAR-10 dataset

With the CIFAR-10 dataset, OBGM variations are the models that appear most frequently in the Pareto front, as can be seen in Table 1. Next, *fasterpam*, with 17 appearances in the soft Pareto front, is the best non-OBGM algorithm. Its main weakness is the fact that it has big memory requirements, thus not being able to handle larger datasets, as OBGM can. Furthermore, if we focus on Figure 6 and 7 that show the average time and average objective function, with the exception of OBGM with a batch of 10 000 samples (onebatch-10k), OBGM computes the medoids between $4.3X$ and $21.9X$ faster for small values of k and between $5.2X$ and $1.25X$ faster for large values of k, at a cost of less than 1% in the objective function.

alternate and *kmeanspp* make 9 and 6 appearances, respectively, mainly as a result of fast execution times for small values of N. However, *alternate*'s execution time is highly dependent on N and as the dataset grows it is worse than OBGM, by as much as 95.8% execution time. On the other hand, *kmeanspp* weakness is the value of k: for small values of k its runtime is very competitive, yet in terms of the solution found it is consistently the second worse algorithm, with about 23% worse objectives obtained compared to our method. As k grows, the algorithm finds competitive solutions (up to less than 1% worse than OBGM) but it slows down significantly, taking as much as 139% more than the slowest OBGM (onebatch-10k) and 3 679% more time than the fastest (onebatch-300). The results of time and objective-function measurements of CIFAR for a subset of size 20 000 can be seen in Figures 6 and 7.

Finally, we highlight the appearance of *bandit-2*. This happens to be one of the degenerate cases, in which an algorithm is included in the front for a small gain in the objective function, despite being much worse in execution time, as can be seen in Figure 8.

Method	Times in front	% in front
kcenters	1	1.01%
kmeanspp	6	6.06%
onebatch-1k	53	53.54%
onebatch-300	78	78.79%
onebatch-3k	34	34.34%
fastermsc	1	1.01%
fasterpam	17	17.17%
alternate	9	9.09%
bandit-2	1	1.01%

Table 1: Counting of appearances of each model on the Pareto front for the CIFAR dataset.

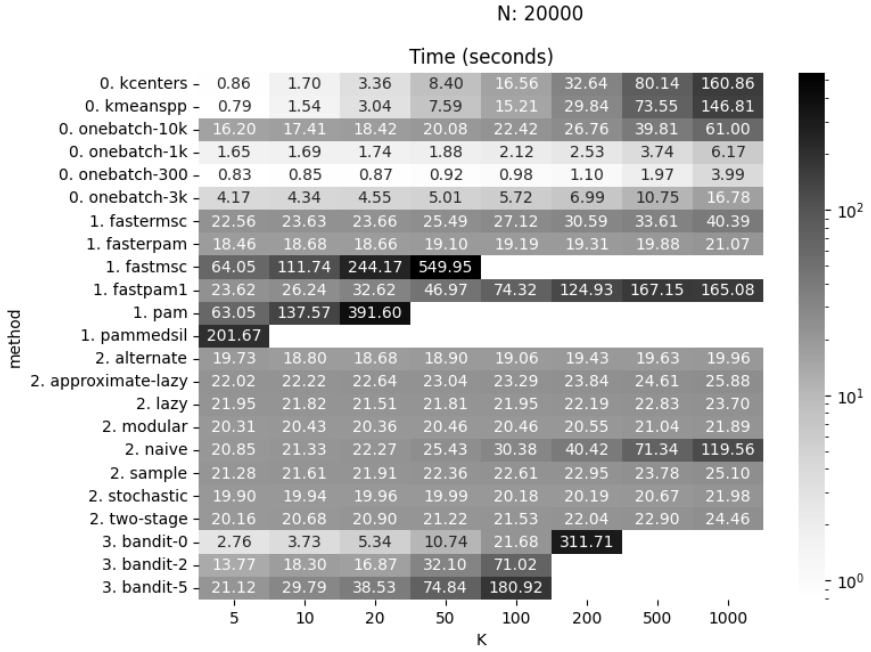


Figure 6: Average execution time value for algorithms on a subsample of size 20 000 of the CIFAR dataset with different values of k.

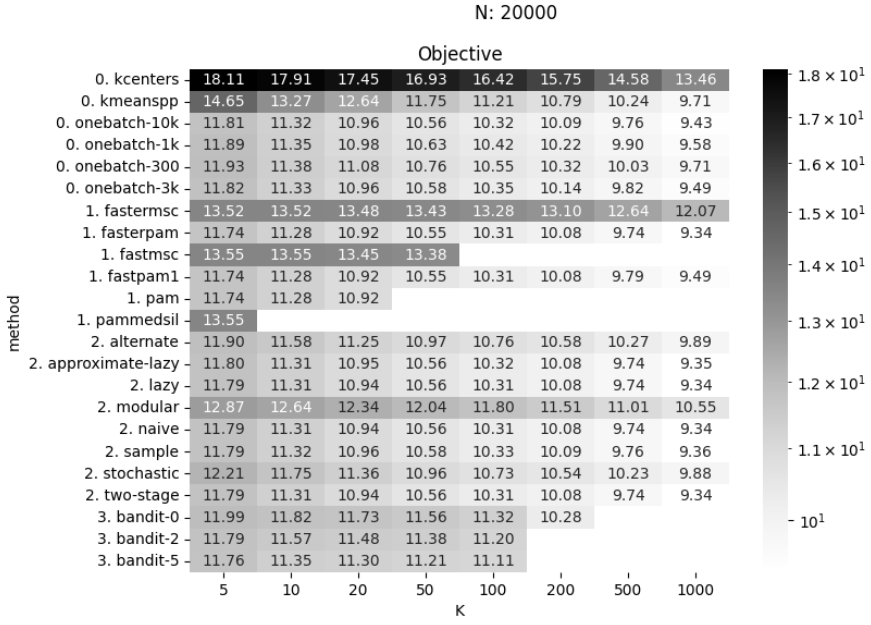


Figure 7: Average objective function value for algorithms on a subsample of size 20 000 of the CIFAR dataset with different values of k.

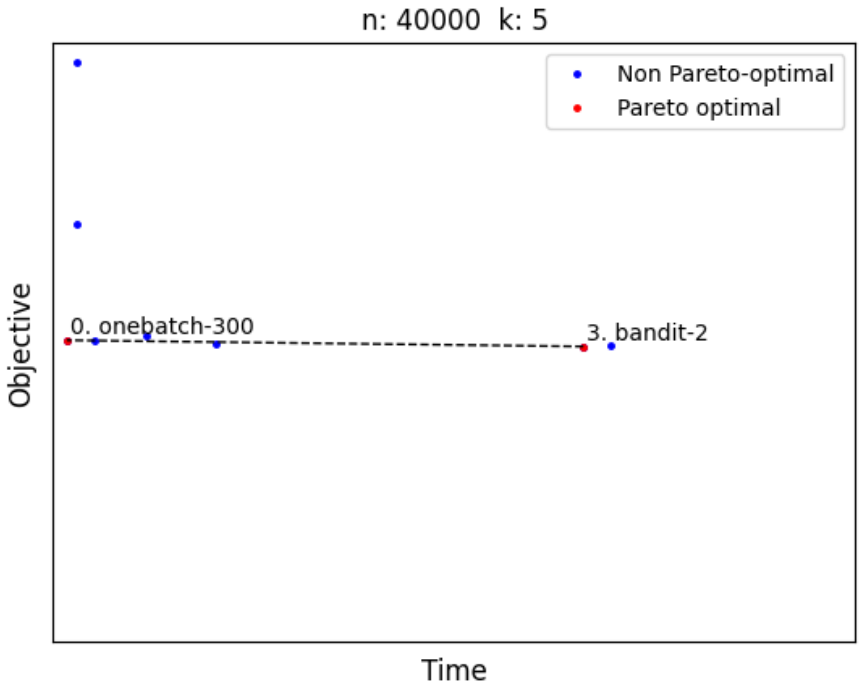


Figure 8: Degenerate Pareto Front for CIFAR dataset with N = 40 000 and K = 5. *Bandit-2* makes it to the Front, despite having a much greater execution time than other algorithms and not an enormous gain on the objective function.

6.2 MNIST dataset

When considering the results for MNIST (Table 2), the behaviour of the results is similar to that of CIFAR-10, and the analysis remains as in section 6.1. We see again that the proposed algorithm outperforms the current alternatives and it dominates the Pareto Front appearances.

Method	Times in front	% in front
kcenters	2	1.71%
kmeanspp	19	16.24%
onebatch-10k	23	19.66%
onebatch-1k	78	66.67%
onebatch-300	92	78.63%
onebatch-3k	56	47.86%
fastermsc	1	0.85%
fasterpam	28	23.93%
alternate	11	9.40%
bandit-2	3	2.56%

Table 2: Counting of appearances of each model on the Pareto front for the MNIST dataset.

6.3 HEPMASS dataset

Finally, Table 3 shows the results for HEPMASS dataset. This case is remarkable since it is one of the main motivations for this whole work: dealing with larger datasets. All algorithms from *apricot-select, k-medoids and BanditPAM packages* had to be left out because the memory or time constraints did not allow us to work with this number of instances. We can see once again the variants of OBGM algorithm outperforming kmeanspp and kcenters in the big picture analysis of the Pareto front. If we turn to images 9 and 10, we see objective and time results, respectively, for N = 100 000. *kcenters* remains competitive only thanks to its speed, but objective-wise it is not competitive at all. On the other hand, *kmeanspp*'s performance is again tighly related to the value of k. With small k it remains faster than OBGM by 1 or 2 orders of magnitude, but objective-wise the results are 10% to 15% worse (bigger objective function).

Method	Times in front	% in front
kcenters	2	5.56%
kmeanspp	25	69.44%
onebatch-10k	6	16.67%
onebatch-1k	27	75.00%
onebatch-300	28	77.78%
onebatch-3k	12	33.33%

Table 3: Counting of appearances of each model on the Pareto front for the HEPMASS dataset.

If we push the limits and take N to 1 000 000, results that can be seen in Figures 12 and 11, there many things to note. First of all, we have reached the time limit with OBGM and batch sizes of 3 000 and 10 000, therefore they do not appear in the figures. Both *kmeanspp* and *kcenters* are also challenged. For small values of k they remain faster than OBGM, yet with objective functions from 13% to 40% higher. As k inreases, so does the runtime and the objective function decreases. However, the runtime for OBGM does it at a slower rate than the others. At $k = 200$ for a batch size of 300 and $k = 500$ of 1 000, OBGM outpaces kcenters and kmeanpspp by as much as 122% in the first case and 60% in the second, margins that get bigger as k continues to grow. This advantage is acompanied by a clear objective-value trump of 35% with respect to kcenters and even results in the case of kmeanspp.

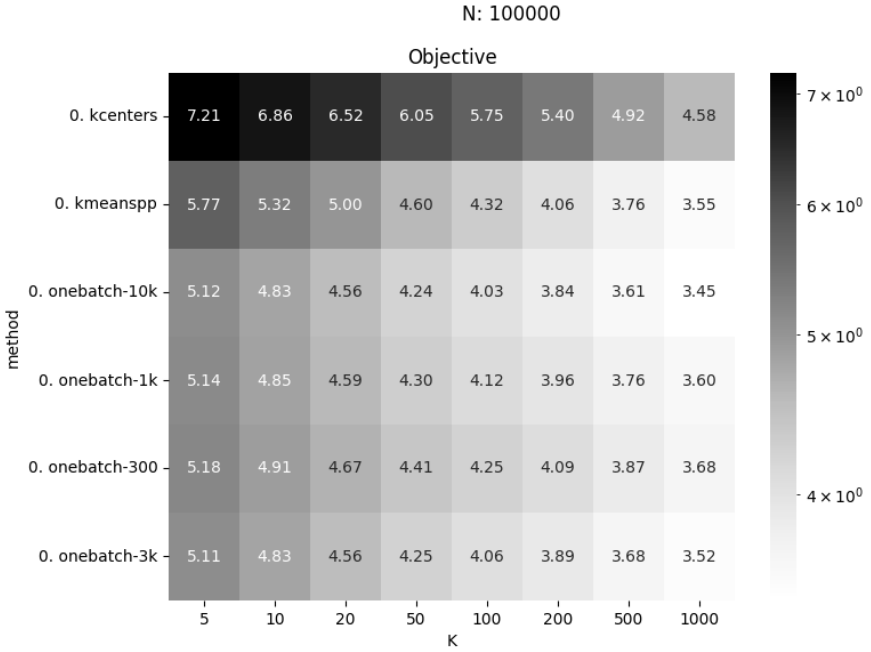


Figure 9: Average objective value for algorithms run on HEPMASS dataset on subsample of size 100 000.

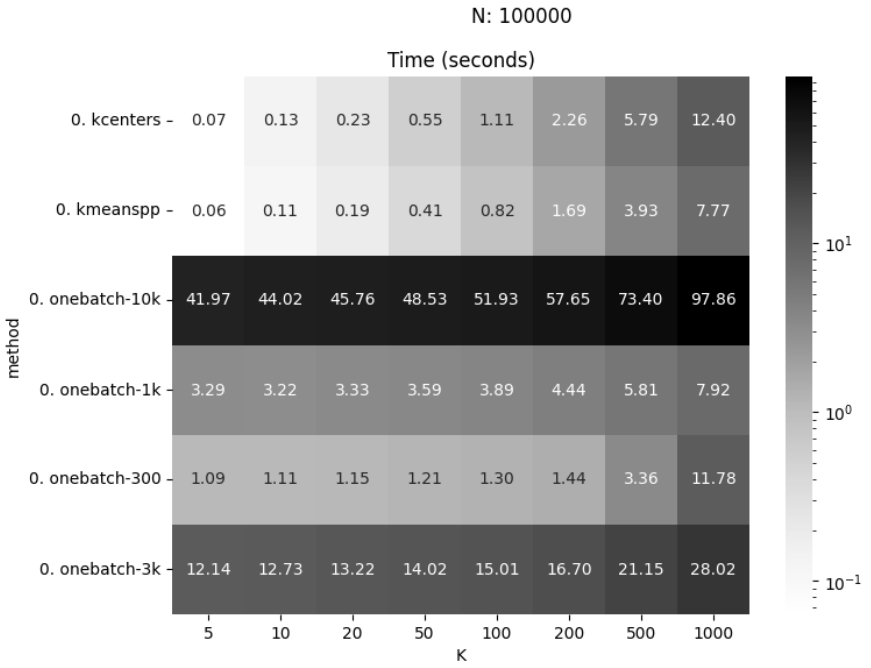


Figure 10: Average execution time for algorithms on a subsample of size 100 000 of the HEPMASS dataset with different values of k.

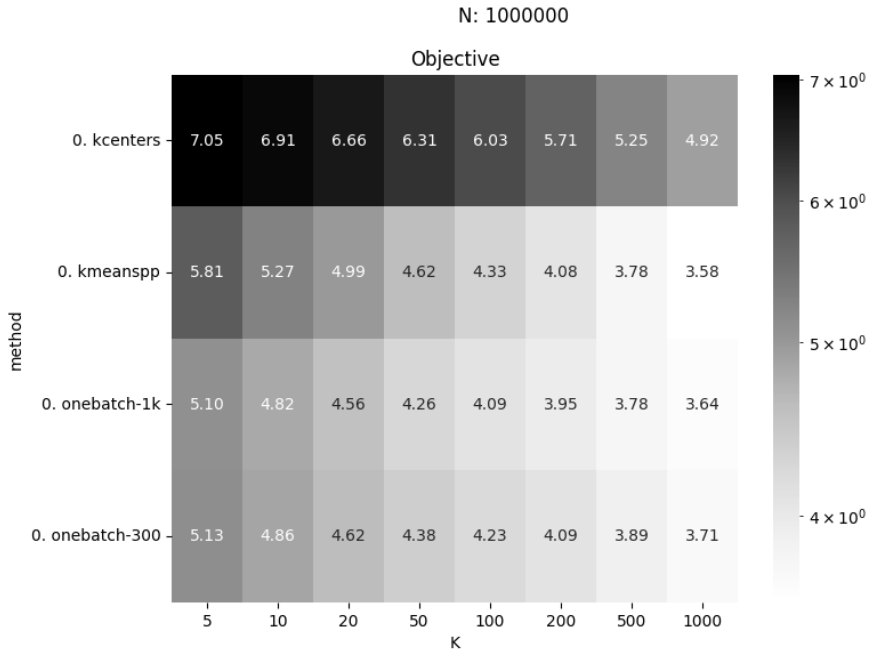


Figure 11: Average objective value for algorithms run on HEPMASS dataset on subsample of size 1 000 000.

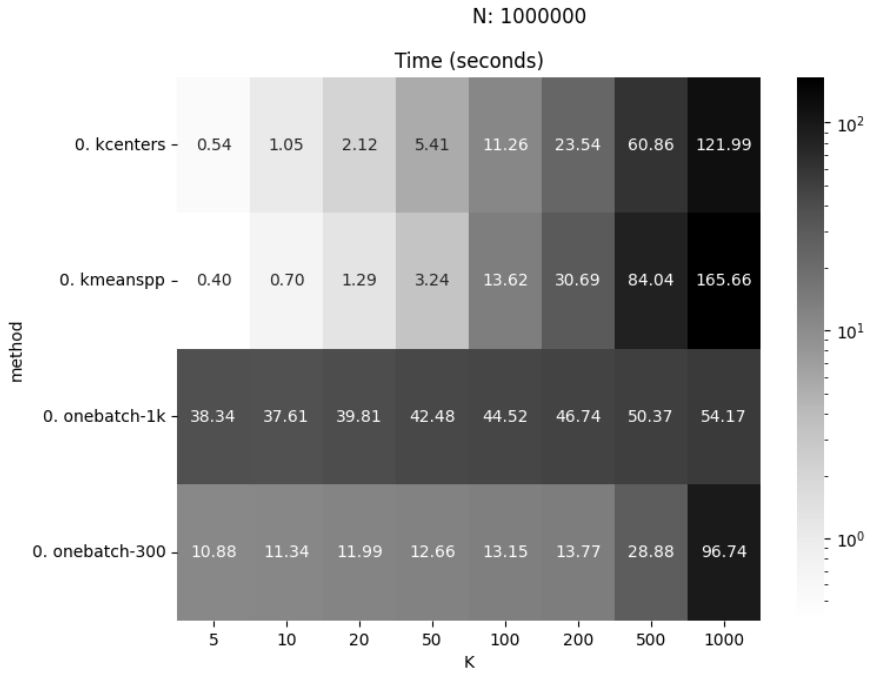


Figure 12: Average execution time for algorithms on a subsample of size 1 000 000 of the HEPMASS dataset with different values of k.

7 Conclusions

In this work we have made a broad literature review of methods for finding k-representative points among a finite set, both from a clustering and a facility location perspective. We studied the history of the problem, the proposed algorithms and its evolution. Then, we developed a new alternative that is able to overcome some of the common drawbacks from mainstream methods: speed and dataset size. Finally, we developed a bench-marking protocol, applied it to three different datasets, in many different settings, and compared the results.

After evaluating the results of the experiments we conclude that we have successfully conceived a high-performance and low-resource alternative to current state-of-the-art medoid clustering algorithms.

OBGM proved to be able to outperform those methods in many different settings and deal with datasets that would not have been able to be handled by most consumer-grade hardware.

This type of developments are of great importance for making clustering techniques available to institutions and individuals that do not count with sufficient economical resources to access more powerful hardware. This lowering of the barrier to entry should encourage people to adopt the medoids-clustering or further working on it to make it better.

We also would like to highlight that those who do posses the more powerful hardware also benefit from this innovation, given that they power and time consumption would be lowered without compromising the quality of the results.

8 Future work

With respect to OBGM itself, given the good performance the implementation of the algorithm directly in Python, there's definitely still room for performance gains by working in a C or C++ implementation with its corresponding binding to Python. Furthermore, our implementation is not optimized for servers with lots of memory, given that the computations are performed in a chunked manner, therefore being slowed down.

We also would like that this work serves as a motivation for other to continue with the development of stochastic alternatives to other hardware intensive algorithms, such that more professionals and enthusiasts alike can make use of them.

References

[AV07] David Arthur and Sergei Vassilvitskii. "k-means++: the advantages of careful seeding". In: *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 978-0-898716-24-5.

[ÄK06] Sami Äyrämö and Tommi Kärkkäinen. *Introduction to partitioning-based clustering methods with a robust example*. Tech. rep. University of Jyvaskyla, Oct. 2006. DOI: 10.13140/RG.2.2.32908.33927.

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, Jan. 2006. URL: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/.

[Boc07] Hans-Hermann Bock. "Clustering Methods: A History of k-Means Algorithms". In: *Selected Contributions in Data Analysis and Classification*. Ed. by Paula Brito et al. Red. by H. -H. Bock et al. Series Title: Studies in Classification, Data Analysis, and Knowledge Organization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 161–172. DOI: 10.1007/978-3-540-73560-1_15. URL: http://link.springer.com/10.1007/978-3-540-73560-1_15 (visited on 07/19/2023).

[DM15] Mark S Daskin and Kayse Lee Maass. "The p-median problem". In: *Location science*. Springer, 2015, pp. 21–45.

[Dre95] Zvi Drezner. "Dynamic facility location: The progressive p-median problem". In: *Location Science* 3.1 (1995), pp. 1–7.

[Erl78] Donald Erlenkotter. "A Dual-Based Procedure for Uncapacitated Facility Location". In: *Operations Research* 26.6 (Dec. 1978), pp. 992–1009. ISSN: 0030-364X, 1526-5463. DOI: 10.1287/opre.26.6.992. URL: https://pubsonline.informs.org/doi/10.1287/opre.26.6.992 (visited on 07/13/2023).

[Est02] Vladimir Estivill-Castro. "Why so many clustering algorithms: a position paper". In: *SIGKDD Explor.* 4 (2002), pp. 65–75.

[For65] E. W. Forgy. "Cluster analysis of multivariate data : efficiency versus interpretability of classifications". In: *Biometrics* 21 (1965), pp. 768–769. URL: https://api.semanticscholar.org/CorpusID:118110564.

[GMW07] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Mathematics. SIAM, 2007. ISBN: 978-0-89871-623-8. DOI: https://doi.org/10.1137/1.9780898718348.

[Hak64] S. L. Hakimi. "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph". In: *Operations Research* 12.3 (1964), pp. 450–459. URL: http://www.jstor.org/stable/168125.

[HFT17] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning*. Springer New York, NY, 2017. DOI: https://doi.org/10.1007/978-0-387-21606-5.

[JM18] Rahul Joshi and Preeti Mulay. "Mind Map based Survey of Conventional and Recent Clustering Algorithms: Learning's for Development of Parallel and Distributed Clustering Algorithms". In: *International Journal of Computer Applications* 181 (July 2018), pp. 14–21. DOI: 10.5120/ijca2018917487.

[KR90a] "Partitioning Around Medoids (Program PAM)". In: *Wiley Series in Probability and Statistics*. Ed. by Leonard Kaufman and Peter J. Rousseeuw. Hoboken, NJ, USA: John Wiley & Sons, Inc., Mar. 8, 1990, pp. 68–125. DOI: 10.1002/9780470316801.ch2. URL: https://onlinelibrary.wiley.com/doi/10.1002/9780470316801.ch2 (visited on 06/08/2023).

[KR90b] Leonard Kaufman and Peter J. Rousseeuw. "Partitioning Aroung Medoids (Program PAM)". In: Wiley Series in Probability and Statistics, 1990. Chap. 2, pp. 68–125. DOI: https://doi.org/10.1002/9780470316801.ch2.

[KAU87] KAUFMAN Leonard and ROUSSEEUW Peter. *Clustering by means of medoids*. 1987.

[KKS14] Noor Kamal Kaur, Usvir Kaur, and Dheerendra Singh. "K-Medoid clustering algorithm-a review". In: *Int. J. Comput. Appl. Technol* 1.1 (2014), pp. 42–45.

[Kri09] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: 2009.

[LPB03] Mark Van der Laan, Katherine Pollard, and Jennifer Bryan. "A new partitioning around medoids algorithm". In: *Journal of Statistical Computation and Simulation* 73.8 (2003), pp. 575–584. DOI: 10.1080/0094965031000136012. eprint: https://doi.org/10.1080/0094965031000136012. URL: https://doi.org/10.1080/0094965031000136012.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[LCB94] Yan LeCunn, Corinna Cortes, and Christopher J.C. Burges. *The MNIST databaseof handwritten digits*. 1994. URL: %5Curl%7Bhttp://yann.lecun.com/exdb/mnist/%7D.

[LS22] Lars Lenssen and Erich Schubert. "Clustering by Direct Optimization of the Medoid Silhouette". In: *Similarity Search and Applications*. Ed. by Tomáš Skopal et al. Cham: Springer International Publishing, 2022, pp. 190–204. ISBN: 978-3-031-17849-8.

[Llo82] Stuart P. Lloyd. "Least squares quantization in PCM". In: *IEEE Trans. Inf. Theory* 28 (1982), pp. 129–136. URL: https://api.semanticscholar.org/CorpusID:10833328.

[MPK83] DésiréL. Massart, Frank Plastria, and Leonard Kaufman. "Non-hierarchical clustering with masloc". In: *Pattern Recognition* 16.5 (1983), pp. 507–516. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(83)90055-9. URL: https://www.sciencedirect.com/science/article/pii/0031320383900559.

[Min78] Michel Minoux. "Accelerated greedy algorithms for maximizing submodular set functions". In: *Optimization Techniques*. Ed. by J. Stoer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 234–243. ISBN: 978-3-540-35890-9.

[Mir+14] Baharan Mirzasoleiman et al. *Lazier Than Lazy Greedy*. 2014. arXiv: 1409.7938 [cs.LG].

[Mur12] Kevin Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 9780262018029.

[NWF78] George Nemhauser, Laurence Wolsey, and M. Fisher. "An Analysis of Approximations for Maximizing Submodular Set Functions—I". In: *Mathematical Programming* 14 (Dec. 1978), pp. 265–294. DOI: 10.1007/BF01588971.

[OD98] Susan Hesse Owen and Mark S. Daskin. "Strategic facility location: A review". In: *European Journal of Operational Research* 111.3 (Dec. 1998), pp. 423–447. ISSN: 03772217. DOI: 10.1016/S0377-2217(98)00186-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221798001866 (visited on 07/19/2023).

[PJ09] Hae-Sang Park and Chi-Hyuck Jun. "A simple and fast algorithm for K-medoids clustering". In: *Expert systems with applications* 36.2 (2009), pp. 3336–3341.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Ree06] J. Reese. "Solution methods for thep-median problem: An annotated bibliography". In: *Networks* 48.3 (Oct. 2006), pp. 125–142. ISSN: 0028-3045, 1097-0037. DOI: 10.1002/net.20128. URL: https://onlinelibrary.wiley.com/doi/10.1002/net.20128 (visited on 06/04/2023).

[RS70] Charles S. ReVelle and Ralph W. Swain. "Central Facilities Location". In: *Geographical Analysis* 2.1 (1970), pp. 30–42. ISSN: 00167363. DOI: 10.1111/j.1538-4632.1970.tb00142.x. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.1970.tb00142.x (visited on 07/20/2023).

[RRR04] Alan P Reynolds, Graeme Richards, and Vic J Rayward-Smith. "The application of k-medoids and pam to the clustering of rules". In: *Intelligent Data Engineering and Automated Learning–IDEAL 2004: 5th International Conference, Exeter, UK. August 25-27, 2004. Proceedings 5*. Springer. 2004, pp. 173–178.

[Rou87] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (Nov. 1987), pp. 53–65. ISSN: 03770427. DOI: 10.1016/0377-0427(87)90125-7. URL: https://linkinghub.elsevier.com/retrieve/pii/0377042787901257 (visited on 07/18/2023).

[SBN20] Jacob Schreiber, Jeffrey Bilmes, and William Stafford Noble. "apricot: Submodular selection for data summarization in Python". In: *Journal of Machine Learning Research* 21.161 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/19-467.html.

[Sch21] Erich Schubert. 2021. URL: %5Curl%7Bhttps://pypi.org/project/kmedoids/%7D.

[SR19] Erich Schubert and Peter J. Rousseeuw. "Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms". In: vol. 11807. 2019, pp. 171–187. DOI: 10.1007/978-3-030-32047-8_16. arXiv: 1810.05691[cs,stat]. URL: http://arxiv.org/abs/1810.05691 (visited on 07/18/2023).

[SR21] Erich Schubert and Peter J. Rousseeuw. "Fast and Eager k-Medoids Clustering: O(k) Runtime Improvement of the PAM, CLARA, and CLARANS Algorithms". In: *Information Systems* 101 (Nov. 2021), p. 101804. ISSN: 03064379. DOI: 10.1016/j.is.2021.101804. arXiv: 2008.05171[cs,stat]. URL: http://arxiv.org/abs/2008.05171 (visited on 05/24/2023).

[Tan+05] Pang-Ning Tan et al. *Introduction to Data Mining*. 2005.

[TFL83] Barbaros C Tansel, Richard L Francis, and Timothy J Lowe. "State of the art—location on networks: a survey. Part I: the p-center and p-median problems". In: *Management science* 29.4 (1983), pp. 482–497.

[Tiw+20] Mo Tiwari et al. "Banditpam: Almost linear time k-medoids clustering via multi-armed bandits". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 10211–10222.

[Uni01] Eurostat Environment Statistics Unit. 2001.

[WIB14] Kai Wei, Rishabh Iyer, and Jeff Bilmes. "Fast Multi-Stage Submodular Maximization". In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, 2014, II–1494–II–1502.

[Whi16] Daniel Whiteson. *HEPMASS*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5PP5W. 2016.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[XT15] Dongkuan Xu and Yingjie Tian. "A Comprehensive Survey of Clustering Algorithms". In: *Annals of Data Science* 2 (2 2015). DOI: 10.1007/s40745-015-0040-1.

[XI05] R. Xu and D. Wunsch II. "Survey of Clustering Algorithms". In: *IEEE Transactions on Neural Networks*. Vol. 16. 3. 2005, pp. 645–78. DOI: https://doi.org/10.1109/TNN.2005.845141.

[YTM23] Roman Yurchak, Mathieu Timothee, and Felix Maximilian. *scikit-learn-extra*. 2023. URL: https://scikit-learn-extra.readthedocs.io/en/stable/index.html.

Glossary

A

API Application programming interface. Software interface between computer programs.

apricot-select Python package for submodular optimization in machine learning.

B

BanditPAM (package) Python package implementing the BanditPAM algorithm.

BanditPAM (algorithm) Stochastic algorithm for finding k-medoids in a set of data.

C

Centroid Representative point of a cluster.

Cluster Each of the groups formed by clustering.

Clustering Problem of grouping observations into groups with shared properties.

CNRS Centre National de la Recherche Scientifique (National Scientifique Research Center).

F

Facility location problem Problem of selecting an optimal subset of size k among a set of $k < K$ possible options, typically associated with the location of production facilities or warehouses.

H

HEPMASS dataset Dataset of High Energy Particles Mass.

K

k-medoids (package) Python package implementing different algorithms to solve the k-medoids problem.

k-medoids (algorithm) Partition clustering problem to find k clusters centered around medoids.

k-means Partition clustering problem to find k clusters centered around means.

M

Medoid High dimensional generalization of the median.

MNIST database Modified National Institute of Standards and Technology database.

O

OBGM One batch greedy medoids. Algorithm developed in this internship.

Operations research discipline that deals with the development and application of analytical methods to improve decision-making.

P

PAM Partitioning Around Medoids.

Pareto front Set of all Pareto efficient solutions in a multi-objective optimization problem.

PCA Principal component analysis.

S

Silhouette Quality measure in cluster analysis.

sklearn (scikit-learn) Python package for scientific computing.

A Sustainable development and social responsibility

As mentioned in Section 2.1, the Centre Borelli is affiliated to five institutions. This internship has been carried out in the École Normale Supérieure Paris-Saclay, reason for which we will be focusing only on this institution, and not all five of them.

Environmental pressures are pressures resulting from human activities which generate changes in the state of the environment [Uni01]. Located in the Paris-Saclay area, the construction of the building was regulated by the RT2012 and the specific environmental requirements of the Plateau de Saclay. Specific to the everyday activities of the Centre Borelli inside ENS Paris-Saclay, the environmental pressures are mainly originated by the transportation of researchers and administrative staff to the site at the Plateau de Saclay, together with an intensive use of computer equipment and processing servers.

A.1 Sustainable Development

A.1.1 Policies, strategy and communication

The ENS Paris-Saclay is signatory of the *Accord de Grenoble*, proposed during the Student-COP2 (2021), where adhering institutions engage to be further involved with socio-ecological challenges. The ENS Paris-Saclay has made 68 commitments under this agreement and the *Conseil d'administration* approved a number of measures, among which the most iconic ones are: https://ens-paris-saclay.fr/sites/default/files/SUPPORTS_COM/Rapport_activite_ENS-PS.pdf

- Train 100% of the students in matters of the ecological transition.
- Create one year of specific education on the socio-ecologial challenges.
- Augment the participation of research in the efforts to transition.
- Promote the research on topics beneficial to the transition.
- Establish a building energy and carbon balance.
- Set up a vegetable garden and composting system.

Following national and ministerial directive, the school has accelerated the energy sobriety plan for the end of 2022. The plan was approved by the *Conseil d'administration* on December 2022, with the objective of reducing energetic consumption while maintaining the high standards of all the activities carried out on the institution. Many points of actions where identified, including: buildings, mobility, purchases and activities of research and education.

With regards to the building, it incorporates a bio-climatic strategy for the comfort of people and taking care of the environment. It has built-in natural ventilation, double-flux ventilation, passive-cooling, an atrium that generates both heat and cool air (according to the seasons), green-roofs, solar panels, as well as recovery and management of rain water. Finally, the heating and cooling network is powered by low-temperature geothermal station, together with mixed (gas and electricity) heating. The school also deployed a network of sensors for real-time analysis of energy consumption that allows to take proactive action on energy consumption, mainly through automation. A policy of room temperatures has been set-up, so as to diminish energy waste on unoccupied spaces. For cold weather a limit of heating to 19°C when the rooms are in use. For hot weather, the use of AC is not activated for temperatures

under 26°C, except specific needs. As part of its engagement to diminishing its electricity consumption, the school takes the following actions: with regards to lighting, the building counts low-energy LED and presence sensors to optimize lighting according to occupation. Shared spaces are managed according to the timetable of the sites. Being intensive users of informatic equipment, the staff equipment is standardized and a life-span of 5 years if considered, which is the same as the guarantee, for maximising its use and reduce purchases. Servers and storage devices are kept for periods of 7 years, after which they are donated. With regards to the use of scientific equipment, the school engaged in an introspective investigation to determine the real needs and avoid waste. For this, the most energy-hungry equipment was identified, the scientific community made aware of the impact and the first balance of emissions should be published by December 2023. Since the metro line that connects the Plateau directly is not expected to open until 2026, the school took actions to facilitate an efficient mobility for the personnel. In particular, shuttles are at their dispossal from Porte d'Orléans, Gentilly and Cachan. Furthermore, a partnership was established with the carpooling company BlaBlaCar, also open for students. Also charging stations for electric vehicles and extensive bike-parking facilities are available. Regarding purchases, the first policy is that of trying to avoid unnecessary purchases by encouraging the re-utilisation and prolongation of life-span for equipment and material. If purchases are judged as needed, then public market offers are set up with strict audits relative to environmental terms accorded with the provider of the good or service. Finally, the communication plan with the objective of raising awareness of the stakes of the energy sobriety plan, includes a DDRS entry on the newsletters both for students and staff.

A.2 Social Responsibility

In this section, I present a summary of the social responsibility actions by the ENS Paris-Saclay.

A.2.1 Gender equality

In June 2021 the technical comitee and the governing body of the ENS Paris Saclay approved the "Plan égalité professionnelle de l'ENS Paris-Saclay" (*ENS Paris-Saclay profesional equality plan*) that proposes actions based on four axes. The proposals are based on findings and the definition of objectives, which are articulated around two major themes: train and inform, with strong actions of awareness-raising and information, as well as training staff and students. An evaluation of the current situation was performed for each of the four axes, and actions were planned for the coming years. Some of the actions include:

1. **Axis 1: Assessing, preventing and addressing pay gaps**
   (a) Inform the different actors of the ENS about women-men equality.
   (b) Make visible the equality plan and policies.
   (c) "Cafés de l'égalité", make periodic encounters to debate about equality.
2. **Axis 2: Ensuring equal access for women and men to corps, grades and jobs**
   (a) Raise awareness of gender bias to all members of comitees and jurys.
   (b) Highlight women working at the core of the ENS

(c) Inform all staff of the opening of campaigns for promotions and bonuses and especially encourage those who might qualify.

3. **Axis 3: The relationship between personal and working life**

   (a) Improve availability of maternity, paternity and parental leave information, by posting it on the intranet

   (b) Setting up and clear reporting of a breastfeeding room within the institution, within the medical-social pole.

   (c) Work on the opportunity of a shared action concerning a Wednesday extracurricular reception (reception set up by CentraleSupélec)

4. **Axis 4: The fight against sexual and gender-based violence, harassment and discrimination**

   (a) Maintain and develop a system to combat sexual harassment and violence

   (b) Maintain and develop events organized to inform students of the mechanism to combat gender and sexual violence and inform on these topics in parallel with the network of prevention of psychosocial risks

   (c) Formalize in a document available online the modalities of taking into account trans, non-binary, intersex identities. Update documents and forms to better reflect these identities.

A.2.2 Financial aids and social action

The web site of ENS provides a lot of information regarding social and financial help services, provided by the ENS itself, Université Paris Saclay, the government or other agents.

The human resources department is ready to receive staff, PhD students or undergraduate students from ENS Paris-Saclay, they all can appoint an interview with the head of social action group to talk about the difficulties being faced, access advice and being accompanied in the search of a solution.

A.2.3 Persons with disabilities

https://www.apa.org/pi/disability/resources/choosing-words

Inside the *health and well-being division* there is a team dedicated to persons with disabilities. This team is accessible undergraduate and graduate students, as well as faculty members and administrative staff.

ENS Paris-Saclay is engaged in promoting the reception and accompaniment of persons with disabilities. In July 2018 a plurianual plan was approved by the Governing Body.

Via the responsible of the *"mission handicap"*, ENS takes part in multiple projects, such as:

- **Tutorat Phratries**, together with association *Fédé*.
- **Handicap task-force Paris-Saclay**.

- **Handicap task-force of the *Conférence des Grandes Écoles*.**

B Operational research perspective

B.1 Facility location

Whether a retail chain siting a new outlet, a manufacturer choosing where to position a warehouse, or a city planner selecting locations for fire stations, strategic planners are often challenged by difficult spatial resource allocation decisions [OD98].

The problem of finding a point in the plane that minimizes the distance to three others can be traced back to at least the times of Fermat in the 17th century [Ree06]. In the 20th century, Alfred Weber added weights [Ree06] and the problem was later generalized to $n \geq 3$ weighted points and $p \geq 1$ medians on a continuous plane. Later, Hakimi [Hak64] studied the problems in networks and proved that, although not all optimal solutions lie on vertices, there is always a collection of p-vertices that is optimal. Therefore, one can find an optimal solution within a discrete representation of the problem.

Consider a connected, undirected network $N = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is the set of n vertices and E the set of edges. Let $w_i \geq 0$ be the weight of each vertex and d_{ij} the shortest distances between vertices i and j. The problem is formulated as an integer programming problem [RS70], lets define:

$$x_{ij} = \begin{cases} 1 & \text{if vertex } v_i \text{ if assigned to vertex } v_j \\ 0 & \text{otherwise} \end{cases}$$

Then the problem becomes:

$$\min z = \sum_{j=1}^{n} \sum_{i=1}^{n} w_i d_{ij} x_{ij} \tag{7}$$

$$\text{subject to} \begin{cases} \sum_j x_{ij} = 1, for i = 1, \ldots, n, & (8) \\ \sum_j x_{ij} = p, & (9) \\ x_{jj} \geq x_{ij} \forall i, j = 1, \ldots, n, & (10) \\ x_{ij} \in 0, 1 & (11) \end{cases}$$

Where the first constraint ensures that each node is assigned to exactly one center. The second ensures the number of centers to be p. The third prevents one vertex being assigned to a non-median vertex.

In the uncapacitated, or simple, facility location problem (UFLP), facilities of unrestricted size are placed among m possible sites with the objective of minimizing the total cost for satisfying fixed demands specified at n locations [Erl78]. In his review, Reese [Ree06] highlights the differences between the UFLP and the p-median problem studied by Hakimi: First,

UFLP involves a fixed cost for locating a facility at a given vertex, and the p-median problem does not. Second, unlike the p-median problem, UFLP does not have a constraint on the maximum number of facilities. Last, typical UFLP formulations separate the set of possible facilities from the set of demand points. In the p-median problem these sets are identical.

B.2 K-medoids as a facility location problem

Kaufman [KAU87] remarks that in the formulation used in clustering the sets of users and locations coincide, the location of a center is interpreted as the selection of an object as the representative object of a cluster and the distance travelled by a user corresponds to the dissimilarity between an object and the medoid of the cluster to which it belongs. Furthermore, unlike the UFLP, when clustering there a priori no need to assign weights to the samples.