PCA-guided search for K-means

Introduction to Probabilistic Graphical Models and Deep Generative Networks

Timothé Boulet
Thomas Lemercier
Théo Saulus
Master MVA
France

ABSTRACT

In this report, we present PCA-guided search [Xu et al. 2015], which aims to enhance the K-means clustering algorithm, one of the most widely employed techniques in data analysis. Despite its popularity, K-means often struggles with converging to a global solution due to its non-convex formulation and the propensity to get trapped in local minima. To tackle this challenge, Xu et al. [2015] proposed method finds a relevant initilization point, by leveraging the fact that the optimal global solution for a relaxed version of K-means clustering lies exactly within the PCA subspace. It is thus possible to start from there before running the actual K-means algorithm. Additionnally, we demonstrate the empirical relevance of this initialization method, by comparing it against modern heuristics and other initialization techniques.

1 INTRODUCTION

Data clustering, also referred to as unsupervised classification, is a crucial method in machine learning and data analysis, aimed at grouping objects into clusters based on their similarities. This approach finds applications in diverse fields, including exploratory pattern analysis, data mining, decision-making, machine learning, vector quantization, and compression scenarios. As previously mentioned, among the various clustering algorithms, the K-means algorithm stands out for its simplicity, efficiency, and parallelizability, rendering it widely employed for addressing practical problems, on parallel and distributed computing platforms.

However, the K-means algorithm, faces challenges in converging to a global solution due to the non-convex nature of its formulation. Local minima near the initialization can hinder its effectiveness, especially in high-dimensional data where there are exponentially many local solutions. To address these challenges, two main directions have been explored:

(1) Methods to escape local minima, often employing meta-heuristics such as simulated annealing, genetic algorithms
(2) Designing better initialization algorithms

[Xu et al. 2015] focus on the second approach, specifically exploring the relationship between K-means clustering and principal component analysis (PCA). They make use of recent theoretical analyses that revealed that the global solution to a relaxed version of K-means clustering lies in the PCA subspace [Ding and He 2004], and introduce their method named *PCA-guided search for K-means clustering*, aiming at a more effective exploration within the PCA subspace. Subsequently, by using the solution obtained in the PCA subspace, the original discrete space is considered.

By leveraging the computational efficiency of K-means clustering in low-dimensional spaces, the proposed approach aims to achieve better solutions with reduced computational complexity.

2 THE PCA AND K-MEANS RELATION

In a previous study [Ding and He 2004], the connection between Principal Component Analysis (PCA) and the K-means algorithm was underlined. This section discusses two key findings that laid the groundwork for the algorithm we will be covering. We will explore the relationship between PCA and K-means, examining how PCA's smooth solutions and K-means' group memberships interact. These findings form the foundation for the PCA-guided search algorithm, which we will delve into in terms of development and analysis in the subsequent sections.

2.1 Solution of K-means clustering lies in PCA-subspace

First, let's introduce common notations for both the PCA and K-means problems.

PCA consists in computing the k principal eigenvectors $U = (u_1, ..., u_k)$ of the covariance matrix $Cov(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\top$ of the data $X \in \mathbb{R}^{p\times n}$ where $\bar{x}$ denotes the mean of X. Then, the solution of the PCA problem is $Y = (y_1, ..., y_n)$, with

$$y_i = U^\top x_i. \tag{1}$$

For this solution, we have by construction that $Cov(Y)$ is diagonal, and we can define the whitened solution by defining $\tilde{Y} = (\tilde{y}_i, ..., \tilde{y}_n)$ with $\tilde{y}_i = (U\Sigma^{-\frac{1}{2}})^\top x_i$ where $\Sigma = diag(\lambda_1, ..., \lambda_k)$ and $(\lambda_1, ..., \lambda_k)$ are the eigenvalues associated to U.

K-means clustering aims at minimizing the function

$$J = \sum_{j=1}^{K}\sum_{x_i\in C_j} \|x_i - \mu_j\|^2 \tag{2}$$

where μ_j is the centroid of cluster C_j, and the norm is Euclidean. The solution for the hard clustering problem is a binary indicator matrix $H = (h_1, ..., h_K) \in \mathbb{R}^{n\times K}$, where

$$H_{ij} = \begin{cases} \frac{1}{\sqrt{\#C_j}} & \text{if } x_i \text{belongs to cluster } C_j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\#C_j$ denotes the cardinal of cluster C_j. The relaxed clustering problem aims at minimizing J, without the constraint of having a binary H, and allowing attribution to clusters in the form of $\tilde{H} \in (-1, 1)$, due to rotations we introduce right bellow.

Theorem 2.1. *Using previous notations, one have* $\tilde{H}^* = \tilde{Y}^\top R^\top$, *where* $\tilde{H}^*$ *is the optimal relaxed indicator matrix, and* $R \in \mathbb{R}^{K\times K}$ *is an orthogonal rotation matrix.*

A proof of this theorem is available in annex : Proof 7.1.1

Definition 2.2. The optimal centroids $M = (\mu_1, \ldots, \mu_K)$ for K-means span a subspace called *cluster centroid subspace*. Conversly, the principle directions of PCA $U = (u_1, \ldots, u_k)$ span the *PCA-subspace*.

Theorem 2.3. *Cluster centroid subspace and PCA subspace are identical.*

A proof is available in annex : Proof 7.1.2

Therefore, the optimal solution of the relaxed K-means clustering problem lies in the k-dimensional PCA subspace, which is smaller than the original space. Using such values for the initialization should allow the non-relaxed K-means to converge more quickly to a global optimum, with less probability to be stuck in a local optium located far away from a global one.

2.2 PCA-guided search for K-means

Based on the theoretical properties of PCA and K-means that were just highlighted, the PCA-Guided Kmeans algorithm is detailed in Algorithm 1.

Algorithm 1 PCA-Guided K-Means

Inputs.

(1) $D_n = (X_i)_{i\in\{1,\cdots,n\}} \in \mathbb{R}^d$, dataset of points
(2) $k \leq n$, number of clusters

Dimension reduction. Apply PCA reduction to the data, setting the dimension of the reduction to k:

$$\tilde{D}_n = \text{PCA}(D_n)$$

Clustering on $\tilde{D}_n$. Apply K-means to the dimensionally reduced dataset, to obtain a binary indicator matrix $H \in \mathbb{R}^{n\times k}$ such that $H_{i,j} = 1$ if point i is assigned to cluster j and 0 otherwise:

$$H = \text{K-Means}(\tilde{D}_n, \text{R2(Random Initialization)})$$

Computing the initial centroids in the original space. Using the binary indicator matrix H, compute the centroids $(\mu_1, \cdots, \mu_k)$ in the original subspace:

$$\mu_j = \frac{\sum_{i=1}^n H_{i,j} X_i}{\sum_{i=1}^n H_{i,j}}$$

Clustering on D_n. Using the previously obtained centroids $(\mu_1, \cdots, \mu_k)$ as the initial centroids, perform the K-Means algorithm:

$$H = \text{K-Means}(D_n, (\mu_1, \cdots, \mu_k))$$

3 OTHER CLUSTERING ALGORITHMS

To evaluate the proposed method, Xu et al. [2015] conducted a comparison with various other K-Means initialization techniques. The methods under comparison are Random Initialization, KMeans++, KKZ algorithm, KR algorithm, Hierarchical Agglomerative Clustering (HAC), and PCA-part that we summarize and explain in this section.

3.1 Random Initialization

Two random initialization methods, R1 and R2, are distinguished in the study. Method R1 randomly partitions the data set into k clusters, with these initial clusters determining the starting centroids [Anderberg 1973; Forgy 1965]. Method R2, alternatively, involves selecting k random points from the data set as initial centroids.

The paper hypothesizes that R2 might be more effective because R1 tends to place every initial centroid near the data set's center, which could hinder the convergence process. However, existing studies [J.M. Pena 1999] indicate that R1 may outperform R2. To assess the validity of this hypothesis, both methods are included in the benchmark analysis.

3.2 K-Means++

K-means++ adopts a greedy approach in selecting K centroids [Arthur and Vassilvitskii 2007]. The first centroid is chosen randomly, and subsequent centroids are selected with a probability proportional to the square of the shortest distance from a data point to the closest centroid.

3.3 KKZ Initialization

The KKZ algorithm, named after the initials of its creators, is a hard deterministic version of KMeans++ Initialization. It initiates by selecting the first centroid with the maximum norm [Katsavounidis et al. 1994]. Successive centroids are chosen based on the maximal distance from previously selected centroids to candidate points, iteratively repeating this process until K centroids are identified.

3.4 KR Initialization

The KR algorithm, named after the initials of its creators, sequentially chooses centroids, with the first centroid selected as the most centrally located in the data set [Kaufman and Rousseeuw 1990]. Subsequent centroids are then chosen to be far away from the previously selected ones while still having many data points close to them. As the algorithm was not clearly identified, we opted to exclude it from our experiments.

3.5 HAC

Several studies use the results of hierarchical agglomerative clustering (HAC) as centroid initialization [Fraley 1998; Meila and Heckerman 2015; Redmond and Heneghan 2007]. HAC, employing a "bottom-up" approach, begins with numerous small clusters and iteratively merges the closest clusters until a single cluster is formed. The algorithm named HAC in this report is simply HAC with the additional Kmeans algorithm applied after it for convergence.

3.6 PCA-part

The PCA-part method involves recursively partitioning a current cluster into two by computing the first principal component [Boley 1998; Su and Dy 2007]. The sign of each element is used to split the cluster, and this process is repeated until K clusters are obtained. Due to time constraint, we chose to exclude this algorithm from our experiments.

3.7 Algorithms included in our benchmark

These varied initialization strategies provide several approaches to address the clustering problem. However for our experiments, we decided to focus on the following algorithm: R1 and R2 initialization, KKZ, HAC, Kmeans++, and PCA-guided search.

4 EXPERIMENTAL SETUP

4.1 Datasets description

We performed experiments on six different datasets selected from the classical clustering or classification datasets found in the literature. In an effort to replicate the findings of the original paper, we incorporated the same four data sets used for their benchmark, and we added the IRIS data set and the CIFAR10 data set.

The first five data sets are all image classification data sets, the last (IRIS) being a tabular data set. The class of an image represents it's cluster, which allows us to specify in advance the number of clusters k for our K-means algorithm, e.g. 10 clusters for MNIST.

The datasets are the following :

- **AT&T** : The *AT&T Face* dataset comprises ten images for 40 different individuals ($k = 40$), each with a size of 92×112 pixels and 256 grey levels per pixel. In our study, we consider pixels as features. The images were resized to 23×28, transformed into vectors of dimension 644 by concatenating the matrix rows.
- **MNIST** : The *MNIST Handwritten Digits* dataset includes 8-bit grayscale images representing digits "0" through "9", ($k = 10$) with around 7000 examples for each digit. We flattened our images to dimension 784.
- **Binary Alphabet** : The *Binary Alphabet* dataset consists of 26 handwritten alphabets "A" to "Z", ($k = 26$) with 39 examples for each letter. Each image is a 20×16 binary image, converted into vectors of dimension 320.
- **Coil20** : We used 360 images of 5 items ($k = 5$), from the *Coil20* dataset, which consist of grayscale images of objects. Each image is 32×32, and we convert them into vectors of dimension 1024.
- **CIFAR10** : The *CIFAR-10* dataset is composed of $60,000$ color images of size 32×32 in 10 different classes ($k = 10$), with $6,000$ images per class. Each image belongs to one of the following classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck. In our analysis, we treat each pixel's RGB values as features. We convert the images into vectors of dimension $32 \times 32 \times 3$ (representing the three color channels).
- **IRIS** : The *Iris* dataset consists of 150 samples of iris flowers, each belonging to one of three species: setosa, versicolor, or virginica ($k = 3$). This is a very simple clustering dataset, often used for classification tasks. Each sample has four features: sepal length, sepal width, petal length, and petal width, all measured in centimeters.

4.2 Synthetic datasets

In order to come with comprehensive robustness analysis, we have built synthetic datasets designed to test various scenarios:

- **Gaussians in large dimensions:** we examine Gaussian distributions with diagonal covariance matrix in $d = 500$-dimensional spaces, a setting where K-means make the correct assumption in terms of clusters independance and gaussianity, but needs to manage the curse of dimensionality. One of the datasets contains $k = 10$ clusters (denoted **Gauss 1**), while another contains 1000 (denoted **Gauss 2**) to test the robustness in the number of clusters, both containing $n = 20,000$ points. The standard deviation is randomly chosen for each cluster, between $\sigma_m(d) = \frac{2}{\sqrt{d - d/10}}$ and $\sigma_M(d) = \frac{2}{\sqrt{d + d/10}}$.
- **Ellipsoidal clusters:** we test the resistance to ellipsoidal clusters with Gaussian distributions by generating data from a multivariate normal distribution with a non-diagonal covariance matrix. First, we generate a diagonal matrix Σ containing standard deviations for each cluster, between $\sigma_m(d)$ and $\sigma_M(d)$, obtained like in the Gaussians datasets. Then it is randomly rotated by sampling a rotation matrix R, such that $\tilde{\Sigma} = R\Sigma R^\top$. This dataset contains 1000 points and 10 clusters, with only 30 features because sampling a random matrix is computationally expensive.
- **Heavy tailed distribution:** to test resilience against heavy-tailed distributions, we introduce a dataset generated by Laplace distribution, containing 1000 points with 250 features and 10 clusters. The scale parameter is randomly chosen for each cluster, between $\sigma_m(d)$ and $\sigma_M(d)$. It will be denoted **Laplace**.
- **Short tailed distribution:** conversely, datasets following a uniform distribution within a bounded ball represent scenarios with short-tailed distributions, testing the algorithm's sensitivity to more homogeneously spread data. The ball radius is randomly chosen for each cluster, between $\sigma_m(d)$ and $\sigma_M(d)$. This dataset has the same other characteristics as the Laplace one. It will be denoted **Uniform**.
- **Nested clusters:** to explore more intricate structures, we built a dataset where some clusters are generated within other clusters, to test the algorithm's ability to discern layered groupings. The points are sampled from a Gaussian distribution which standard deviations are randomly chosen for each cluster, between $\sigma_m(d)$ and $\sigma_M(d)$. This dataset has the same other characteristics as the Laplace one.
- **Bridging dataset:** we also consider a tricky dataset where two distinct clusters are connected by a linear sequences of points, to simulate transitional zones between two clusters. Thus, only $k = 3$ groups are here to be found: the two main clusters, and the bridge. The points from the two main clusters are sampled from a Gaussian distribution which standard deviations σ_1 and σ_2 are randomly chosen for each, between $\sigma_m(d)$ and $\sigma_M(d)$. The bridge points are created by linearly interpolating between the means of the two main clusters, with a standard deviation equal to $\min(\sigma_1, \sigma_2)/2$, to keep the bridge narrow. This dataset has the same other characteristics as the Laplace one.

Collectively, these synthetic tests are designed to understand more finely the strengths and limitations of clustering algorithms under diverse and controlled conditions. A summary of all datasets tested is available in Table 1.

4.3 Metrics

Like in the original paper, we use distortion to measure the performance of our algorithms, and we propose additionnal classical metrics for clustering: silhouette score, Davies-Bouldin index, and Calinski-Harabasz index:

- **distortion** is the sum of the L_2 distance between a data point and the centroid of its cluster assigned by the clustering algorithm:

$$M_D = \sum_{j=1}^{K} S_j \text{ with } S_j = \sum x_i \in C_j \|x_i - \mu_j\|^2.$$

  In each run, we execute multiple iterations of the algorithm and record the best distortion achieved since the start of the run. This metric is referred to as "best distortion" or "best distortion over time":

$$M_{D\text{best}}(t) = \max_{0 \le s \le t} M_D(s)$$

  with $M_D(s)$ being the distortion obtained for the s-th run. M_D ranges from 0 to ∞, the lower the better.
  This metric signifies the practical performance of our algorithms and mirrors the outcomes that the algorithms would manifest in real-world scenarios. The original paper overlooked this metric, as it solely showcased the distortion in decreasing order, failing to accurately depict the results one would encounter when deploying those algorithms in practical applications.
- **Silhouette score** measures how similar each point is to its own cluster compared to other clusters [Rousseeuw 1987]:

$$M_S = \frac{1}{K} \sum_{j=1}^{K} \frac{1}{\#C_j} \sum_{x_i \in C_j} \frac{b - a}{\max a, b},$$

  where $\#C_j$ denotes the cardinal of cluster C_j, a is the average distance of the point to its cluster, and b is the average distance of the point to its closest neighboring cluster:

$$a = \frac{1}{\#C_j - 1} \sum_{\substack{x_k \in C_j \\ x_k \neq x_i}} \|x_i - x_k\|^2, \quad b = \min_{l \neq j} \frac{1}{\#C_l} \sum_{x_k \in C_l} \|x_i - x_k\|^2$$

  M_S ranges from -1 to 1, the higher the better as it indicates that most points are clearly belonging to a cluster far away from neighboring clusters.
- **Davies-Bouldin index** measures the average similarity between each cluster and its most similar one [Davies and Bouldin 1979]:

$$M_{DB} = \frac{1}{K} \sum_{j=1}^{K} \max_{j \neq l} \frac{\frac{1}{\#C_j} S_j + \frac{1}{\#C_l} S_l}{\|\mu_j - \mu_l\|^2}$$

  M_{DB} ranges from 0 to ∞, the lower the better, indicating better separation between clusters.
- **Calinski-Harabasz index** is the ratio of the sum of between-clusters dispersion and the sum of within-cluster dispersion for all clusters [Caliński and Harabasz 1974]:

$$M_{CH} = \frac{(n - K) \sum_{j=1}^{K} \#C_j \cdot \|\mu_j - \mu_{global}\|^2}{(K - 1) \sum_{j=1}^{K} S_j}$$

  where μ_{global} is the centroid of all points in the dataset. M_{CH} ranges from 0 to ∞, the higher the better, indicating dense and separated clusters. The values are highly dependent on the size of the dataset, increasing with it.

Due to the heavy computational load associated with computing many distances in the silhouette score, Davies-Bouldin index, and Calinski-Harabasz index, only the distortion metric is used for the largest datasets MNIST and CIFAR-10.

4.4 Experimental settings

For every algorithm and dataset, we conducted a certain number n_{runs} of runs of the algorithm on the dataset. This number has been computed such that the time spend for each dataset is equivalent.

For *stochastic algorithms* (Random (R1 and R2), Kmeans++ and PCA Guided Search), we ran 10 runs with different seeds to measure the empirical mean and empirical standard deviation over our metrics. We plot the mean and the standard deviation, represented by the shaded envelope around the mean. For *deterministic algorithms* (KKZ and HAC), only one run was performed and the result is expanded to each iterations. The algorithm presented in Algorithm 2 provides a pseudocode representation of the experiments conducted.

All algorithms and metrics were re-implemented by ourselves in Python using *numpy* for minimal library dependencies. An alternative version is sometimes proposed with *scikit-learn*, when available. The code, along with an explicative README document, can be found publicly in our GitHub repository.

5 RESULTS

5.1 Performance

In this initial section, our objective was to validate the efficacy of the proposed algorithm empirically. To achieve this, we adhered to the experimental settings outlined earlier, wherein we will graphically represent the algorithm's performance in relation to iterations ($iter$) as well as time (t).

Regarding performance, we will exclusively chart the best metric attained after a specified number of iterations, or after a certain duration. This approach enables us to simulate the outcomes one would obtain by executing the various algorithms on a specific dataset, and be more practical than the view proposed by the original paper in which the metric is sorted before plotting.

We will present the results on a per-dataset basis. If an algorithm is not depicted, there could be two reasons:

(1) If the algorithm is absent from the plot's legend, it indicates that the algorithm was too slow to be effectively compared with others in terms of the plot as a function of time.
(2) If the algorithm is included in the legend but not visible on the actual plot, it signifies that the performance was subpar and did not meet the criteria for inclusion in the plot.

5.1.1 IRIS. The results can be found in Figure 1. The outcomes on the IRIS dataset are misleading because the clustering task is overly simplistic on this simple dataset, leading to nearly identical performances from all algorithms in nearly every iterations.

5.1.2 Binary Alphabet. The results are available in Figure 2. In this dataset, the algorithm with the highest performance is the one proposed. However, given the relatively small dimensionality of the dataset and, consequently, the still fast execution time of the HAC algorithm, it's important to highlight that in the early stages of approximately the first 200 iterations, the HAC algorithm outperformed the proposed one. This timeframe corresponds to roughly the initial five seconds.

5.1.3 AT&T. The results are available in Figure 3. For the AT&T dataset, it is evident that HAC emerges as the most effective clustering algorithm. This observation aligns with the findings from the Binary Alphabet dataset, reinforcing the notion that for datasets with relatively small dimensionality, HAC proves to be particularly effective. Following HAC, we observe KMeans++ and then the proposed PCA-Guided algorithm in terms of effectiveness.

5.1.4 Coil20. The results are available in Figure 4. The results from this dataset stand out, as HAC exhibits lower performance compared to its performance on other datasets. In contrast, the other initializations, except for KKZ, rapidly achieved a similar level of performance with only a few iterations. This suggests that this specific dataset may not be well-suited for hierarchical clustering.

5.1.5 MNIST. The results are available in Figure 5. Remarkably, despite the dataset featuring a substantial number of samples and high dimensionality, the results suggest a relatively straightforward clustering task, as all algorithms, except for kkz, converge to the same distortion value. However, it's worth noting that this is the first dataset where an algorithm is absent from the plot, indicating that HAC significantly lagged behind all the other algorithms in terms of execution time.

5.1.6 Cifar10. The results are available in Figure 6. On Cifar, for the limited iterations possible due to the dataset's substantial size and dimensionality, KKZ emerges as the best-performing algorithm, despite ranking the lowest in all other datasets. However, judging from the observed trend in the initial iterations, it appears that given additional time, every other algorithm could have achieved similar, if not better, performance. Remarkably, the second-best initialization is R2, surpassing PCA-Guided and KMeans++ in terms of both iteration and time-dependent performance.

5.1.7 Conclusion. From this limited set of datasets, it is evident that clustering is a highly challenging task, and no single algorithm surpasses others in all aspects. The proposed algorithm does not outperform existing algorithms but presents a new alternative that achieves a comparable level of performance.

Furthermore, it is crucial to emphasize that contrary to the authors' claim, the R2 initialization is not consistently superior to the R1 initialization, as indicated by the results from the Binary Alphabet dataset.

5.2 Time comparaison

While clustering algorithms are known to be highly sensitive to dataset characteristics, it is evident in our case that the top-performing algorithms include PCA-Guided Kmeans, Kmeans++, and HAC. To provide a more comprehensive understanding of these algorithms, we have compiled the average runtime for a single run of each on every dataset. The mean execution times for all the algorithms across all datasets are summarized in Table 2.

When examining this table, it's important to emphasize that it is only suitable for comparing execution times across algorithms, not datasets. This limitation arises from differences in data set dimensions and the fact that the computations were carried out on distinct computers (still the same computer for one dataset).

From a time-to-performance standpoint, PCA-Guided Search emerges as the winner as the dimensionality of the data increases. Nevertheless, for the datasets tested, the execution times of the various algorithms are still within the same order of magnitude, except for the HAC algorithm, which experiences a significant increase as the data dimensionality grows.

5.3 Initilization Sensibility

This section aims to enhance our understanding of the robustness of the various initializations. Specifically, we will examine the quartile coefficient of dispersion of the distortion across all the runs for each algorithm and dataset. The summarized results can be found in Table 3.

Surprisingly, PCA-Guided Kmeans does not appear to significantly enhance the stability and robustness of the Kmeans algorithm, even for high-dimensional data. At this point, we are unable to fully explain this result, except to suggest that the Kmeans++ algorithm may already be close to the optimal solution for the datasets tested.

5.4 Other metrics

The investigation of both the silhouette score and Calinski-Harabasz index yields the same conclusion as the distortion. Consequently, the corresponding plots will not be included in this report. However, the Davies-Bouldin index provides different insights and introduces some nuance to the performance evaluation.

The identical plots to those used for the distortion analysis can be found in the appendix (7, 8, 9, 10).

5.4.1 Iris. Regarding this metric, the Iris dataset remains too simple, and all algorithms converge to the same optimum.

5.4.2 Binary Alphabet and AT&T. For both of these datasets, the best-performing algorithm changed, with R2 becoming the top performer for Binary Alphabet and PCA-Guided Kmeans emerging as the best for the AT&T dataset.

5.4.3 Coil20. In the case of this last dataset, where all algorithms performed equally for the distortion metric, we observe that PCA-Guided Kmeans is underperforming, ranking second to last.

5.5 Synthetic datasets

5.5.1 Gauss1. The results are available in Figure 11. For this simple model, PCA-Guided Kmeans does not exibit any advantages over

others advanced initialization schemes such as Kmeans++ that almost always converges towards the same minimum of distortion.

5.5.2 Gauss2. The results are available in Figure 12. For this task, it must be noted that PCA-Guided is not able to run, due to the fact that $d < k$. Additionally, increasing the number of clusters significantly diminishes the potential of stochastic algorithms, other than Kmeans++, to achieve minimal distortion.

5.5.3 Ellipsoid. The results are available in Figure 13. In this setting, KKZ shines for its stability, and best result in terms of Davies-Bouldin among all. Interestingly, the variance of R1 is significantly lower than any other stochastic algorithm, all others having very large dispersion. On the other hand, PCA-Guided is slower than its counterparts.

5.5.4 Laplace. The results are available in Figure 14. From a pure performance standpoint, none of the algorithms seems to stand out in the case of a heavy-tail distribution and non-negligible outliers. Once again, Kmeans++ demonstrates its stability by achieving the smallest quantile coefficient of dispersion among all the stochastic algorithms (see Table 4).

5.5.5 Uniform. The results are available in Figure 15. While the performance appears to be comparable for all the stochastic algorithms, except for Kmeans++ which stands out once again, the plot as a function of time highlights that PCA-guided Kmeans is notably slower compared to other methods on this particular dataset.

5.5.6 Nested. The results are available in Figure 16. For this dataset, all methods converge quickly towards the same scores, which seem to indicate that they all find good solutions, and thus that K-means is robust to nested clusters. However, KMeans++ is notably quicker in doing so, and PCA-Guided the slowest of stochastic algorithms.

5.5.7 Bridging. The results are available in Figure 17. For this dataset, all methods converge quickly towards similar scores, suggesting that they all find good solutions, except for PCA-guided Kmeans, which converges towards a worse solution in terms of distortion.

5.5.8 Conclusion. With the synthetic datasets, where the clustering tasks are relatively straightforward most of the time due to well-defined clusters, each tested algorithm can achieve the best overall distortion. However, Kmeans++ demonstrates its robustness by not only being the fastest algorithm to converge but also the least sensitive, as evidenced by Table 4. On the other hand, PCA-Guided Kmeans exhibits some clear limitations when clusters are nested or intersect with each other.

5.6 Comparison with the original paper

Finally, the most surprising conclusion we draw is the disparity between our findings and the results reported in the original paper: PCA-GuidedSearch does not consistently outperform the Kmeans++ algorithm across all datasets. On most real datasets we tested, it performs comparably to Kmeans++, and it only surpasses Kmeans++ on the Binary Alphabet dataset. Even more surprising: on synthetic datasets targeting specific robustness tasks, PCA-GuidedSearch is often the least reliable in terms of speed and disparity.

6 CONCLUSION

In conclusion, we have investigated the PCA-guided search algorithm for K-means clustering, exploring both theoretical aspects and on a benchmark against the typical concurrent methods. The algorithm is very promising from a theoretical point of view, the results of the benchmark are overall positive. Notably, the implementation we built based on the scikit-learn package in Python has demonstrated that the algorithm is very time efficient. However, we must point out the limits of K-means itself, which suffers from strong assumptions such as spherical cluster shapes, equal cluster sizes, and the idea that the mean is the best central tendency measure for all clusters. These hypothesis may not hold true in all datasets, which affects the performances of the algorithm. PCA-guided search for K-means opens up interesting possibilities for research, particularly in investigating whether similar dimensionality reduction techniques could be beneficially applied to other classic clustering methods, with similar theoretical insights.

Notably, we propose two experiments that we did not have the time to run and gain insight about:

- Analyze more closely the effect of the dimension of reduction in term of performance and execution time, which has always been set to the number of clusters, as proposed by the original paper.
- Instead of using classical Kmeans on the PCA subspace, we suggest trying the Kmeans++ algorithm. Its overall robustness could potentially enhance the robustness of PCA-Guided Kmeans and lead to improved performance.
- Continue the experiments with more datasets, notably non-computer vision datasets.

REFERENCES

M.R. Anderberg. 1973. *Cluster Analysis for Applications.*

David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms.*

Daniel Boley. 1998. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery* 2 (1998), 325–344.

T. Caliński and J Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics* 3, 1 (1974), 1–27. https://doi.org/10.1080/03610927408827101

David L. Davies and Donald W. Bouldin. 1979. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1, 2 (1979), 224–227. https://doi.org/10.1109/TPAMI.1979.4766909

Chris Ding and Xiaofeng He. 2004. K-Means Clustering via Principal Component Analysis. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04).* Association for Computing Machinery, New York, NY, USA, 29. https://doi.org/10.1145/1015330.1015408

E. W. Forgy. 1965. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics* 21 (1965), 768–769.

Chris Fraley. 1998. Algorithms for Model-Based Gaussian Hierarchical Clustering. *SIAM J. Sci. Comput.* 20 (1998), 270–281.

J.A. Lozano P. Larranaga J.M. Pena, 1. 1999. An empirical comparison of four initialization methods for the K-Means algorithm. *Pattern Recognition Letters.* 20 (1999), 1027–1040.

Ioannis Katsavounidis, C.-C. Jay Kuo, and Zhen Zhang. 1994. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters* 1 (1994), 144–146.

Leonard Kaufman and Peter J. Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley.

Marina Meila and David Heckerman. 2015. An Experimental Comparison of Several Clustering and Initialization Methods. arXiv:1301.7401 [cs.LG]

Stephen J. Redmond and Conor Heneghan. 2007. A method for initialising the K-means clustering algorithm using kd-trees. *Pattern Recognition Letters* 28, 8 (2007), 965–973. https://doi.org/10.1016/j.patrec.2007.01.001

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Ting Su and Jennifer G. Dy. 2007. In search of deterministic methods for initializing K-means and Gaussian mixture clustering. *Intell. Data Anal.* 11 (2007), 319–338.

Qin Xu, Chris Ding, Jinpei Liu, and Bin Luo. 2015. PCA-guided search for K-means. *Pattern Recognition Letters* 54 (2015), 50–55.

7 APPENDIX

7.1 Proofs of the theorems

7.1.1 *Proof of Theorem 2.1.*

Proof. By definition, the centroid of each cluster is given by $\mu_j = \frac{1}{n_j} \sum_{x_i \in C_j} x_i = \frac{1}{\sqrt{n_j}} X h_j$. We can thus rewrite the objective function (also know as the distortion J):

$$\begin{aligned}
J &= \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \\
&= \sum_{j=1}^{K} \sum_{x_i \in C_j} \left(\|x_i\|^2 - 2\mu_j^\top + \|\mu_j\|^2 \right) \\
&= \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i\|^2 - \sum_{j=1}^{K} \mu_j^\top \sum_{x_i \in C_j} \left(2x_i - \mu_j \right) \\
&= \sum_{i=1}^{n} \|x_i\|^2 - \sum_{j=1}^{K} n_j \mu_j^\top \mu_j \\
&= \sum_{i=1}^{n} \|x_i\|^2 - \underbrace{\sum_{j=1}^{K} h_j^\top X^\top X h_j}_{tr(H^\top X^\top X H)}
\end{aligned}$$

Thus, $\min J = \max tr(H^\top X^\top X H) = \max tr(\tilde{H}^\top X^\top X \tilde{H})$, where $\tilde{H} = HR$, and R is any orthogonal rotation matrix. Besides, we have $H^\top H = I = \tilde{H}^\top \tilde{H}$ due to the orthogonality of clustering membership. Now, the optimization problem has become:

$$\max_{\tilde{H}} \quad tr(\tilde{H}^\top X^\top X \tilde{H}) \text{ such that } \tilde{H}^\top \tilde{H} = I \tag{4}$$

which is a classic discrete optimization problem known to be NP-hard. The optimal solution $\tilde{H}^*$ of the relaxed version (obtained by allowing H to have values in $(0, 1)$) is the K eigenvectors $V = (v_1, \ldots, v_K)$ of the Gram matrix $X^\top X$ associated with the K largest eigenvalues $(\xi_1, \ldots, \xi_K)$: $\tilde{H}^* = V$, i.e., $H^* = VR^T$. Finally, we can make use of the singular value decomposition (SVD) of X:

$$X = \sum_{i=1}^{r} \sigma_i u_i v_i^\top = U \Sigma V^\top, \tag{5}$$

with r the rank of X, to obtain that $\tilde{Y} = (U\Sigma^{-1})^\top X = V^\top = \tilde{H^*}^\top = R^\top \tilde{H^*}^\top$, and that $\sigma_k = \sqrt{\lambda_k} = \sqrt{\xi_k}$. This proves that $\tilde{H}^* = \tilde{Y}^\top R$ □

7.1.2 *Proof of Theorem 2.3.*

Proof. Using previous notations, we start by constructing an orthogonal basis of the cluster centroid subspace $\tilde{M} = M(M^\top M)^{-1/2}$. Therefore, the projection of any data point x_i on the cluster centroid subspace is given by $\tilde{M}\tilde{M}^\top x_i$. Now, we recall that the centroid of each cluster is written $\mu_j = \frac{1}{\sqrt{n_j}} X h_j$, hence $M = XHN^{-1/2}$ with $N = diag(n_1, \ldots, n_K)$. It follows that:

$$\begin{aligned}
\tilde{M}\tilde{M}^\top &= M(M^\top M)^{-1} M^\top \\
&= XHN^{-1/2}(N^{-1/2} H^\top X^\top X H N^{-1/2})^{-1} (XHN^{-1/2})^\top \\
&= XH(H^\top X^\top X H)^{-1} (XH)^\top \\
&= X\tilde{H}(\tilde{H}^\top X^\top X \tilde{H})^{-1} (X\tilde{H})^\top.
\end{aligned}$$

Thanks to Theorem 2.1, and the SVD of X, we obtain that the solution of K-means clustering is $HR = V$, i.e., $\tilde{H} = V$. Thus we have

$$\tilde{M}\tilde{M}^\top = XV(V^\top X^\top X V)^{-1} (XV)^\top. \tag{6}$$

From (5), we obtain that: $Xv_i = \sigma_i u_i$, i.e., $XV = U\Sigma$. Hence

$$\tilde{M}\tilde{M}^T = U\Sigma(\Sigma^\top U^\top U \Sigma)^{-1} (U\Sigma)^\top = UU^\top, \tag{7}$$

which is equivalent to saying that the cluster centroid subspace is equal to the PCA subspace. □

7.2 Experimental setting details

Experiments were run on our personal laptops. To have relevant metrics in term of algorithm execution time, all analysis related to one dataset have been performed on one computer only. Thus, execution time comparison across datasets should not be considered.

Dataset	n	d	k
MNIST	70000	784	10
AT&T	400	644	40
Binary Alphabet	1014	320	26
Coil20	360	1024	5
IRIS	150	4	3
CIFAR10	60000	3072	10
Gauss1	10000	500	10
Gauss2	10000	500	1000
Ellipsoid	1000	30	10
Laplace	1000	200	10
Unif	1000	200	10
Nested	1000	200	10
Bridging	1000	200	3

Table 1: Summary of synthetic and real-world datasets used in the study.

Algorithm 2 Experimental setting

Inputs. $D_i, i \in \{1, \cdots, n\}$, a set of datasets
Inputs. $A_j, j \in \{1, \cdots, m\}$ a set of algorithms

for $i \in \{1, \cdots, n\}$ **do**
  for $j \in \{1, \cdots, m\}$ **do**
    for $k \in \{1, \cdots, n_{\text{runs}}\}$ **do**
      for iter $\in \{1, \cdots, l\}$ **do**
        Run Algorithm (A_j) on dataset (D_i)
        Compute and save the metrics
      end for
    end for
  end for
end for

7.3 Distortion analysis for real datasets

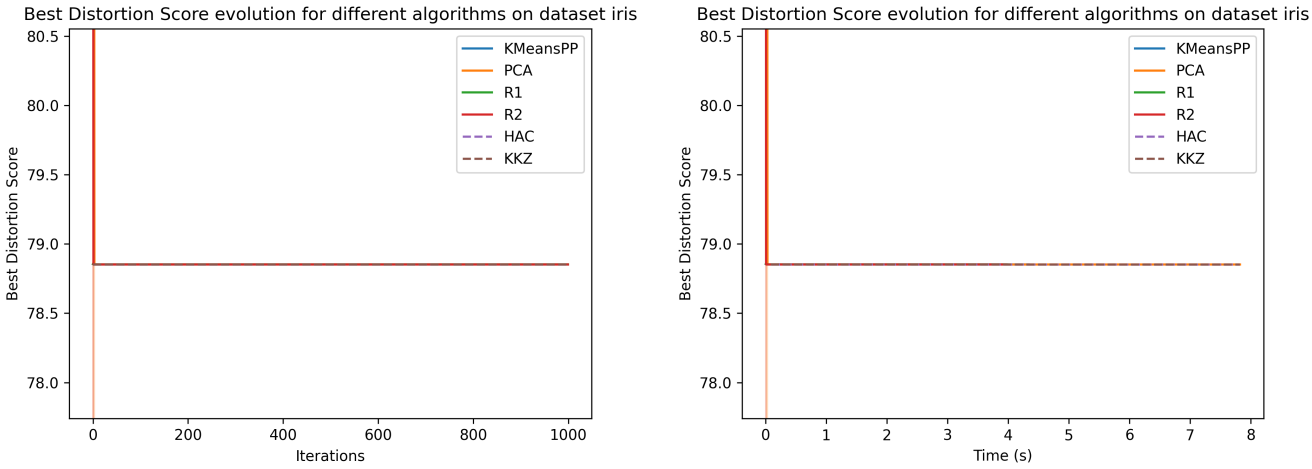

Figure 1: Results on IRIS, as a function of the number of iterations (right) and of the runtime (left)

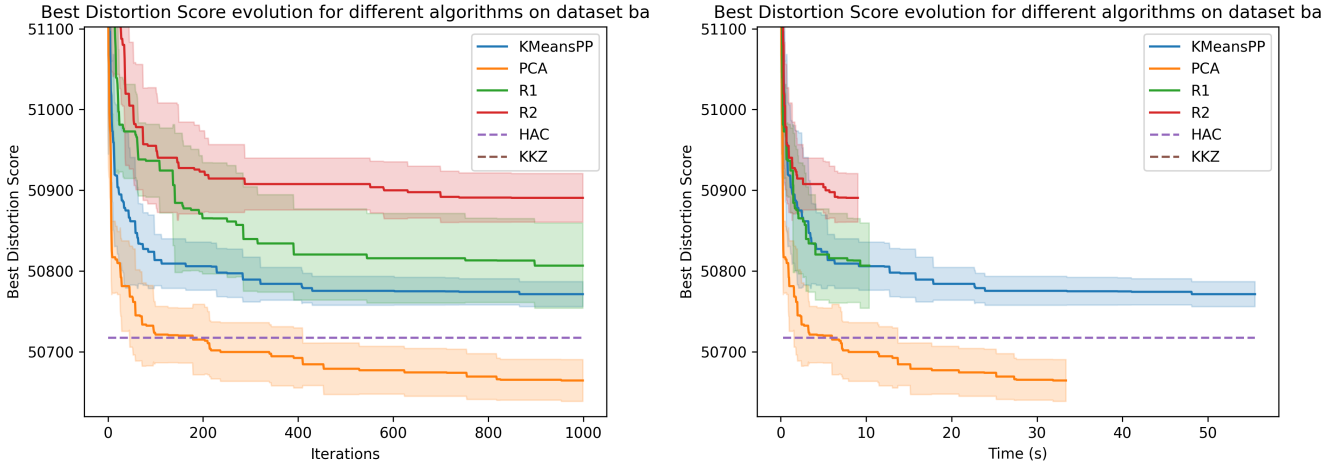

Figure 2: Results on Binary Alphabet, as a function of the number of iterations (right) and of the runtime (left)

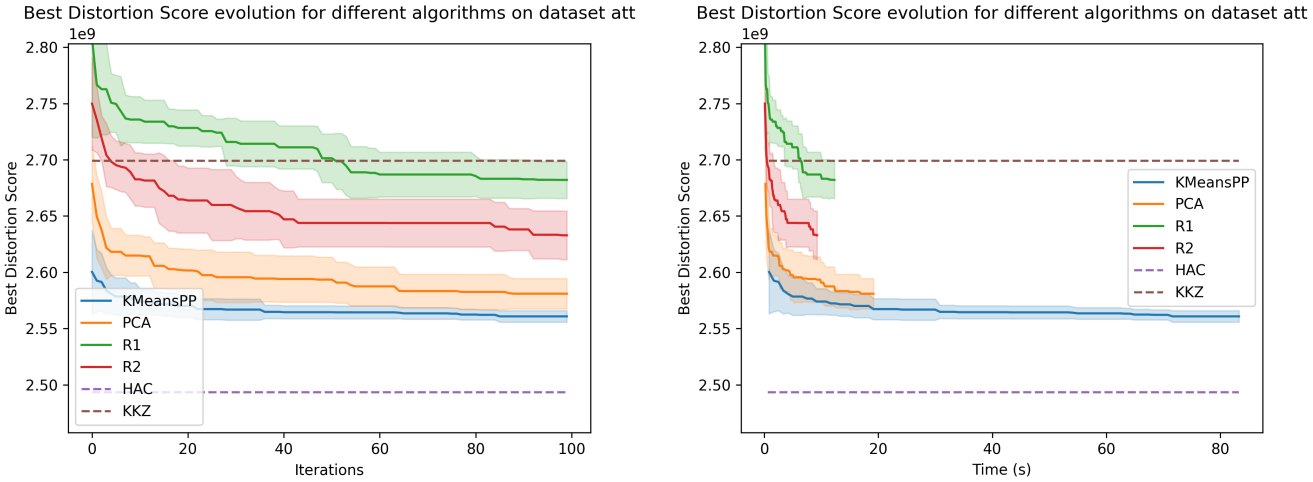

Figure 3: Results on the AT&T, as a function of the number of iterations (right) and of the runtime (left)

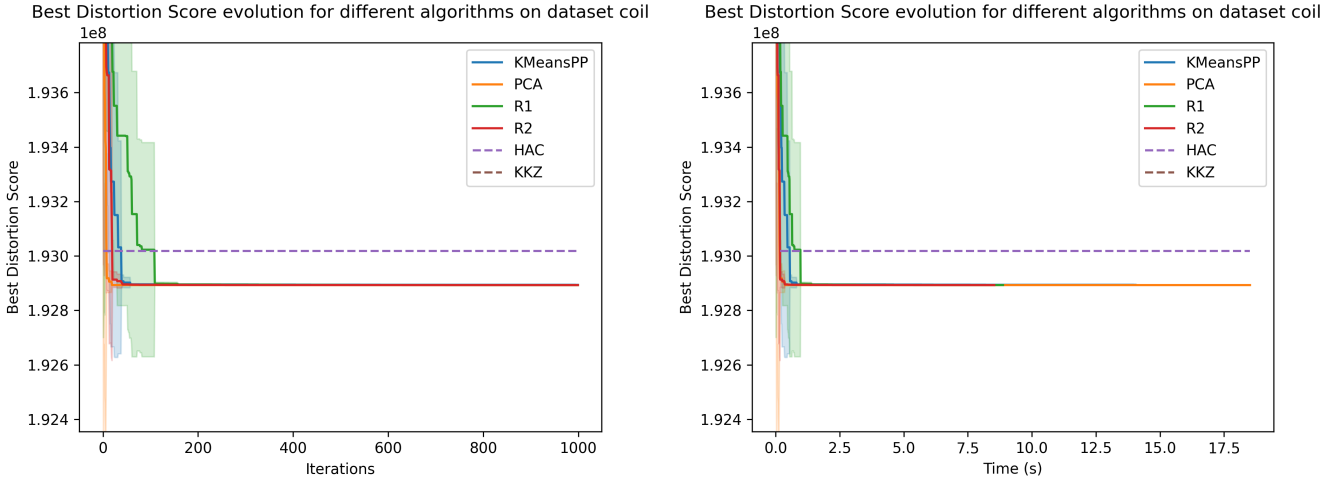

Figure 4: Results on Coil20, as a function of the number of iterations (right) and of the runtime (left)

7.4 Execution time and dispersion tables for real datasets

Table 2: Execution Time (s)

Algorithm	Iris	B.A.	AT&T	Coil20	MNIST	Cifar10
HAC	0.11	0.25	0.63	0.17	1666.25	15053.75
KKZ	0.088	0.20	0.80	0.15	3.09	11.09
KMeans++	0.0043	0.055	0.83	0.014	2.92	9.91
PCA-GuidedSearch	0.0078	0.033	0.19	0.018	2.39	**4.28**
R1	**0.0040**	0.010	0.12	0.0089	1.93	5.45
R2	**0.0040**	**0.0090**	**0.093**	**0.0085**	**1.838**	5.20

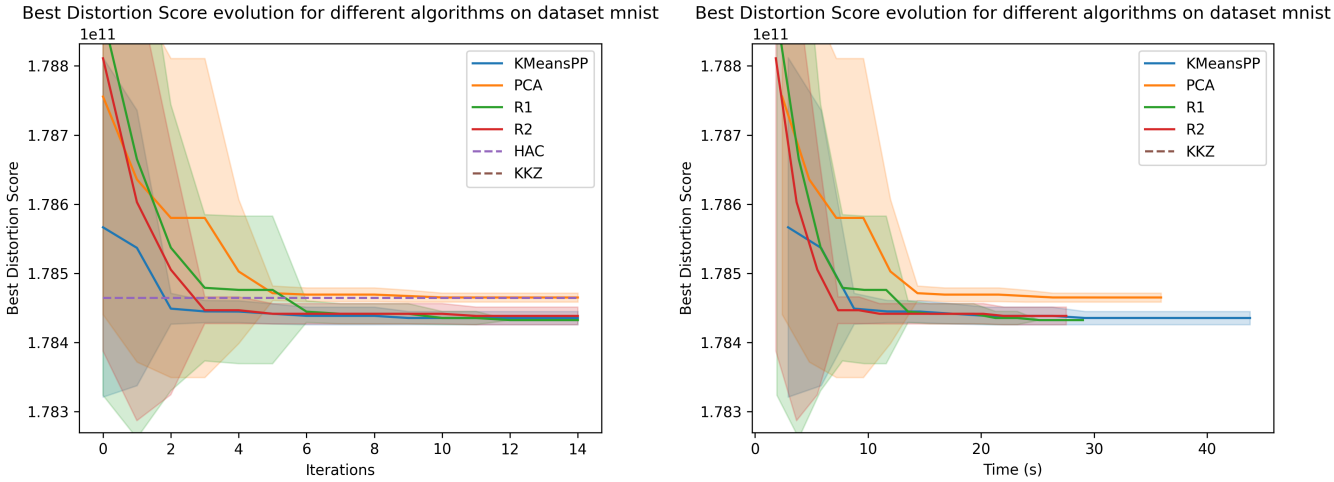

Figure 5: Results on MNIST, as a function of the number of iterations (right) and of the runtime (left)

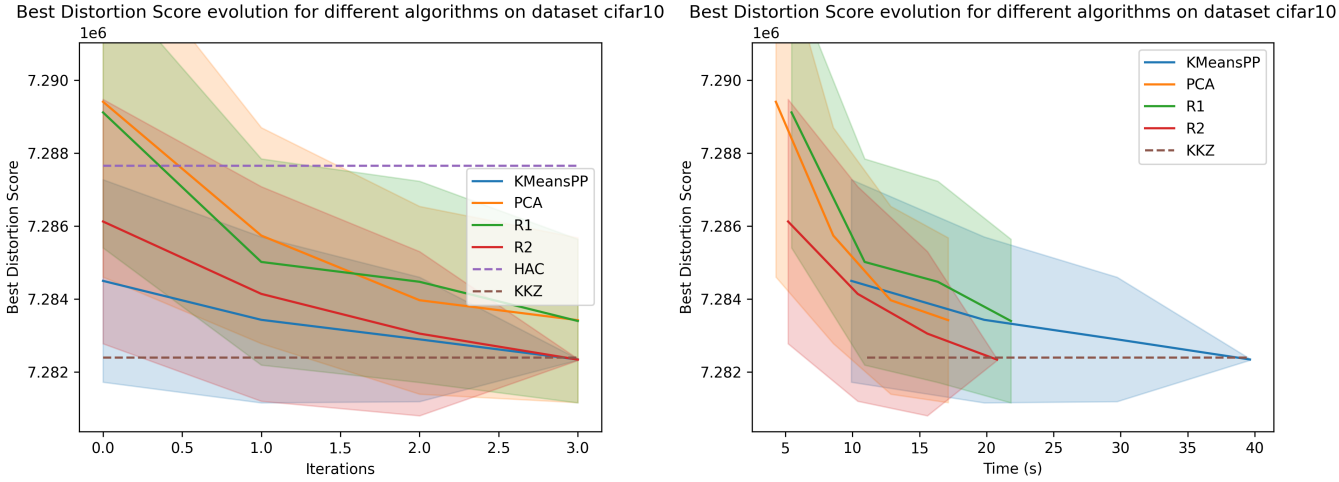

Figure 6: Results on CIFAR10, as a function of the number of iterations (right) and of the runtime (left)

Table 3: Quartile Coefficient of Dispersion

Algorithm	Iris	B.A.	AT&T	Coil20	MNIST	Cifar10
HAC	0	0	0	0	0	0
KKZ	0	0	0	0	0	0
KMeansPP	0.000027	**0.0027**	**0.0075**	**0.0064**	0.0016	**0.00037**
PCA-GuidedSearch	0.000027	**0.0027**	0.011	0.012	**0.0015**	0.00040
R1	**0**	0.0033	0.013	0.013	0.001608	0.00049
R2	0.000027	0.0033	0.013	0.027	0.00189	0.00037

7.5 Davies-Bouldin index analysis for real datasets

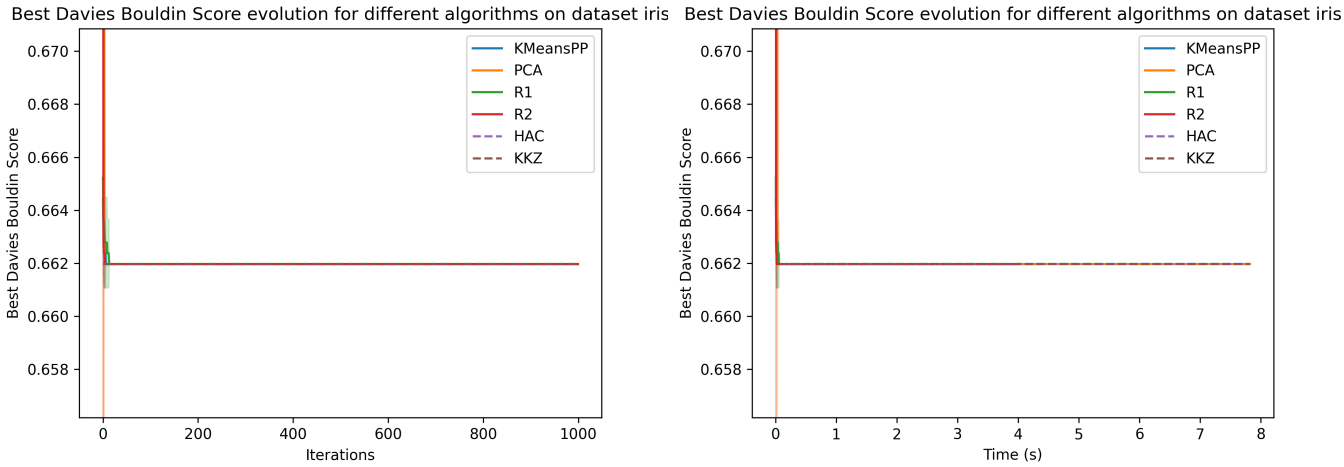

Figure 7: Results on IRIS, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

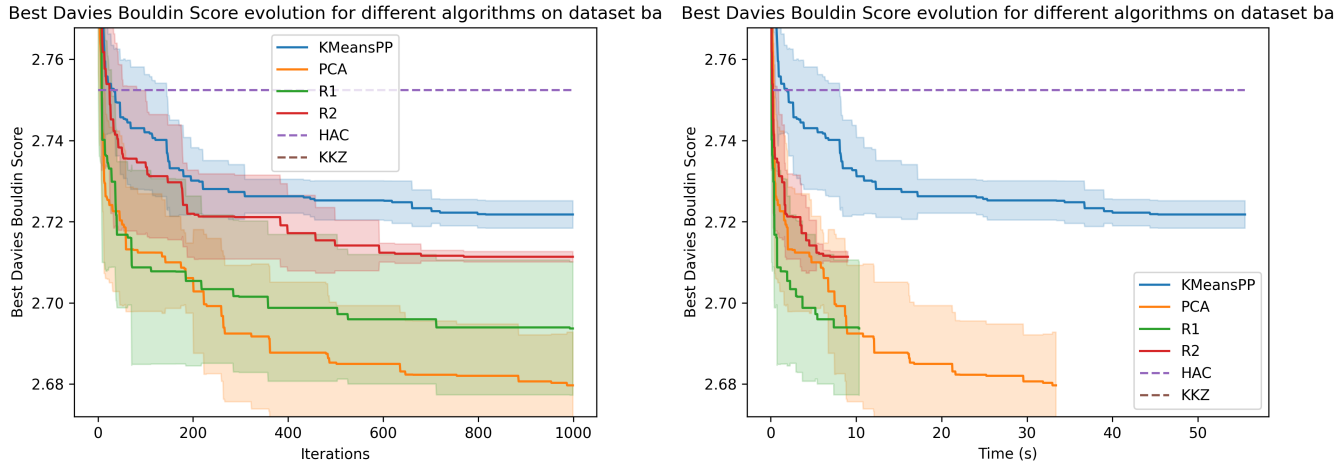

Figure 8: Results on Binary Alphabet, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

7.6 Distorsion index analysis for synthetic datasets

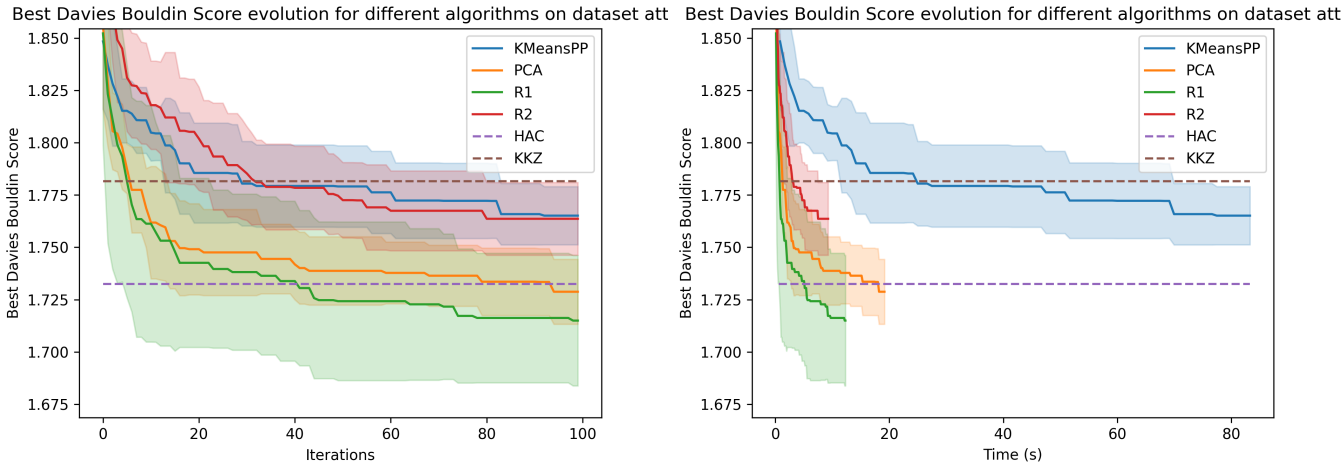

Figure 9: Results on AT&T, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

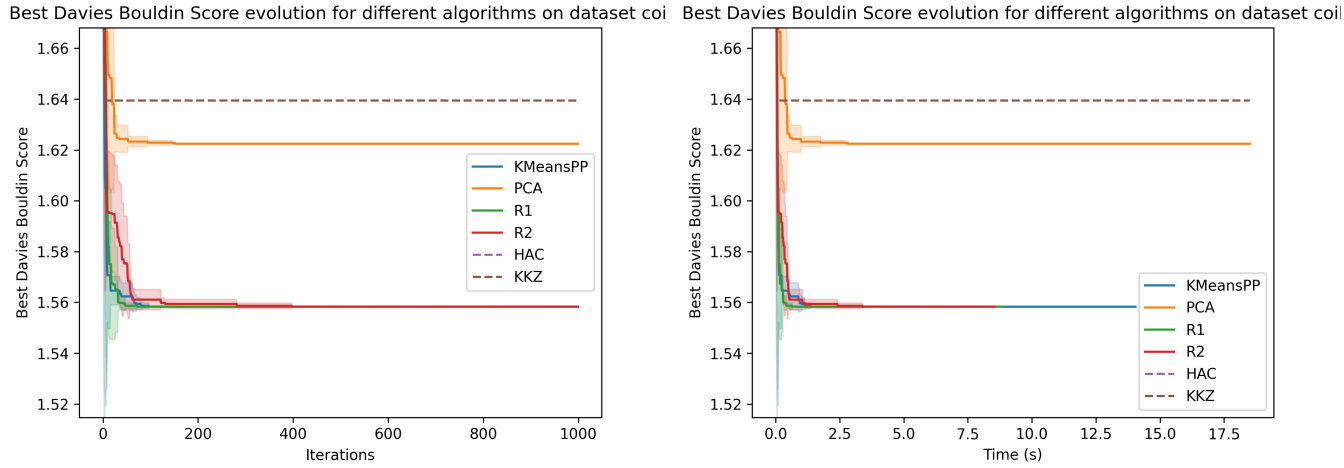

Figure 10: Results on Coil20, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

7.7 Execution time and dispersion tables for synthetic datasets

Table 4: Quartile Coefficient of Dispersion on synthetic data

Algorithm	Bridging	Ellipsoidal	Gauss1	Gauss2	Laplace	Nested	Uniform
HAC	0	0	0	0	0	0	0
KKZ	0	0	0	0	0	0	0
KMeansPP	0.041	0.041	0.016	0.0041	0.089	0.12	0.032
PCA-GuidedSearch	0.27	0.131	0.34	0.052543	0.32	0.305	0.32
R1	0.31	0.13	0.26	0.058	0.29	0.31	0.29
R2	0.31	0.14	0.43	0.036	0.34	0.29	0.35

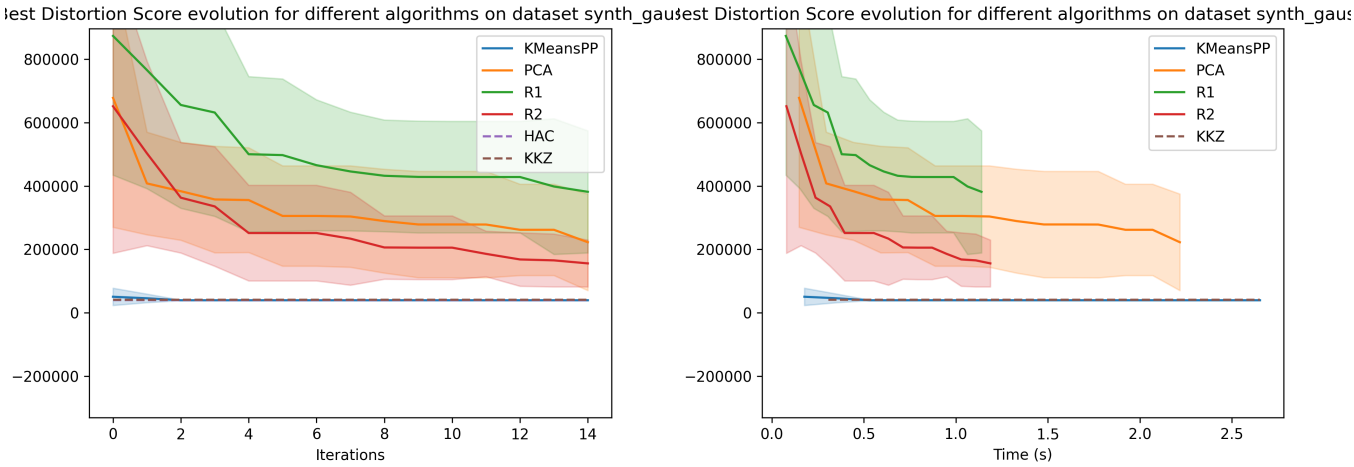

Figure 11: Results on Gauss 1 Synthetic, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

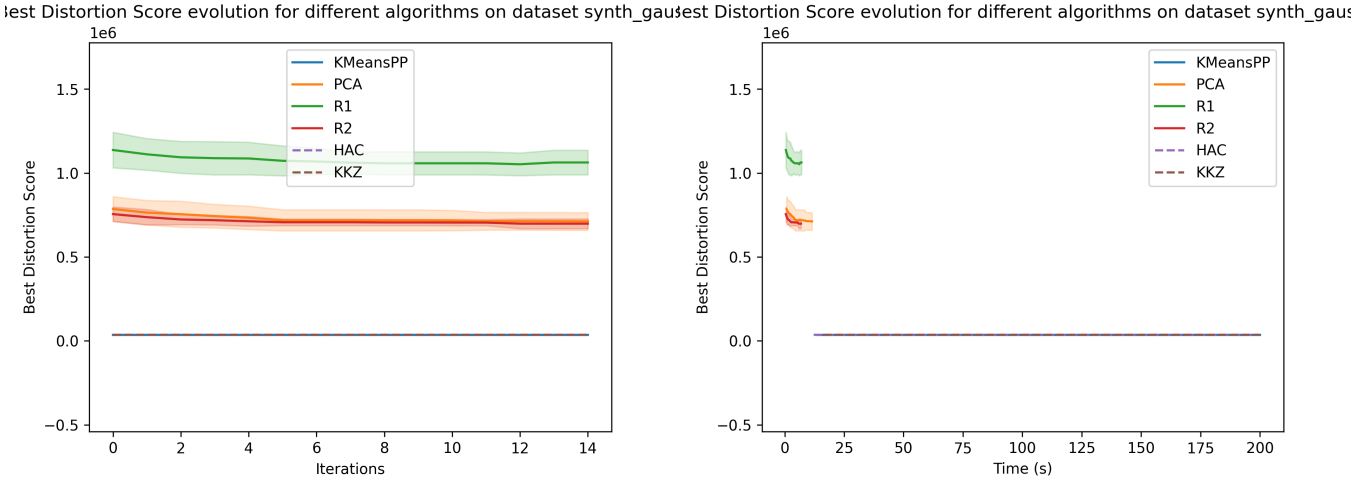

Figure 12: Results on Gauss 2 Synthetic, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

Table 5: Execution time on synthetic data

Algorithm	Bridging	Ellipsoidal	Gauss1	Gauss2	Laplace	?ested	Uniform
HAC	0.16	0.13	11.39	12.11	0.462898	0.14	0.20
KKZ	0.11	0.0940	0.31	16.30	0.377892	0.097	0.18
KMeansPP	0.019	0.010	0.18	13.33	0.016	0.020	0.015
PCA-Guided Search	0.023	0.019	0.15	0.76	0.019	0.024	0.020
R1	0.0080	0.0073	0.076	0.47	0.0076	0.0084	0.0071
R2	0.0077	0.0075	0.079	0.45	0.0073	0.0083	0.0081

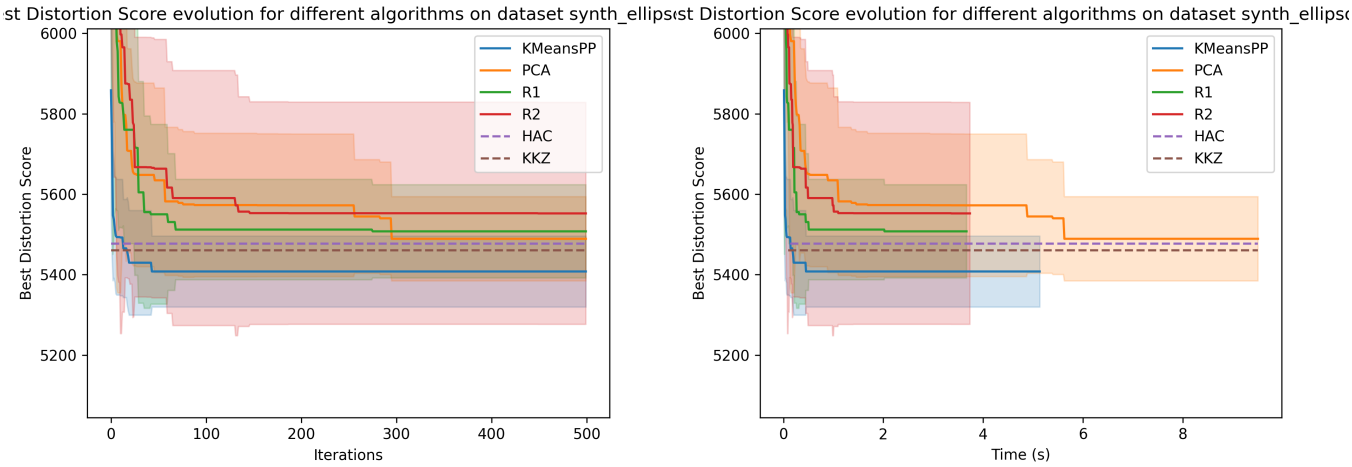

Figure 13: Results on Ellipsoidal Synthetic, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

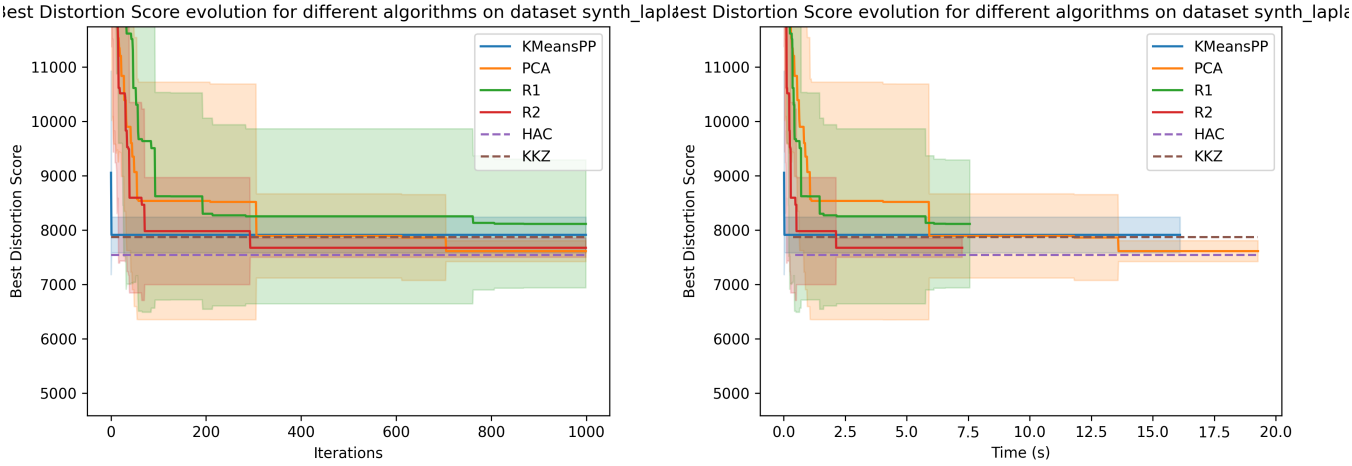

Figure 14: Results on Laplace Synthetic, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

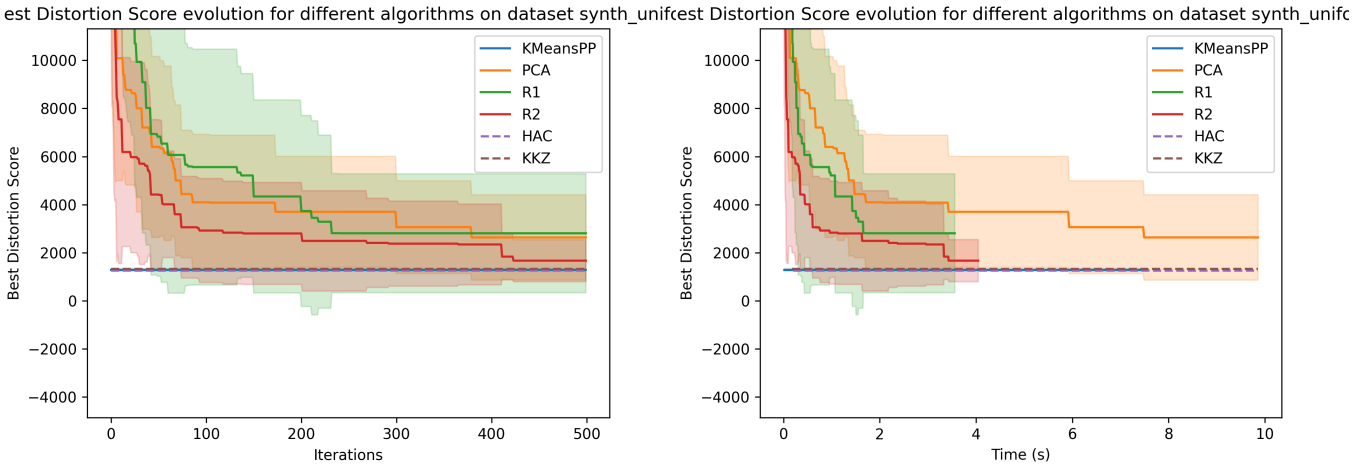

Figure 15: Results on Uniform Synthetic, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

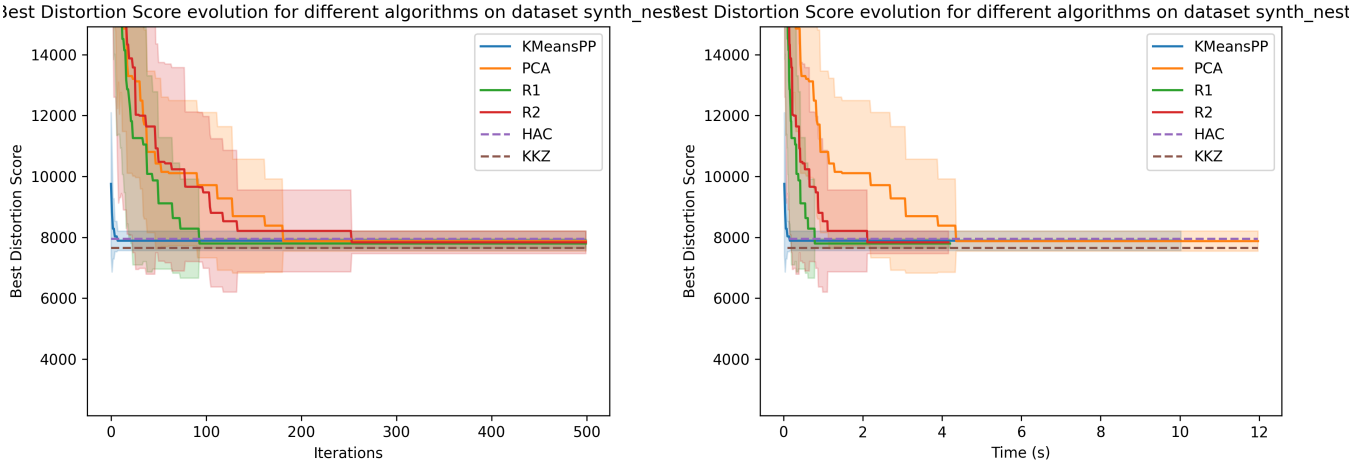

Figure 16: Results on Nested Synthetic, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index

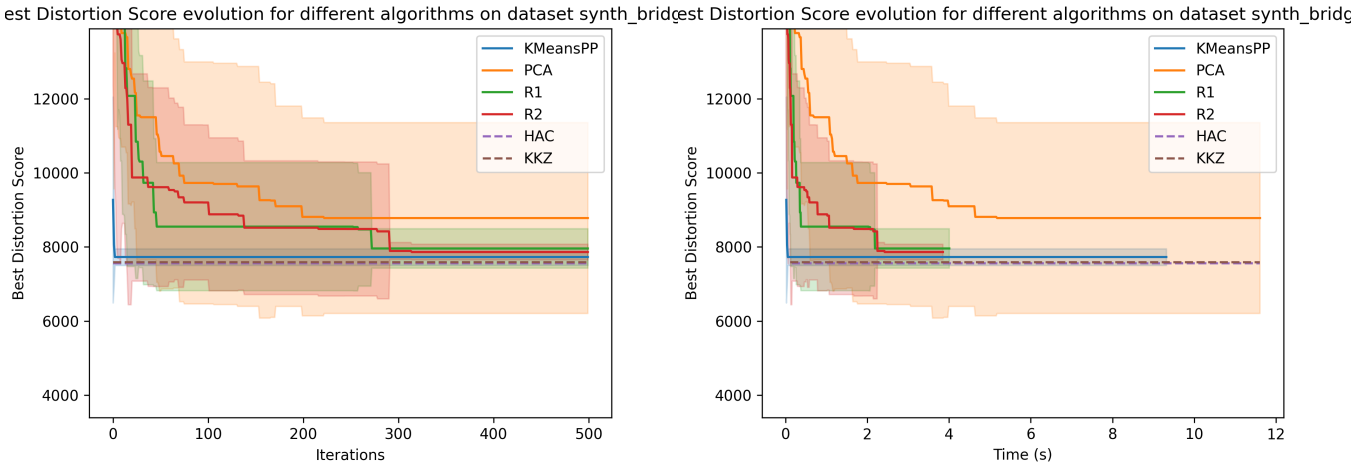

Figure 17: Results on Bridging Synthetic, as a function of the number of iterations (left) and of the runtime (right) for the Davies-Bouldin index