Machine Learning (2021) 110:1975–2003
https://doi.org/10.1007/s10994-021-06021-7

An empirical comparison between stochastic and deterministic centroid initialisation for K-means variations

Avgoustinos Vouros[1] · Stephen Langdell[2] · Mike Croucher[2] · Eleni Vasilaki[1]

Received: 29 November 2019 / Revised: 14 June 2021 / Accepted: 15 June 2021 /
Published online: 12 July 2021

Abstract
K-Means is one of the most used algorithms for data clustering and the usual clustering method for benchmarking. Despite its wide application it is well-known that it suffers from a series of disadvantages; it is only able to find local minima and the positions of the initial clustering centres (centroids) can greatly affect the clustering solution. Over the years many K-Means variations and initialisation techniques have been proposed with different degrees of complexity. In this study we focus on common K-Means variations along with a range of deterministic and stochastic initialisation techniques. We show that, on average, more sophisticated initialisation techniques alleviate the need for complex clustering methods. Furthermore, deterministic methods perform better than stochastic methods. However, there is a trade-off: less sophisticated stochastic methods, executed multiple times, can result in better clustering. Factoring in execution time, deterministic methods can be competitive and result in a good clustering solution. These conclusions are obtained through extensive benchmarking using a range of synthetic model generators and real-world data sets.

Keywords K-Means clustering · Deterministic clustering · Benchmarking

1 Introduction

The most well-known algorithm in the field of clustering analysis is the K-Means algorithm. Its simplicity, versatility and efficiency makes it popular in many different research fields (Pena et al., 1999; Jain, 2010). Despite its reputation and success in many different studies, it has a series of disadvantages such that it can detect only spherical and well-separated clusters, it is sensitive to outliers, highly dependent on the

Editor: Tijl De Bie

✉ Avgoustinos Vouros
avouros@gmail.com

[1] Department of Computer Science, University of Sheffield, Sheffield S10 2TN, UK

[2] Numerical Algorithms Group (NAG), Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, UK

features (dimensions) of the data set and it only converges to local minima (Jain, 2010). Over the years a number of K-Means variations (Lloyd's K-Means—Slonim et al., 2013, Hartigan–Wong's K-Means—Hartigan, 1975), K-Means inspired algorithms (K-Medians—Charu & Chandan, 2013), and K-Means initialisation methods (Celebi et al., 2013) have been proposed in order to overcome some of these issues. Such methods have also enhanced KMeans with additional properties such as feature selection mechanisms (Witten & Tibshirani, 2010; Kondo et al., 2016), semi-supervised capabilities (Bilenko et al., 2004; Vouros & Vasilaki, 2021), and outliers robustification (Al Hasan et al., 2009; Brodinová et al., 2017).

In the literature there are various studies regarding the importance of the initial selection of cluster centroids for the performance of the K-Means algorithm (Jain, 2010) and extensive testing on various initialisation techniques (Celebi et al., 2013; Fränti & Sieranoja, 2019), but a detailed comparison on the effects on these techniques on common K-Means variations is not available. We hypothesize that sophisticated initialisation methods alleviate the need for complex clustering and, if deterministic, they could lead to satisfactory solutions within a single execution of the clustering algorithm. Consequently, they would alleviate the need for executing a stochastic method multiple times and picking the best clustering based on some criterion.

In order to investigate this hypothesis we compare different clustering initialisation methods, namely Random (MacQueen, 1967), K-Means++ (Arthur & Vassilvitskii, 2007), Maximin (Gonzalez, 1985), ROBust INitialisation (ROBIN) (Al Hasan et al. 2009), Kaufman and Rousseeuw (2009) and Density K-Means++ (DK-Means++) (Nidheesh et al., 2017), and their effects on common K-Means variations, Lloyd's K-Means (Jain, 2010), Hartigan–Wong's K-Means (Hartigan & Wong, 1979; Hartigan, 1975) and K-Medians (Charu and Chandan, 2013). We show that more sophisticated initialisation methods reduce on average the performance difference among the K-Means implementations and that the deterministic DK-Means++ method can achieve better average performance than stochastic methods. Nevertheless there is a trade-off, simplistic stochastic methods can achieve better clustering performance if executed multiple times due to the potential of discovering better local minima. For very large data sets where execution time is a factor, a single run using a deterministic initialisation method can be competitive compared to multiple runs using stochastic initialisation methods.

A similar study comparing many different intialisation methods has been performed by Celebi et al. (2013) but it is focused on algorithms of linear complexity without considering various K-Means implementations. Recently, another study (Fränti and Sieranoja 2019) was performed on stochastic initialization heuristics for K-Means and on how much the algorithm can be improved by repetition. They based their conclusions on a clustering benchmark (Fränti & Sieranoja, 2018, 2019) which contains standalone data sets with different properties and they showed that K-Means performance is in general poor on unbalanced data sets and that the algorithm is not affected by high dimensionality while more iterations can improve its performance on overlapping clusters. In our case we performed a more extensive benchmarking by taking into consideration data set generation models as well as standalone data sets. The models gave us the ability to perform hypothesis testing in order to strengthen our conclusions and to account for variability.

The code of the clustering methods, data set model generators, scripts and a standalone application to reproduce this research are available in the GitHub repository https://github.com/avouros/Code-KMeans-benchmark (under the branch additions).

2 Material and method

2.1 The K-means algorithm

Given K initial centroids, the K-Means algorithm (Jain, 2010) assigns the data points into K clusters in a way that minimizes the within cluster sum of squares (WCSS):

$$WCSS = \sum_{k=1}^{K} \sum_{\begin{pmatrix} i=1 \\ x_{i:} \in c_k \end{pmatrix}}^{n_k} \sum_{j=1}^{p} (x_{ij} - m_{kj})^2, \tag{1}$$

where K is the number of clusters, n_k the number of data points (observations) of the k-th cluster and p the dimensionality (number of features) of a given dataset; x_{ij} is the value of the j-th feature of the i-th data point, $x_{i:}$ is the vector representing the i-th datapoint; $m_{k:}$ specifies the location of the *k-th* cluster centroid. This problem is equivalent to maximizing the between cluster sum of squares (BCSS) which is given by Witten and Tibshirani (2010):

$$BCSS = \sum_{j=1}^{p} \left(\sum_{i=1}^{n} (x_{ij} - \mu_{1j})^2 - \sum_{k=1}^{K} \sum_{\begin{pmatrix} i=1 \\ x_{i:} \in c_k \end{pmatrix}}^{n_k} (x_{ij} - m_{kj})^2 \right), \tag{2}$$

where $\mu_{1:}$ specifies the global centroid assuming that all the data points belong to one cluster.

2.1.1 Lloyd's K-means

Lloyd's method is the most commonly used K-Means and the standard K-Means clustering method in many programming languages such as MATLAB (MATLAB, 2019) and Python (van Rossum, 1995). The steps of this algorithm are as follows (Jain, 2010):

1. Initialise K initial centroids $m_{1j}, \ldots, m_{Kj}$ using some initialisation method.
2. Assign each data point $x_{i:}$ to cluster c_{k*},

$$k^* = \underset{k}{argmin} \left\{ \sum_{j=1}^{p} (x_{ij} - m_{kj})^2 \right\}. \tag{3}$$

3. Recompute each cluster centroid using the formula,

$$m_{kj} = \frac{\sum_{\begin{pmatrix} i=1 \\ x_{i:} \in c_k \end{pmatrix}}^{n_k} x_{ij}}{n_k}. \tag{4}$$

4. Go to step 2 until converge.

2.1.2 Hartigan–Wong's K-means

Hartigan–Wong's K-Means algorithm is an alternative to Lloyd's K-Means and the default K-Means of the R language (R Core Team, 2013). In the study of Slonim et al. (2013) it is shown that this method has lower probability of converging to a local minima solution compared to Lloyd's method in exchange of extra complexity. The steps of the algorithm are as follows (Hartigan, 1975; Slonim et al., 2013):

1. Initialise K initial centroids $m_{1j}, \dots, m_{Kj}$ using an initialisation method.
2. Assign each data point $x_{i:}$ to cluster $c_{k'}$,

$$k' = \underset{k}{argmin}\left\{ \sum_{j=1}^{p}(x_{ij} - m_{kj})^2 \right\}.$$

3. Set an indicator $s = 1$.
4. For each data point $x_{i:}$:
   (a) Remove it from its cluster $c_{k'}$.
   (b) Compute the centroid of $c_{k'}$ using the remaining points in that cluster,

$$m_{k'j} = \frac{1}{n_{k'}} \sum_{\begin{pmatrix} i = 1 \\ x_{i:} \in c_{k'} \end{pmatrix}}^{n_{k'}} x_{ij}.$$

   (c) Assign $x_{i:}$ to cluster c_{k^*},

$$k^* = \underset{k}{argmin}\left\{ \sum_{j=1}^{p}(x_{ij} - m_{kj})^2 \right\}.$$

   (d) Recompute the centroid of the cluster c_{k^*},

$$m_{k^*j} = \frac{1}{n_{k^*}} \sum_{\begin{pmatrix} i = 1 \\ x_{i:} \in c_{k^*} \end{pmatrix}}^{n_{k^*}} x_{ij}.$$

   (e) If $k^* \neq k'$ set $s = 0$.

5. If $s = 0$, set $s = 1$ and go to step 4.

2.1.3 The K-medians algorithm

The K-Medians algorithm (Charu & Chandan, 2013) is similar to the K-Means but uses the median instead of the mean to calculate the cluster centroid. The objective function of the algorithm is given by the equation:

$$E = \sum_{k=1}^{K} \sum_{\substack{i=1 \\ x_{i:} \in c_k}}^{n_k} \sum_{j=1}^{p} |x_{ij} - \bar{m}_{kj}|, \tag{5}$$

where $\bar{m}_{kj}$ specifies the location of the *k-th* cluster centroid in the *j*-th dimension which is computed by taking the median of the data points x_{ij} belonging to that cluster. K-Medians corresponds to the L_1-norm as opposed to the L_2-norm of K-Means (Charu & Chandan, 2013). The use of median in place of the mean makes the K-Medians algorithm robust to outliers (Feldman & Schulman, 2012; Whelan et al., 2015) since the median has a breaking point of 0.5, i.e. even if half of the data set is corrupted by outliers the median of the corrupted data set will be similar to the median of the original data set (Lopuhaa & Rousseeuw, 1991). The common implementation of the algorithm is similar to Lloyd's K-Means where, in the $3rd$ step of Lloyd's algorithm, the median is used to compute the new centroids locations instead of the mean., i.e. $\forall k\ \bar{m}_{k:} = median(\{x_{i:}\}), x_{i:} \in c_k$.

2.2 K-means initialisation methods

Let $D(x_{i:})$ denote the distance between data point $x_{i:}$ and the nearest of the selected cluster centroids, $m_{k:}$, $k = 1, \dots, L$, with L being the number of selected centroids ($L \leq K$):

$$D(x_{i:}) = \min_k \sqrt{\sum_{j=1}^{p} (x_{ij} - m_{kj})^2}. \tag{6}$$

2.2.1 Random

The initialisation method of MacQueen (1967) proposes a random selection of data points from the data set which will be the initial centroids. This is one of the earliest clustering initialisation techniques and an improvement of Jancey's method (1966). The latter study suggested the centroids to be at random locations within the minimum hypersphere of the data set but this might result in empty clusters to be generated after the execution of the K-Means algorithm.

2.2.2 K-means++

K-Means++ (Arthur & Vassilvitskii, 2007) is a standard clustering initialisation technique in many programming languages such as MATLAB and Python. It has linear complexity $\mathcal{O}(N)$ and it uses a probabilistic approach in order to select as initial centroids data points that are far away from each other. The steps of this algorithm are as follows:

1. Select randomly a data point $x_{i:}$ as the first centroid $m_{1:}$ and set $k = 2$.
2. Choose another data point $x_{i':}$ as the next centroid $m_{k:}$ with probability

$$p(x_{i':}) = \frac{D(x_{i':})^2}{\sum_{i=1}^{n} D(x_{i:})^2}$$

and set $k = k + 1$.

3. While $k \leq K$ go to step 2.

2.2.3 Maximin

The maximin method of Gonzalez (1985) picks data points as cluster centroids that are far apart form each other.

1. Select randomly a data point $x_{i:}$ as the first centroid $m_{1:}$ and set $k = 2$.
2. Select as the next centroid $m_{k:} = x_{i':}$ with $i' = \underset{i}{argmax}\{D(x_{i:})\}$ and set $k = k + 1$.
3. While $k \leq K$ go to step 2.

Maximin has linear complexity $\mathcal{O}(N)$. The study of Katsavounidis et al. (1994) proposed a modification in the first step of the algorithm to select as the first centroid the data point with the maximum Euclidean norm (Celebi et al., 2013). In this way the method can become deterministic.

2.2.4 Kaufman

Kaufman and Rousseeuw (2009) proposed a deterministic method for centroids initialisation. Their method is as follows (Pena et al., 1999):

1. Select the closest data point to the global centroid of the data set as the first centroid $m_{1:}$ and set $k = 2$.
2. For every two non-selected data points $x_{i:}$ and $x_{i':}$ calculate,

$$C_{i'i} = max\left\{ D(x_{i:}) - \sqrt{\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2}, 0 \right\}.$$

3. Select as the next centroid $m_{k:} = x_{i^*:}$, with $i^* = \underset{i}{argmax}\{\sum_{i'} C_{i'i}\}$ and set $k = k + 1$.
4. While $k \leq K$ go to step 2.

Kaufman's and Rousseeuw's algorithm has quadratic complexity $\mathcal{O}(N^2)$ because of the computation of the pairwise distances (Celebi et al., 2013).

2.2.5 Robust initialisation (ROBIN)

Al Hasan et al. (2009) is a robust to outliers initialisation method. It uses the Local Outlier Factor (LOF) (Breunig et al., 2000) in order to select as initial centroids data points that are far away from each other and also representative points of dense regions in the data set. In addition it requires one more tuning parameter which is the number of neighboring data points mp to be consider when computing the LOF of each data point. The LOF score of each data point, $LOF(x_{i:}, mp)$, is given by the algorithm below (Al Hasan et al., 2009) (where $N(x_{i:}, mp)$ is the set of the mp nearest data points to the $x_{i,:}$ data point, with $|N(x_{i:}, mp)| \geq mp$):

1. Compute the density of each data point $x_{i:}$,

$$density(x_{i:}, mp) = \frac{|N(x_{i:}, mp)|}{\sum_{x_{i':} \in N(x_{i:}, mp)} \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}}, i \neq i'. \tag{7}$$

2. Compute the average relative density of each data point $x_{i:}$,

$$ard(x_{i:}, mp) = \frac{density(x_{i:}, mp)}{\frac{\sum_{x_{i':} \in N(x_{i:}, mp)} density(x_{i':}, mp)}{|N(x_{i:}, mp)|}}. \tag{8}$$

3. Compute the LOF score of each data point $x_{i:}$,

$$LOF(x_{i:}, mp) = \frac{1}{ard(x_{i:}, mp)}. \tag{9}$$

The ROBIN algorithm ($K > 1$) is described below (Al Hasan et al., 2009):

1. Pick a reference data point $x_{r:}$ from the data set.
2. Sort the data points in decreasing order of their distance from $x_{r:}$.
3. For each $x_{i:}$ in sorted order, pick the first data point $x_{i':}$ for which $LOF(x_{i':}, mp) \approx 1$ as the first centroid $m_{1:}$ and set $k = 2$.
4. Sort the data points in decreasing order based on $D(x_{i:})$. For the formula of $D(x_{i:})$ refer to Eq. 6.
5. For each $x_{i:}$ in sorted order, pick the first data point $x_{i':}$ for which $LOF(x_{i':}, mp) \approx 1$ as the next centroid $m_{k:}$ and set $k = k + 1$.
6. While $k \leq K$ go to step 4.

The computational cost of this method is dominated by the complexity of sorting, which is $\mathcal{O}(N \log N)$ (Celebi et al., 2013) but for the LOF score calculation we have a choice of algorithms varying from $\mathcal{O}(N)$ to $\mathcal{O}(N^2)$, that can be chosen based on dimensionality-related constraints, see Breunig et al. (2000). Regarding the 4th step of the algorithm, in an R implementation (refer to the study of Brodinová et al. (2017)) the formula $LOF(x_{r':}, mp) < 1.05$ was used but since the LOF score can also be lower than 1, in our experiments, we used the formula $1 - e < LOF(x_{r':}, mp) < 1 + e$ where e was set to 0.05. In the original algorithm (Al Hasan et al., 2009) the authors are using the algorithm in a deterministic manner by setting the reference point on step 2, x_r : to the origin. In the R implementation of Brodinová et al. (2017) the reference point is chosen at random. In this study we test both methods, *ROBIN(S)* will refer to the stochastic method of Brodinová et al. (2017) while *ROBIN(D)* will refer to the deterministic method of Al Hasan et al. (2009).

2.2.6 Density K-means++ (DK-means++)

DK-Means++ (Nidheesh et al., 2017) is a deterministic method for centroids initialisation based on data density. It is an improved method of Rodriguez and Laio (2014), Lan et al. (2015) since it requires only to define the number of clusters K and utilizes a heuristic to detect dense regions in the data set based on a radius ε in order to select optimal centroids. The radius ε can be computed form the following algorithm (Nidheesh et al., 2017):

1. Construct the minimum spanning tree of the distance matrix of the data set.

2. Let Λ be the weights of edges of the Minimum Spanning Tree and IQR the Inter Quartile Range. Then,

$$\varepsilon = 3 \cdot IQR(\Lambda) + 75thpercentile(\Lambda).$$

The DK-Means++ algorithm is described below (Nidheesh et al., 2017):

1. Compute the local density $p(x_{i:})$ of each data point using the formula:

$$p(x_{i:}) = \sum_{x_{i'} \in \varepsilon\text{-}neighbors(x_{i:})} \exp\left(\frac{-\sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}}{\varepsilon} \right).$$

   where ε-*neighbors*$(x_{i:})$ are the data points falling under the hypersphere with centroid $x_{i:}$ and radius ε.
2. Normalize $p(x_{i:})$ using the *min–max* normalization.
3. The first cluster centroid $m_{1:}$ is the data point $x_{i^*:}$ for which $p(x_{i^*:}) = max\{p(x)\}$. Then $m_{1:} = x_{i^*:}$ and $k = 2$.
4. Compute the prospectiveness all data points that are not selected as centers given the formula, $\phi(x_{i:}) = p(x_{i:}) \cdot D(x_{i:})$. For the formula of $D(x_{i:})$ refer to Eq. 6.
5. The next centroid $m_{k:}$ is the data point with maximum prospectiveness: $m_{k:} = x_{i^*:}$ with $i^* = \underset{i}{argmax}\{\phi(x_{i:})\}$ and $k = k + 1$.
6. While $k \leq K$ go to step 4.

The computation of ε-*neighbors* contributes to the complexity of DK-Means++. It is dominated by the distance matrix computation, which is $O(n^2)$. The computation of the Minimum Spanning Tree depends of the algorithm used to compute it, efficient implementations of Kruskal's algorithm and Prim's algorithm require $O(n \log n)$ time to compute a minimum spanning tree (Moret & Shapiro, 1992). As $O(n^2)$ is the dominating complexity, the entire process of finding initial centroids, of DK-Means++ has a time complexity of $O(n^2)$.

2.3 Clustering evaluation

In the literature there are many indexes for assessing the clustering performance (Al Hasan et al., 2009; Rendón et al., 2011; Fränti & Sieranoja, 2018). Here, in order to evaluate the clustering solutions, we selected to use one external (supervised) criterion, the Purity and one internal (unsupervised) criterion, the Silhouette index.

2.3.1 Purity

Let our data belong to different classes ℓ_{k} and the number of classes K equal the number of clusters K, for each cluster the purity is defined as Rendón et al. (2011),

$$P_{c_k} = \frac{1}{n_k} \underset{\mathsf{k}}{max}\{n_{\ell_{\mathsf{k}}}^{(k)}\}, \tag{10}$$

where $\underset{\mathsf{k}}{max}\{n_{\ell_{\mathsf{k}}}^{(k)}\}$ specifies the number of data points of the dominant class of the k-th cluster. The overall clustering purity index is then computed as,

$$P = \sum_{i=1}^{K} \frac{n_k}{n} P_{c_k}. \tag{11}$$

The purity index is bounded between (0 1]; larger values of purity correspond to better performance accuracy and a purity of 1 specifies an accuracy of 100% meaning that each cluster has data points from only one class.

2.3.2 Silhouette index

The Silhouette index is a value that specifies the degree of similarity between a data point and other data points of the same cluster and the dissimilarity between a data point and other data points in different clusters. The Silhouette index of the data point $x_{i:} \in c_k$ is given by 12 (Rousseeuw, 1987),

$$S_{x_{i:}} = \frac{b_{x_{i:}} - a_{x_{i:}}}{max\{b_{x_{i:}}, a_{x_{i:}}\}}, \tag{12}$$

where $a_{x_{i:}}$ is the average distance of $x_{i:}$ and all the other data points in the cluster that $x_{i:}$ belongs to

$$a_{x_{i:}} = \frac{1}{n_k - 1} \sum_{\begin{pmatrix} i' = 1 \\ x_{i':} \in c_k \end{pmatrix}}^{n_k} \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2} \tag{13}$$

and $b_{x_{i:}}$ is the minimum average distance of $x_{i:}$ to all the other data points in other clusters,

$$b_{x_{i:}} = \underset{k'}{min} \frac{1}{n_{k'}} \sum_{\begin{pmatrix} i' = 1 \\ x_{i':} \in c_{k'} \end{pmatrix}}^{n_{k'}} \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2, k' \neq k.} \tag{14}$$

We can then define the average Silhouette index,

$$S = \frac{1}{n} \sum_{i=1}^{n} S_{x_{i:}}. \tag{15}$$

The Silhouette index is commonly used to estimate the number of clusters in a data set but it can also be used to assess the clustering quality (Rousseeuw, 1987). The Silhouette index is bounded between −1 and 1, where 1 specifies maximum separation of clusters and maximum within cluster density while other indexes, such as the distortion score, gives only an estimation of the latter.

3 Benchmarks

In our experiments we use the synthetic data from the studies of Tibshirani et al. (gap statistic) (2001), Yan and Ye (weighted gap statistic) (2007) and Brodinová et al. (2017). A summary of the models can be found in Table 1 grouped by specific properties of the

Table 1 Gap (Tibshirani et al., 2001) and weighted gap statistic (Yan & Ye, 2007) data sets models

	Points	p	K
Gaussian models			
gap model (gap 2)	25,25,50	2	3
gap model 3 (gap 3)	25 or 50	3	4
weighted gap model 1 (wgap 1)	25 to 50	2	6
weighted gap model 6 (wgap 6)	50 each	2	6
10-D Gaussian models			
gap model 4 (gap 4)	25 or 50	10	2
weighted gap model 5 (wgap 5)	25 to 50	10	2
Unbalanced model			
weighted gap model 2 (wgap 2)	100,15	2	2
Exponential model			
weighted gap model 3 (wgap 3)	50 each	2	4
Elongated models			
gap model 5 (gap 5)	100 each	3	2
weighted gap model 4 (wgap 4)	100 each	2	2

Points: the number of data points per cluster, *or* indicates that a random number was selected among the specified numbers for each cluster, *to* indicates that a random number was selected between the specified numbers for each cluster; p: number of features or attributes of the data set (dimensions); K: number of generated clusters. Gaussian models: clusters of low dimensionality generated from Gaussian distributions. 10-D Gaussian models: clusters of higher dimensionality generated from Gaussian distributions. Elongated models: clusters generated by adding Gaussian noise across lines. Unbalanced model: data sets containing Gaussian clusters of very different sizes, exponential: non-Gaussain clusters generated from the exponential distribution. For more information about the parameters used for the Gaussians refer to the relevant studies and the supplementary material. Visualization (when possible) of a data set from each model is available in Fig. 1

models. For more information refer to the relevant studies and also to Fig. 1 for a sample visualization of each model. We exclude model 1 from the gap statistic study (Tibshirani et al., 2001) since it contains only one cluster. The Brodinová et al. (2017) generator was used to generate high-dimensional data sets consisting of informative and non-informative features. No noise injection (attributes with noise contamination) was considered in the current study. To avoid situations of overlapping clusters the minimum Euclidean distance between any two points in different clusters was set to 3 and data sets violating this rule were re-generated. A summary of the models can be found in Table 2. Next we use our own synthetic data sets models consisted of clusters with mixed properties. We will refer to these models as *mixed* (refer to Fig. 1 for a sample visualization of these models).

- *model 1* generates 3-dimensional clusters, 1 spherical and 2 elongated. The spherical cluster is an 80 points Gaussian cluster at the origin with standard deviation of 0.1. The two elongated clusters have 100 points each and are generated as follows: $x_1 = x_2 = x_3 = t$ with t taking 100 equally spaced values from -1 to 1. We then add Gaussian noise with standard deviation of 0.3 to each dimension. The second dimension of the first elongated cluster was shifted by 2 from the centre of the spherical clus-

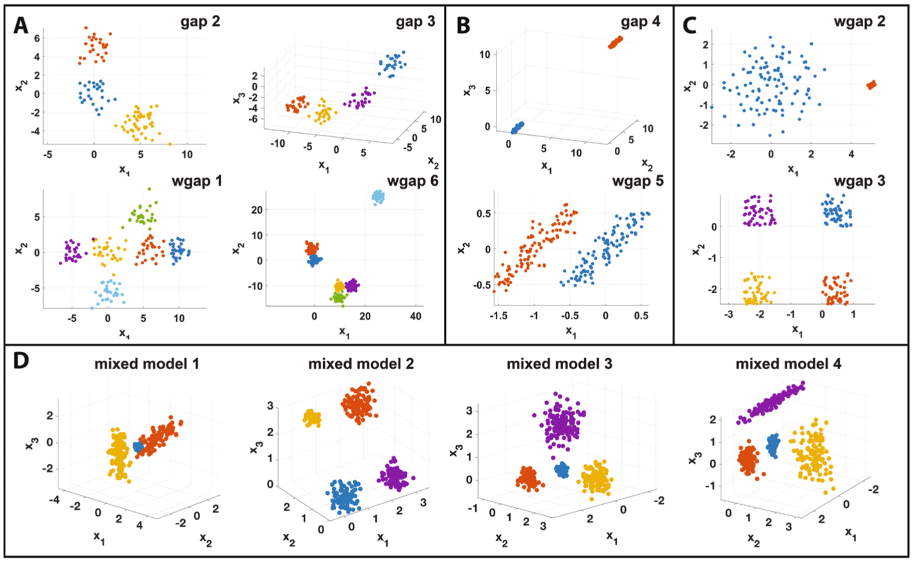


Fig. 1 Model visualization. Examples of data used in this study. Gap (gap) and weighted gap (wgap) models are separated into three categories: **a** 4 Gaussian models, **b** 2 elongated models and **c** one highly unbalanced model (wgap2) and one non-Gaussian model (wgap3) in which the clusters are generated from the exponential distribution. **d** Mixed models: these are the additional models proposed in our study that contain clusters with mixed properties such as different sizes (unbalanced) and/or generated from Gaussian and non-Gaussian distributions

ter. Similarly the second dimension of the second elongated cluster was shifted by − 2 and the first dimension was rotated by 180^0.

- *model 2* generates 3-dimensional non-Gaussian and normal clusters. It generates: (a) a cluster from an exponential distribution with rate of 1 and truncated at [−11] containing 80 points, (b) a cluster from an exponential distribution with rate of 1 and and truncated at [23] with 100 points, (c) a Gaussian cluster of 80 points with mean [0.5, 2.5, 2.5] and standard deviation of 0.1 in every dimension and (d) a Gaussian cluster of 100 points with mean [2.5, 0.5, 0.5] and standard deviation of 0.2 in every dimension.
- *model 3* generates 3-dimensional Gaussian clusters with different standard deviations. The first cluster has 80 points with mean at the origin and standard deviation of 0.1 on each dimension. The second cluster has of 100 points with mean [2, 0, 0] and standard deviation of 0.2 on each dimension. The third cluster has of 120 points with mean [0, 2, 0] and standard deviation of 0.3 on each dimension. The forth cluster consists of 140 points with mean [0, 0, 2] and standard deviation of 0.4 on each dimension.
- *model 4* generates 3-dimensional mixed Gaussian clusters. The first cluster consists of 80 points with mean [0, 0, 0] and standard deviations [0.1, 0.1, 0.2]. The second cluster consists of 100 points with mean [2, 0, 0] and standard deviations [0.1, 0.2, 0.3]. The third cluster consists of 120 points with mean [0, 2, 0] and standard deviations [0.2, 0.4, 0.6]. The forth cluster consists of 140 points with mean [0, 0, 2] and standard deviations [1.0, 0.1, 0.1].

Table 2 Brodinova model generator (Brodinová et al., 2017)

Name	Points	p		K
		+	-	
brod 1	120	20	0	3
brod 2	400	20	0	10
brod 3	120	15	5	3
brod 4	400	15	5	10
brod 5	120	10	10	3
brod 6	400	10	10	10
brod 7	120	1000	0	3
brod 8	400	1000	0	10
brod 9	400	1500	0	10
brod 10	1250	1500	0	50
brod 11	50 to 100	1000	0	3
brod 12	50 to 100	1000	0	10

The minimum allowed Euclidean distance between two data points in different clusters was set to 3 and no noise injection was considered. Name: name of the model; Points: the total number of data points in the data set; p: number of features or attributes of the data set, *Informative (+)* indicates attributes that are required to describe the data set while *Non-informative (–)* indicates variables that should be ignored; K: number of generated clusters. These models are creating high-dimensional Gaussian clusters of different shapes using two different distributions, one for the informative and one for the uninformative variables. *Left table:* Each cluster contains 40 data points. The parameters of these models are selected to test the performance of the clustering algorithm in data sets with different degrees of informative and/or uninformative features *Right table:* The first four models create higher-dimensional balanced clusters (clusters of equal sizes) and the last two higher-dimensional unbalanced clusters each with number of points randomly selected between 50 to 100. The parameters of these models are selected to test the performance of the clustering algorithm in balanced and unbalanced data sets of higher dimensionality with increasing number of clusters. The input space was selected to be sparse, i.e. a few hundred points in 1000 or 1500 dimensions to avoid the slow computation of the Kaufman algorithm

Table 3 Real data sets from the UCI repository (Asuncion & Newman, 2007)

Name	Points	p	K
Iris	50,50,50	4	3
Ionosphere	225,126	34	2
Wine	59,71,48	13	3
Breast cancer	444,239	9	2
Glass	70,76,17, 19,9,29	9	6
Yeast	463,5,35,44, 51,163,244, 429,20,30	8	10

Points: the number of data points per cluster; p: number of features or attributes of the data set (dimensions); K: number of generated clusters

We also consider the *S-sets* (Fränti & Virmajoki, 2006) and the *A-sets* (Kärkkäinen & Fränti, 2002) obtained from the "clustering basic benchmark" which was used in the studies of Fränti and Sieranoja (2018), Fränti and Sieranoja (2019). The aforementioned studies were dedicated to the K-Means properties, advantages and disadvantages and assessed used various synthetic data sets. Both models contain 2-dimensional data; *S-sets* contains 4 data sets with 5000 data points distributed among 15 Gaussian clusters with different degree of clustering overlap (Fränti & Virmajoki, 2006) and *A-sets* contains 3 data sets with 20, 35 and 50 clusters and 150 data points per cluster (Kärkkäinen & Fränti, 2002). For more information about these data sets refer to the relevant studies. Finally we considered a selection of data sets from the UCI repository (Asuncion & Newman, 2007): Iris, Ionosphere, Wine, Breast Cancer, Glass and Yeast. More information about these data sets is shown on Table 3.

4 Results

We test the performance of the K-Means variations, Lloyd's (Jain, 2010) Hartigan–Wong's (Hartigan & Wong, 1979; Hartigan, 1975) and K-Medians (Charu & Chandan, 2013) initialised using the eight different clustering initialisation methods named: Random (MacQueen, 1967), K-Means++ (Arthur & Vassilvitskii, 2007), Maximin(S) (Gonzalez, 1985), ROBIN(S) (Brodinová et al., 2017), Kaufman (Kaufman & Rousseeuw, 2009), ROBIN(D) (Al Hasan et al., 2009), DK-Means++ (Nidheesh et al., 2017) and Maximin(D) (Katsavounidis et al., 1994). For the ROBIN variations the *mp* parameter specifying the number of neighbor data points was set to 10 as in the original study (Al Hasan et al., 2009). For the Hartigan–Wong's algorithm NAG's implementation was used (Numerical Algorithms Group (NAG), 2019).

We conceptually consider a "sophistication" scale for the initialisation methods based not only on their execution time but also on the complexity of their underlying operators. For example DK-Means++ and ROBIN would be considered more sophisticated than Kaufman since they incorporate more advanced statistics while Kaufman uses only distances and still has a complexity of $O(N^2)$. Our scale is as follows: Random < K-Means++ < Maximin < Kaufman < ROBIN < DK-Means++.

In our experiments we use the synthetic data sets models from the studies of gap statistic (Tibshirani et al., 2001) and weighted gap statistic (Yan & Ye, 2007) (refer to Table 1, 10 sets in total), Brodinová et al. (2017) (refer to Table 2, 12 sets in total) and other four custom data sets models (refer to Methods and Fig. 1, 4 sets in total). From each model we generated 40 data sets and for each data set the stochastic methods were executed 50 times. These numbers were selected to provide a good statistical sample. We also use the "clustering data sets" (*S-sets* Fränti & Virmajoki, 2006 and *A-sets* Kärkkäinen & Fränti, 2002) from the studies of Fränti and Sieranoja (2018, 2019) and real-world data sets from the UCI repository (Asuncion & Newman, 2007): Iris, Ionosphere, Wine, Breast Cancer, Glass and Yeast (see Table 3). For each of these data sets we use the same set up of executing the stochastic methods 50 times.

For all our hypothesis testing on the data set models we used the Paired Samples Wilcoxon Test, a non-parametric alternative to paired t-test, For the outcome of the test we generally use the following symbols for the level of significance, * for *p*-value

< 0.05; ** for p-value < 0.01; *** for p-value < 0.001; **** for p-value < 0.0001. We evaluated the monotonic relationship of Silhouette index and Purity via a large sample of clustering results on the multiple executions of the methods across all our data sets (20000 cases). Using Spearman's rank correlation coefficient, we confirmed that Purity and Silhouette have a strong monotonic relation (Spearman's Rho 0.97). As a side note, we also considered Distortion (Al Hasan et al., 2009) as an unsupervised index but found the monotonic relationship with Purity weaker (Spearman's Rho 0.65).

4.1 Comparison on the average performance among stochastic and among deterministic methods

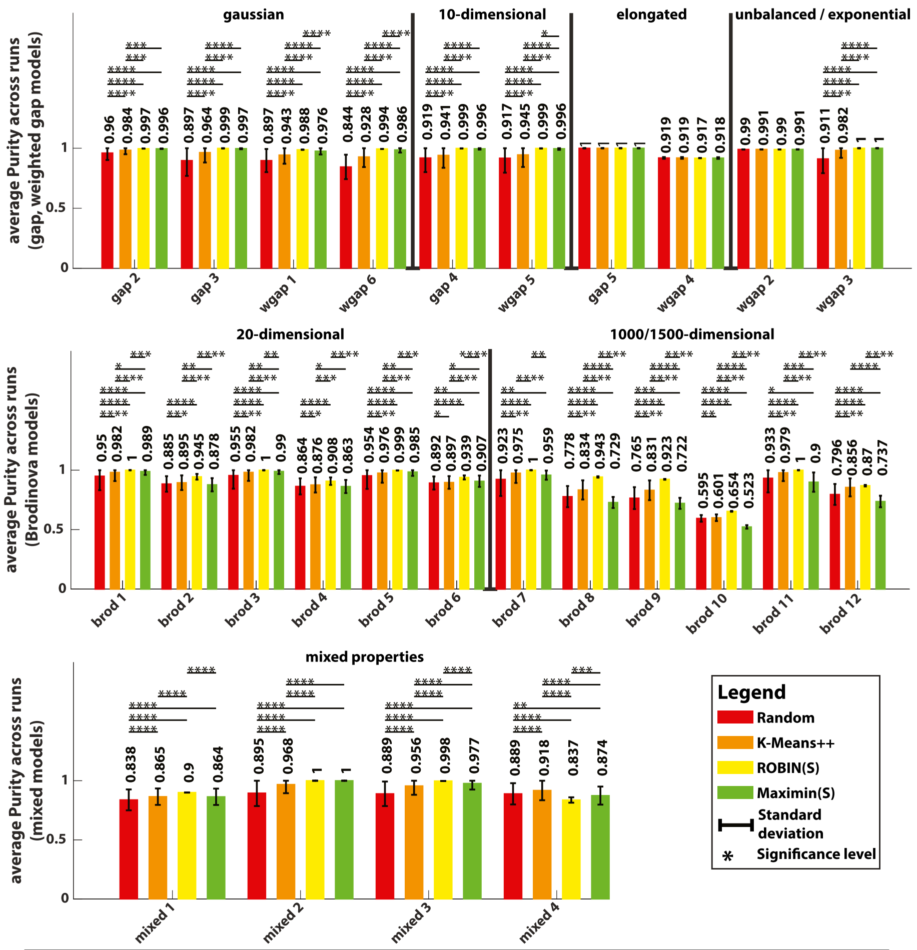
We assess on average the performance of stochastic methods as well as the performance of deterministic methods. We calculate the average performance of stochastic methods on 50 different runs across 40 different data sets for each one of our 26 models (10 gap and weighted gap, 12 Brodinova, 4 mixed models). Deterministic methods were executed once on the 40 data sets.

Based on Fig. 2 the average performance of K-Means variations increases by using more sophisticated initialisation methods and ROBIN(S) initialisation provides the best average performance followed by Maximin(S) and K-Means++ while Random initialisation results in the poorest performance. For the deterministic methods, shown on Fig. 3, we observe again that the average performance of K-Means variations increases by using more sophisticated initialisation methods. DK-Means++ achieved the best average performance followed by ROBIN(D) and then by Kaufman and Maximin(D). Finally we wanted to assess if more sophisticated initialization methods alleviate the need for complex clustering. For this reason we performed comparisons among the K-Means variations initialised with either Random and K-Means++ or Kaufman and DK-Means++ methods. Maximin and ROBIN have both stochastic and deterministic variations of equal sophistication thus we excluded them. As shown in Table 4, deterministic methods that are more sophisticated (as per our definition) reduce performance differences among the different variants of the K-Means algorithms.

Table 4 Average performance comparison among K-Means variations using simple stochastic and more sophisticated deterministic initialisation methods

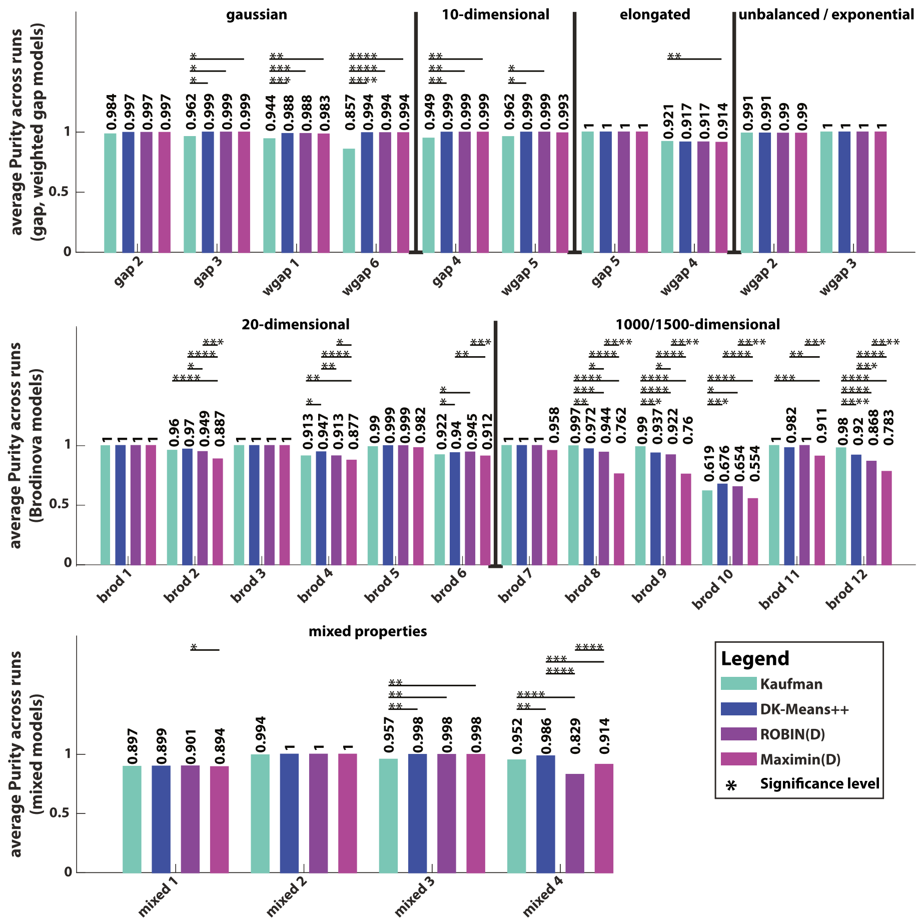
Initialization method	Total number of instances	Significantly better average performance		
		HW vs Ll	HW vs KMed	Ll vs KMed
Random	26	13 vs 0	18 vs 4	6 vs 4
K-Means++	26	10 vs 0	13 vs 3	2 vs 5
Total	52	23 vs 0	39 vs 12	8 vs 9
Kaufman	26	1 vs 1	2 vs 6	1 vs 6
DK-Means++	26	1 vs 0	2 vs 4	1 vs 5
Total	52	2 vs 1	4 vs 10	2 vs 11

We compare two K-Means variations (Hartigan–Wong's K-Means: HW, Lloyd's K-Means: Ll and K-Medians: KMed) initialised with the same method on 26 occasions (10 gap and weighted gap, 12 Brodinova and 4 mixed models). To calculate performance, we averaged the Purity index across the 50 initial conditions and 40 data sets for each model and the comparison is based on the times that there was significant difference between the two algorithms



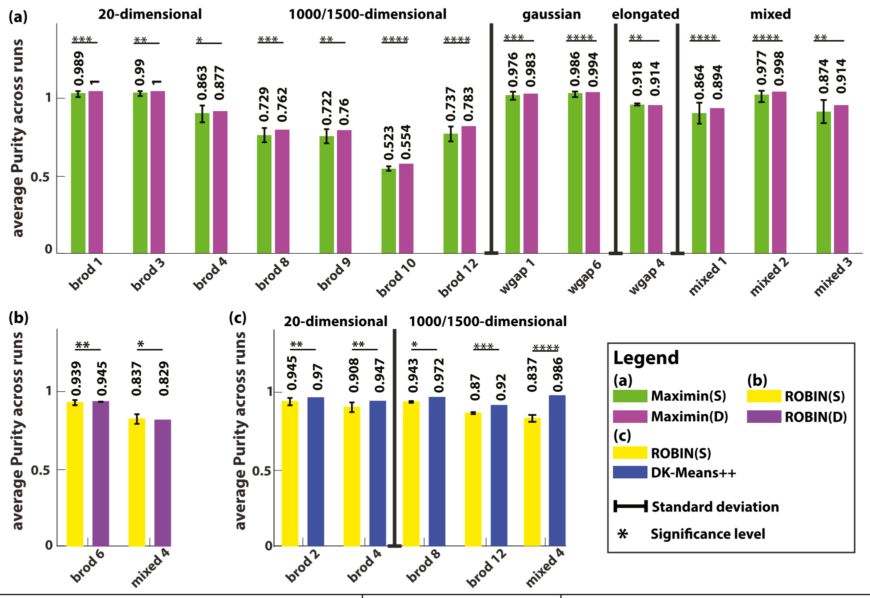
Initialization method	Total number of instances	Significantly better average performance		
		HW	Ll	KMed
Random vs K-Means++	26	0 vs 23	0 vs 23	0 vs 24
Random vs ROBIN(S)	26	1 vs 22	3 vs 22	2 vs 23
Random vs Maximin(S)	26	6 vs 15	6 vs 16	6 vs 16
K-Means++ vs ROBIN(S)	26	1 vs 21	3 vs 21	2 vs 22
K-Means++ vs Maximin(S)	26	8 vs 13	9 vs 14	7 vs 13
ROBIN(S) vs Maximin(S)	26	17 vs 1	18 vs 3	19 vs 1

Fig. 2 Comparison on the average performance among stochastic initialisation methods. Each plot shows the performance of the Hartigan–Wong's K-Means clustering solution using the Purity index (y-axis) on different data sets models (x-axis) and initialized with different stochastic methods. To calculate performance, we averaged the Purity index across the 50 initial conditions and 40 data sets for each model (gap, weighted gap, Brodinova and mixed). The errorbars are showing the (average) standard deviation across the 40 data sets. Solid lines on any two bars underline the level of significant difference between the corresponding methods. The accompanying Table below the figure shows a summary of the comparisons through all the K-Means variations (Hartigan–Wong's K-Means (HW), Lloyd's K-Means (Ll) and K-Medians (KMed)) where there is a significant performance difference between the compared methods



Initialization method	Total number of instances	Significantly better average performance		
		HW	Ll	KMed
Kaufman vs DK-Means++	26	3 vs 10	3 vs 9	4 vs 8
Kaufman vs ROBIN(D)	26	4 vs 8	6 vs 9	4 vs 6
Kaufman vs Maximin(D)	26	8 vs 5	11 vs 5	9 vs 5
DK-Means++ vs ROBIN(D)	26	6 vs 0	7 vs 0	6 vs 1
DK-Means++ vs Maximin(D)	26	9 vs 0	11 vs 0	10 vs 1
ROBIN(D) vs Maximin(D)	26	9 vs 1	10 vs 1	9 vs 1

Fig. 3 Comparison on the performance among deterministic initialisation methods. Each plot shows the performance of the Hartigan–Wong's K-Means clustering solution using the Purity index (y-axis) on different data sets models (x-axis) and initialized with different deterministic methods. To calculate performance, we averaged the Purity index across the 40 data sets for each model (gap, weighted gap, Brodinova and mixed). Solid lines on any two bars underline the level of significant difference between the corresponding methods (cases of no significant differences are not shown). The accompanying Table below the figure shows a summary of the comparisons through all the K-Means variations (Hartigan–Wong's K-Means (HW), Lloyd's K-Means (Ll) and K-Medians (KMed)) where there is a significant performance difference between the compared methods

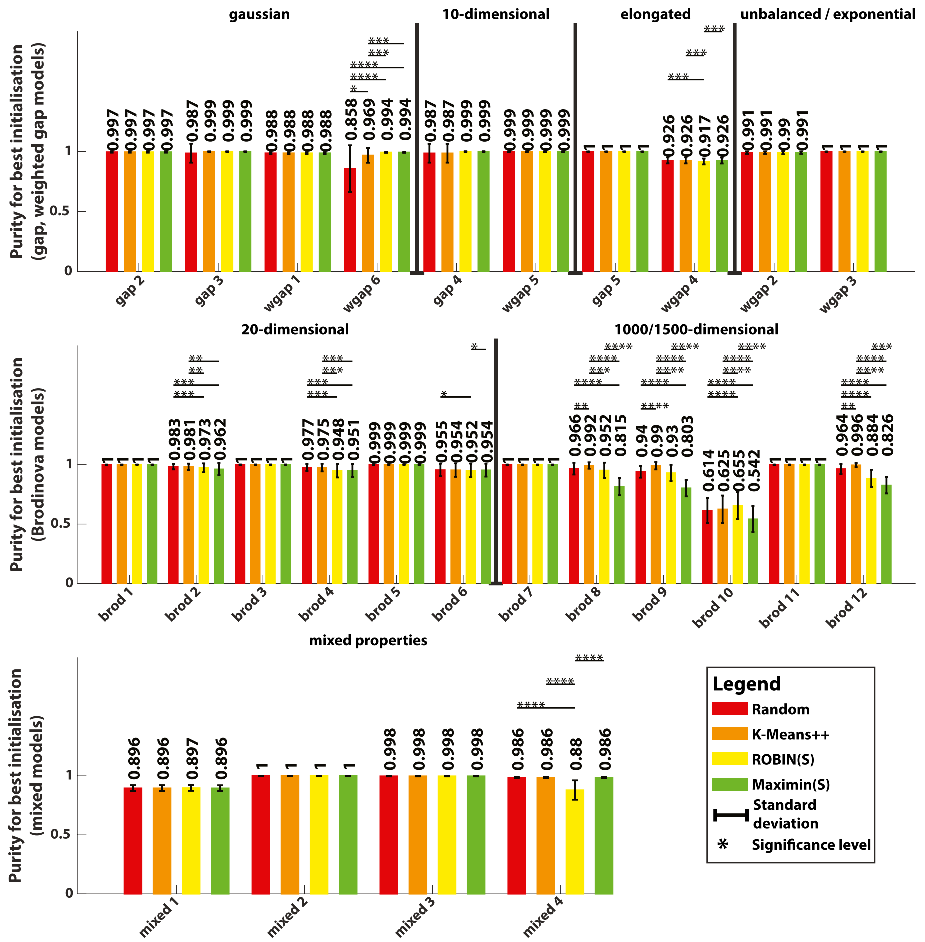


Initialization method	Total number of instances	Significantly better average performance		
		HW	Ll	KMed
Maximin(S) vs Maximin(D)	26	1 vs 12	3 vs 12	0 vs 13
ROBIN(S) vs ROBIN(D)	26	1 vs 1	1 vs 2	1 vs 0
ROBIN(S) vs DK-Means++	26	0 vs 5	0 vs 7	1 vs 5

Fig. 4 Comparisons on the average performance between stochastic and deterministic methods. Each plot shows the performance of the Hartigan–Wong's K-Means clustering solution using the Purity index (y-axis) on different data sets models (x-axis) and initialized with different stochastic methods (only the cases where significant difference was present are shown). To calculate performance, we averaged the Purity index across the 50 initial conditions and 40 data sets for each model (gap, weighted gap and Brodinova). The errorbars (on the stochastic methods only) are showing the average standard deviation across the 40 data sets. Solid lines on any two bars underline the level of significant difference between the corresponding methods (cases of no significant differences are not shown). The accompanying Table below the figure shows a summary of the comparisons through all the K-Means variations (Hartigan–Wong's K-Means (HW), Lloyd's K-Means (Ll) and K-Medians (KMed)) where there is a significant performance difference between the compared methods

4.2 Comparison of the average performance between stochastic and deterministic methods

Following on from our previous conclusions, we wanted to assess if overall deterministic methods provide on average better performance than stochastic methods. For this reason we compared the stochastic and deterministic variations of Maximin and ROBIN as well as the best stochastic performer (ROBIN(S) see Fig. 2) and the best deterministic performer (DK-Means++ see Fig. 3). Based on the results in Fig. 4: (a) Maximin(D) is on average better than Maximin(S); (b) ROBIN(D) and ROBIN(S) are on average equivalent; (c) DK-Means++ is better than ROBIN(S).



Initialization method	Total number of instances	Significantly better maximum performance Purity for best Silhouette		
		HW	Ll	KMed
Random vs K-Means++	26	0 vs 4	0 vs 4	1 vs 4
Random vs ROBIN(S)	26	6 vs 3	8 vs 2	6 vs 2
Random vs Maximin(S)	26	6 vs 1	9 vs 1	9 vs 1
K-Means++ vs ROBIN(S)	26	7 vs 2	9 vs 2	8 vs 2
K-Means++ vs Maximin(S)	26	6 vs 1	9 vs 1	9 vs 1
ROBIN(S) vs Maximin(S)	26	4 vs 3	4 vs 4	4 vs 2

Fig. 5 Comparison on the maximum performance among stochastic initialisation methods. Each plot shows the performance of the Hartigan–Wong's K-Means clustering solution using the purity corresponding to the best Silhouette score achieved within 50 different executions (y-axis) on different data sets models (x-axis) and initialized with different stochastic methods. Purity for best Silhouette score was averaged over the 40 data sets for each model (gap, weighted gap and Brodinova) The errorbars are showing the standard deviation across the 40 data sets. Solid lines on any two bars underline the level of significant difference between the corresponding methods. The accompanying Table below the figure shows a summary of the comparisons through all the K-Means variations (Hartigan–Wong's K-Means (HW), Lloyd's K-Means (Ll) and K-Medians (KMed)) where there is a significant performance difference between the compared methods

4.3 Comparison of the maximum performance across multiple runs of stochastic and deterministic methods

Next, we wanted to compare the stochastic and the deterministic methods but based on the maximum performance that the former can achieve on multiple repetitions. We run each of the stochastic methods 50 times and select the best outcome based on the Silhouette index. We then report its corresponding value according to the purity index. We expect that due to the many repetitions, stochastic methods can find different local minima and potentially result in a better performance at the cost of multiple repetitions.

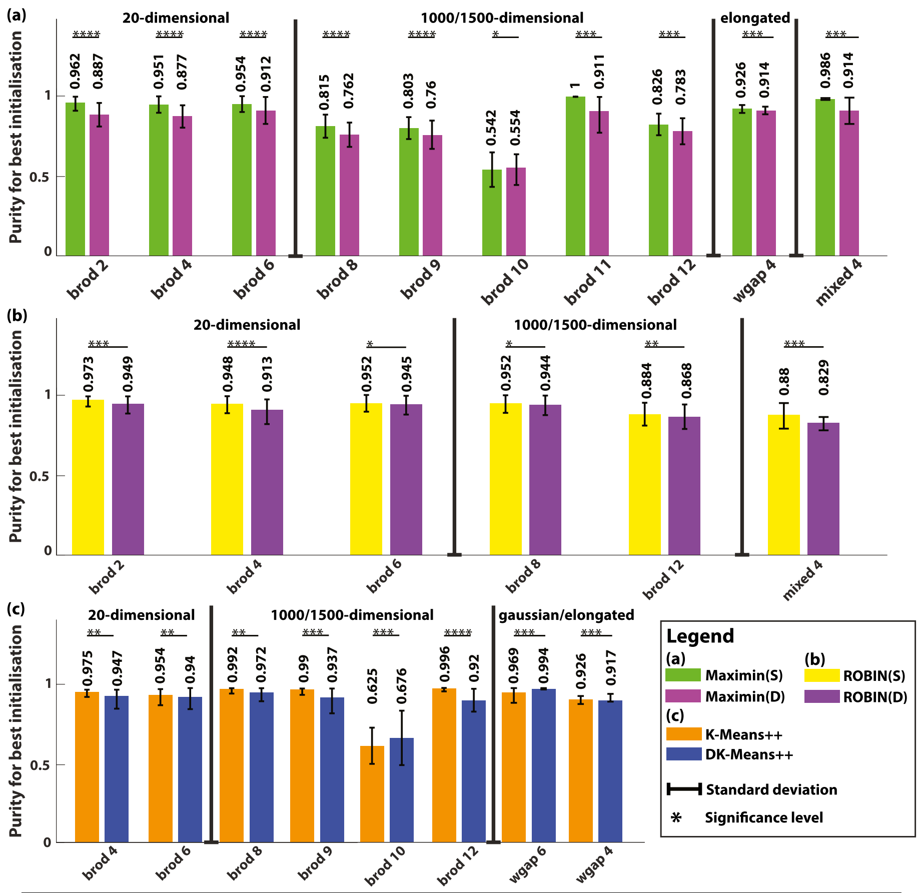
Firstly, we repeat the comparison among the different stochastic methods but based on the maximum performance that they can achieve. Figure 5 shows the relevant results and, opposite to our observations on the average performance, stochastic methods have higher chances of obtaining a better clustering result with multiple execution. K-Means++ is the best method followed by Random while ROBIN(S) and Maximin(S) have almost similar performance.

Afterwards, we compared the maximum performance of stochastic methods with the performance of deterministic methods similarly to our previous experiment (refer to Fig. 4). We compare the stochastic and deterministic variations of Maximin and ROBIN as well as the best stochastic performer of the current experiment (K-Means++) and the best deterministic performer (DK-Means++ see Fig. 3). Based on the results in Fig. 6, stochastic variations of Maximin and ROBIN achieve overall better performance than their deterministic counterparts and K-Means++ is better than DK-Means++.

We also compare the K-Means variations using different intialisation methods. Based on the result on Table 5 K-Medians achieves the best performance followed by Hartigan–Wong's; Lloyd's was the worst performer. Nevertheless, these systematic differences correspond to only 1.5% purity difference.

4.4 Standalone synthetic and real-world data sets

We regard the standalone data sets as cases where supervised information is unknown and we assess the performance of the algorithms based on the Silhouette index. Detailed results for each data set (minimum, maximum, average performance and variance for each K-Means variation) are illustrated in the supplementary material. Given that these data sets are unique it is difficult to draw definite conclusions similar to the ones where data set models were used but we would like to highlight some observations. DK-Means++ was always able to achieve the best performance of the unsupervised methods while ROBIN(D) failed to achieve the best performance in the cases of A-sets 1, S-Sets 3 and S-Sets 4 when Lloyds and K-Medians were considered; Kaufman and Maximin(D) where the worst performers. From the stochastic methods ROBIN(S) always managed to achieve the maximum performance apart from one case of S-Sets 3 when the Harigan-Wong K-Means was considered; Random was the worst performer. For the real-world data sets most algorithms behaved the same but Maximin(S) outperformed everyone else in the cases of Yeast (all K-Means variations) and Ionosphere (Lloyd's K-Means only). In the case of Glass (all K-Means variations) K-Means++ and Maximin(S) had the best performances. However, with the real data sets we should consider the fact that in rare situations the number of clusters equals to the number of classes (Gehring et al., 2015) thus it might not be the best examples for clustering benchmarking. Also the relatively better performance of Maximin(S) appears only in these few cases where the Silhouette index indicates poor clustering results in general. In such cases comparative conclusions may not be meaningful as these specific results could be a product of chance.



Initialization method	Total number of instances	Significant better average performance		
		HW	Ll	KMed
Maximin(S) vs Maximin(D)	26	9 vs 1	11 vs 1	10 vs 1
ROBIN(S) vs ROBIN(D)	26	6 vs 0	6 vs 0	6 vs 0
K-Means++ vs DK-Means++	26	6 vs 2	6 vs 2	7 vs 2

Fig. 6 Comparisons on the maximum performance between stochastic and deterministic methods. Each plot shows the performance of the Hartigan–Wong's K-Means clustering solution using the purity corresponding to the best Silhouette score achieved within 50 different executions (y-axis) on different data sets models (x-axis) and initialized with different stochastic methods (only the cases where significant difference was present are shown). Purity for best Silhouette score was averaged over the 40 data sets for each model (gap, weighted gap and Brodinova). The errorbars are showing the standard deviation across the 40 data sets. Solid lines on any two bars underline the level of significant difference between the corresponding methods (only cases with significant difference are showed). The accompanying Table below the figure shows a summary of the comparisons through all the K-Means variations (Hartigan–Wong's K-Means (HW), Lloyd's K-Means (Ll) and K-Medians (KMed)) where there is a significant performance difference between the compared methods

Table 5 Best performance comparison on K-Means variations using different initialisation methods

Initialization method	Total number of instances	Significantly better maximum performance		
		HW vs Ll	HW vs KMed	Ll vs KMed
Random	26	4 vs 1	3 vs 7	1 vs 7
K-Means++	26	5 vs 1	2 vs 7	1 vs 4
ROBIN(S)	26	4 vs 0	1 vs 3	1 vs 7
Maximin(S)	26	2 vs 1	3 vs 5	2 vs 3
Total	104	15 vs 3	9 vs 22	5 vs 21
Kaufman	26	1 vs 1	2 vs 6	1 vs 6
DK-Means++	26	1 vs 0	2 vs 4	1 vs 5
ROBIN(D)	26	6 vs 0	2 vs 2	1 vs 6
Maximin(D)	26	7 vs 0	2 vs 2	1 vs 6
Total	104	15 vs 1	8 vs 14	4 vs 23

Each row compares two K-Means variations (Hartigan–Wong's K-Means: HW, Lloyd's K-Means: Ll and K-Medians: KMed) initialised with the same method on 26 occasions (10 gap and weighted gap, 12 Brodinova and 4 mixed models). The comparison is based on the times that there was significant difference between the two methods on their maximum performance based on the purity for the best Silhouette score

4.5 Average number of runs for which stochastic methods reach or surpass deterministic methods

In the aforementioned experiments we considered 50 executions of the clustering algorithm using stochastic methods. On average deterministic methods provide better results than stochastic methods but overall stochastic methods may lead to a better clustering solution. We would therefore like to quantify how often this happens. To estimate this, we divide the number of total repetitions (50) by the number of cases where the stochastic method performed better than the deterministic. Table 6 summarises the results of this analysis on selected data sets based on their size, dimensionality and number of clusters among two stochastic (Random and K-Means++) and two deterministic methods (DK-Means++ and ROBIN(D)). Based on the results the number of repetitions required for the clustering method using K-Means++ in order to match or surpass the performance of DK-Means++ and ROBIN(D) are less compared with Random. This was expected given the performance comparison of Random and K-Means++ but an important result is the following: there are cases (Yeast, A-sets 2 and A-sets 3) where these stochastic methods fail to match or surpass the performance of deterministic methods under 50 runs. We also observe that when the size of the data set surpasses the 1000 data points the number of required repetitions is significantly high. Finally we should mention that these performance differences are minor, in the order of 10^{-3} (purity) on average.

4.6 Execution time analysis

Finally, we performed an execution time analysis on the initialisation methods using a selection of the data sets depending on their size, dimensionality and number of clusters; data sets with equivalent properties were omitted. The analysis was performed as follows:

Table 6 Average number of runs for which stochastic initialisations achieve equivalent or better performance than deterministic initialisations

	Rand DKM++	Rand ROBIN(D)	KM++ DKM++	KM++ ROBIN(D)	Size, dimensions, number of clusters
gap 2	5	5	4	4	100, 2, 3
wgap 2	6	6	4	4	115, 2, 2
wgap 4	6	6	5	5	200, 2, 2
wgap 3	3	3	3	3	200, 2, 4
gap 5	4	2	4	2	200, 3, 2
wgap 6	8	8	7	6	300, 2, 6
gap 3	5	5	4	4	*143, 3, 4*
gap 4	10	10	6	6	*158, 10, 2*
wgap 1	7	7	6	6	*227, 2, 6*
wgap 5	**14**	**14**	6	6	*141, 10, 2*
brod 1	4	4	5	5	120, 20, 3
brod 2	**18**	**17**	**17**	**17**	400, 20, 3
Iris	7	7	3	3	150, 4, 3
Wine	3	3	2	2	178, 13, 3
Glass	7	7	6	6	214, 9, 6
Ionosphere	3	3	2	2	351, 34, 2
Breast cancer	**12**	**12**	7	7	683, 9, 2
Yeast	**N/A**	**16**	**27**	**14**	1484, 8, 10
A-sets 1	**28**	**26**	**26**	**16**	3000, 2, 15
S-Sets 1	**34**	**34**	**26**	**15**	5000, 2 ,15
A-sets 2	**N/A**	**N/A**	**34**	**33**	5250, 2, 35
A-sets 3	**N/A**	**N/A**	**N/A**	**N/A**	7500, 2, 50

Each column shows a comparison between clustering initialised with stochastic and deterministic methods (Rand = Random, DKM++ = DK-Means++, KM++ = K-Means++). Each cell value corresponds to the number of executions of the K-Means clustering initialised with the stochastic method to reach or surpass its performance if it was initialised with the deterministic method and executed once. N/A values mean that in these occasions the stochastic clustering was not able to match or surpass the performance of the deterministic clustering under 50 executions. Values higher than 10 are marked in bold. The data sets are arranged based on their *size*, *dimensionality* and *number of clusters* (see info on last column; numbers in italics correspond to the average number of elements should the model generated data sets of different sizes

each initialisation methods followed by K-Means clustering (Lloyd's K-Means) was executed 50 times and the average running time was taken into consideration.

- The benchmarking was exclusively performed on a *personal laptop* with the following properties: Dell G7; Intel i7-9750H processor; 16 GB RAM; Windows 10 Pro edition.
- All the algorithms were written in MATLAB but the LOF score for ROBIN was computed using R code (specifically the dbscan package Hahsler et al., 2019) because we found that the MATLAB implementation was very slow.
- The running time recording includes the initialisation method and the clustering algorithm. For ROBIN the computation of LOF was included in the execution duration as well as the computation of ε for the DK-Means++.

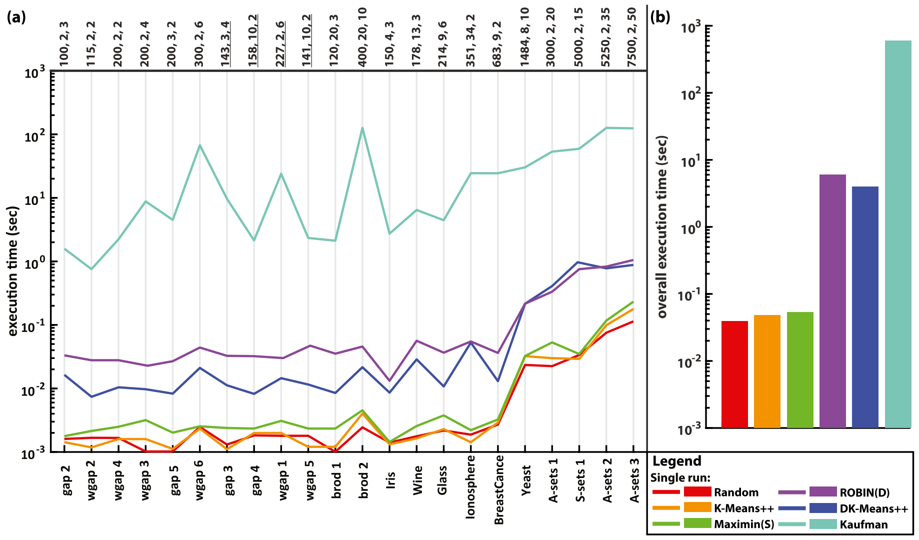


Fig. 7 Execution time analysis. **a** Each line shows the execution duration of an initialisation method on different data sets selected based on size, dimensions and number of clusters. We average the execution time for 50 repetitions and, in case of models, across the 40 data sets of each model. The data sets are arranged based on their *size*, *dimensionality* and *number of clusters* (see info on top, underlined numbers means that for these models generate data sets of different sizes). **b** Each bar shows the summed execution time across all data sets of **a**

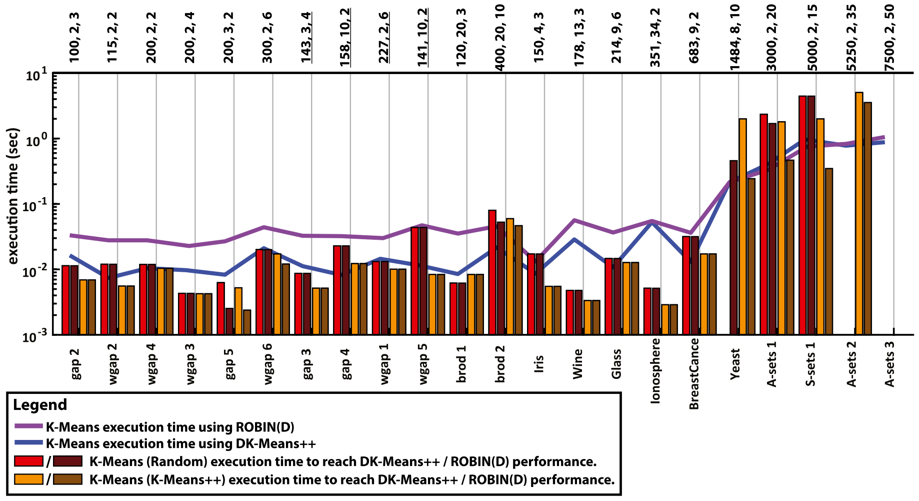


Fig. 8 Execution time until stochastic methods reach or surpass the performance of deterministic methods. Each bar shows the running time requirements of the clustering algorithm initialised with a stochastic method to reach or surpass the performance of the same algorithm initialised with a deterministic method. The time requirements of the clustering algorithm using a deterministic method are shown as lines for comparison. Cases where bars are not shown mean that, up to 50 runs, the clustering algorithm using a stochastic method was unable to surpass the performance of the deterministic method

Based on the results in Fig. 7 Kaufman is the worst method in terms of execution duration and it is affected both by the size, dimensionality and number of clusters. Random and K-Means++ are the fastest methods followed by Maximin(S). DK-Means++ is almost always better than ROBIN(D) in terms of speed for our implementation.

Furthermore, and based on the results of Table 6 we performed an analysis on the time requirements of the stochastic methods to reach or surpass the performance of deterministic methods with multiple executions of the clustering algorithm using different seeds. In Fig. 8 we show the single run execution time of the stochastic initialisation (plus the clustering overhead) multiplied by the number of iterations required to surpass the deterministic methods (see Table 6). Based on the results shown in Fig. 8 we observe that in many occasions running DK-Means++ once is better in terms of execution time than repeated runs of the clustering with a stochastic method. Equivalent conclusions can also be obtained from the Maximin(S) initialisation method (supplementary material).

We also investigated the number of iterations required for the clustering algorithm to reach convergence using stochastic and deterministic methods. We find that stochastic methods overall provide worst initial conditions (with the exception of ROBIN(S) vs Maximin(D)), and as a consequence the clustering algorithm requires more iterations to converge, which adds to the overhead of the stochastic initialisation methods (for results refer to the supplementary material).

5 Discussion

K-Means clustering remains one of the most common clustering techniques in many different research fields and frequently it is used as a component of more complex algorithms (e.g. hierarchical clustering Jain, 2010). Following similar benchmark studies on K-Means (Celebi et al., 2013; Fränti & Sieranoja, 2018, 2019), in this study we compare stochastic and deterministic initialisation methods on K-Means variations. We particularly investigated the methods of ROBIN and DK-Means++ since to the best of our knowledge they have not been studied as extensively as other initialisation methods. Experimentally we showed:

- More sophisticated initialisation methods can lead, on average, to better clustering regardless of the K-Means variation (see Table 4). From the stochastic methods, ROBIN(S) can achieve the best average performance compared with Random, K-Means++ and Maximin(S) (see Fig. 2). From the deterministic methods, DK-Means++ can achieve the best performance compared with Kaufman, Maximin(D) and ROBIN(D) (see Fig. 3). In addition, DK-Means++ can achieve better performance from the average performance of stochastic methods (see Fig. 4). Overall, deterministic methods have on average less performance variability across the data sets of each model we tested and lead to more stable solutions than stochastic methods (see supplementary material) and can surpass the performance of stochastic methods (see Fig. 4).
- When executed multiple times stochastic methods can achieve better performance than deterministic methods. Opposite to the first point, in that case, less sophisticated methods (such as Random and K-Means++ as opposed to ROBIN(S)) can achieve better performance (see Fig. 6). K-Means++ with 50 executions achieved the best performance followed by Random (see Fig. 5). The only deterministic method that can still

compete to an extent is DK-Means++ (see supplementary material where we provide a full list of all comparisons among all the initialisation methods).

- We found (see Table 5) that as indicated by Slonim et al. (2013) Hartigan–Wong K-Means is better than Lloyd's K-Means and as shown by Brusco et al. (2017) (only for one K-Means variant) K-Medians is better than both Hartigan–Wong and Lloyd's K-Means. However these differences add up to performance difference of only 1.5% as measured by the purity index.
- Regarding execution time requirements, Random and K-Means++ are fastest performers in terms of single runs while Kaufman the slowest (see Fig. 7). Maximin(S) is slightly slower than K-Means++. Nevertheless these methods require multiple executions in order to reach the performance of determinsitic methods (refer to Table 6) especially with bigger data sets (number of elements to thousands). Multiple executions of these methods have almost similar requirements as a single run of deterministic methods DK-Means++ and ROBIN(D) (refer to Fig. 8). This is due to the fact that the clustering algorithm requires more iterations to reach converge when stochastic methods are used (refer to the supplementary material). Between DK-Means++ and ROBIN(D) the former is faster than the latter.

Overall, and from a practical point of view, the stochastic Random and the deterministic Kaufman methods are not advisable. The first method despite being the simplest and the fastest can be replaced with K-Means++ that has better probability of achieving superior performance. The latter method is extremely slow and there are better alternatives such as the DK-Means++ that has both better performance and execution time. Maximin(D) and ROBIN(S) are not advisable either since the former is relatively fast and multiple executions of Maximin(S) can be performed instead while the latter has much more time requirements, small variability on its solutions and when an approximate clustering is required ROBIN(D) can be used instead. DK-Means++ is a good option when determinism is required since with a single run it can achieve better performance compared with other deterministic methods and comparable performance to multiple executions of stochastic methods that would require the same or more running time. In applications where exhausted search of optimal initial centroids needs to be performed K-Means++ should be considered (the study of Celebi et al. (2013) has also benchmarked a greedy version of this method which is also recommended). In these cases, if time requirements are flexible, a strategy would be to perform first DK-Means++ which would give an indication about the clustering capabilities of the data set and then multiple executions of K-Means++. We should add that in the mixed model 4, ROBIN(S) and ROBIN(D) performed significantly low compared with other cases because both where placing two initial centroids on the sides of the most elongated cluster while DK-Means++ were placing correctly a centroid almost in the middle of the cluster. This indicates that the DK-Means++'s heuristic might be more robust to applications than the LOF score of ROBIN for clusters detection. It should be noted that more complex techniques like DK-Means++ can be considered as clustering algorithms themselves since they produce good initial clusters. This observation was mentioned in the study of Celebi et al. (2013) and we also performed a study (refer to the supplementary material) on the number of iterations until the K-Means algorithm reaches convergence when it is initialized with different methods. Based on the results deterministic methods cause K-Means to converge faster than when it is initialized with stochastic methods. As expected, ROBIN(S) and DK-Means++ were again the best stochastic and deterministic methods on that account.

These conclusions were based on extensive benchmarking considering many different clustering models from other studies: Gaussian, high-dimensional (10 dimensions), elongated, unbalanced, non-Gaussian from the studies of Tibshirani et al. (2001) and Yan and Ye (2007); high-dimensional (20 dimensions) containing informative and uninformative features and higher-dimensional (1000 and 1500 dimensions) with varying number of clusters (3, 10, 50 clusters) and cluster sizes (50–100 points) Brodinová et al. (2017). We also used our own models which contain clusters with different properties (unbalanced, elongated and Gaussian; unbalanced Gaussian and non-Gaussian; unbalanced, Gaussian with different variability among their dimensions).

With the use of synthetic data set generators we had the ability to generate multiple data sets and run hypothesis testing to further strengthen our conclusions but we also considered standalone data sets. The "clustering data sets" S-sets (Fränti & Virmajoki, 2006) and A-sets (Kärkkäinen & Fränti, 2002) were selected from the studies of Fränti and Sieranoja (2018), Fränti and Sieranoja (2019) because both are containing more clusters and data points than the generated ones and also because in the case of the S-sets the clusters are having different overlap degrees. The conclusions we obtained from the data set generators match with the conclusions of the standalone S-sets and A-sets data sets. Specifically for our higher dimensional data sets (1000, 1500 dimensions) generated using the Brodinova generator (Brodinová et al., 2017) (see Table 2) we selected to have small clusters due to the Kaufman initialization method which requires significant amount of time to be executed. However, we also generated data sets with larger clusters (approximately five times bigger) and we tested the ROBIN(D) and DK-Means++ methods on them. The results (not shown) and conclusions were similar to the ones reported already.

Based on the previous studies (Fränti & Sieranoja, 2018, 2019) the authors have clearly demonstrated that K-Means performs worse when there is large number of clusters and that dimensionality does not have a direct effect on the performance of the algorithm. In our experiments using the Brodinova models (see Figs. 3, 6 brod 1 to brod 12) we observe that indeed the performance of all the methods drops when the number of clusters is increased regardless of the dimensionality, especially in the case of Brodinova brod10 model where we generate data sets having 50 clusters. Apart from the last extreme case, we observed that multiple executions of stochastic methods improve the performance of K-Means. It should also be noted that the deterministic DK-Means++ method achieves (similar to multiple executions of stochastic methods i.e. Random, K-Means++, Maximin(S) and ROBIN(S)) the highest performance on the clustering basic benchmark (Fränti & Sieranoja, 2018, 2019) in all the cases (see the supplementary material) even though these data sets have high number of clusters (A-sets: 20, 30, 50; S-sets: 15). The same authors (Fränti & Sieranoja, 2018, 2019) also demonstrated that strong cluster unbalances affect negatively the K-Means clustering. In our experiments and specifically for the weighted gap 2 model we observed that data sets with unbalanced clusters do not cause any particular issues to the maximum performances of the algorithms. For the performance between K-Means and K-Medians, similar to the results of Brusco et al. (2017), we found that K-Medians outperforms K-Means on synthetic data set models but on a small difference of 1% of purity and on standalone data sets (both synthetic and real-world) any particular differences among the K-Means variations couldn't be clearly detected.

In order to show application to "real world problems" previous studies have chosen to use standard classification data sets as benchmarks for clustering. While this approach is commonly used, in these data the mapping from classes to clusters is somehow forced: it is possible that data from one class belong to different clusters, and assuming that number of clusters equals number of classes is likely to underestimate the true number of clusters.

This can be seen from the low value of the Silhouette index especially in the cases of Ionosphere and Yeast data sets. For this we base our conclusions mostly on the benchmark models that allows us to generate multiple samples and evaluate the statistical significance of the results. In fact, we considered a broad combination of different clusters, in terms of normality (Gaussian, non-Gaussian), shape (spherical, elongated) and size (clusters with different number of data points) including high dimensional data, as found in real world applications such as bioinformatics (Wang et al., 2008).

It should also be noted that many clustering frameworks designed to deal with complex data sets (e.g. sub-clustering Biswas & Jacobs, 2014, or sparse clustering Witten & Tibshirani, 2010; Kondo et al., 2016; Brodinová et al., 2017) are using the K-Means or some variant of it and are dependent on good clustering initialisation. Our experimental work revealed that there are deterministic methods (DK-Means++ Nidheesh et al., 2017) that lead to a good clustering solution with a single execution of the K-Means algorithm.

A limitation of the current study is that the execution time analysis is subject to the machine that executed it. More powerful machines or code optimisation of the algorithms and initialisation methods can change time analysis results. Nevertheless the rest of the analysis including the number of different seeds for stochastic methods to reach the performance of deterministic is, on average, reproducible. Statistics on average performance comparison are representative since, the analysis of Sects. 4.1–4.3 had been also performed on 25 instances of the various data sets models instead of 50 and led to the same conclusions.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10994-021-06021-7.

Acknowledgements This research was funded by the Numerical Algorithms Group (NAG). We would also like to thank the reviewers for their detailed and constructive comments especially on the execution time considerations and on the inclusion of both a supervised and unsupervised benchmarking comparison.

Authors contribution AV performed the experiments and wrote the manuscript. EV contributed to the statistical and algorithmic analysis, provided feedback and edited the manuscript. SL oversaw the benchmarking process, provided feedback on the results presentation and the manuscript. MC contributed to the literature review and the discussion and performed the final proofreading.

References

Al Hasan, M., Chaoji, V., Salem, S., & Zaki, M. J. (2009). Robust partitional clustering by outlier and density insensitive seeding. *Pattern Recognition Letters, 30*(11), 994–1002.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for industrial and applied mathematics* (pp. 1027–1035)

Asuncion, A., & Newman, D. (2007). Uci machine learning repository.

Bilenko, M., Basu, S., & Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning* (p 11). ACM.

Biswas, A., & Jacobs, D. (2014). Active subclustering. *Computer Vision and Image Understanding, 125,* 72–84.

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). Lof: identifying density-based local outliers. *ACM Sigmod Record, 29,* 93–104.

Brodinová, Š., Filzmoser, P., Ortner, T., Breiteneder, C., & Rohm, M. (2017). Robust and sparse k-means clustering for high-dimensional data. *Advances in Data Analysis and Classification*, 1–28.

Brusco, M. J., Shireman, E., & Steinley, D. (2017). A comparison of latent class, k-means, and k-median methods for clustering dichotomous data. *Psychological Methods, 22*(3), 563.

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications, 40*(1), 200–210.

Charu, C. A., & Chandan, K. R. (2013). Data clustering: Algorithms and applications.

Feldman, D., & Schulman, L. J. (2012). Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on discrete algorithms. Society for industrial and applied mathematics* (pp. 1343–1354).

Fränti, P., & Sieranoja, S. (2018). K-means properties on six clustering benchmark datasets. *Applied Intelligence, 48*(12), 4743–4759.

Fränti, P., & Sieranoja, S. (2019). How much can k-means be improved by using better initialization and repeats? *Pattern Recognition, 93,* 95–112.

Fränti, P., & Virmajoki, O. (2006). Iterative shrinking method for clustering problems. *Pattern Recognition, 39*(5), 761–765. https://doi.org/10.1016/j.patcog.2005.09.012.

Gehring, T. V., Luksys, G., Sandi, C., & Vasilaki, E. (2015). Detailed classification of swimming paths in the morris water maze: Multiple strategies within one trial. *Scientific Reports, 5,* 14562.

Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science, 38,* 293–306.

Hahsler, M., Piekenbrock, M., Arya, S., & Mount, D. (2019). dbscan: Density based clustering of applications with noise (DBSCAN) and related algorithms. https://github.com/mhahsler/dbscan.

Hartigan, J. A. (1975). Clustering algorithms.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society Series C (Applied Statistics), 28*(1), 100–108.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters, 31*(8), 651–666.

Jancey, R. (1966). Multidimensional group analysis. *Australian Journal of Botany, 14*(1), 127–130.

Kärkkäinen, I., & Fränti, P. (2002). Dynamic local search algorithm for the clustering problem. Technical Report. A-2002-6, Department of Computer Science, University of Joensuu, Joensuu, Finland.

Katsavounidis, I., Kuo, C. C. J., & Zhang, Z. (1994). A new initialization technique for generalized lloyd iteration. *IEEE Signal Processing Letters, 1*(10), 144–146.

Kaufman, L., & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis,* (Vol. 344). John Wiley & Sons.

Kondo, Y., Salibian-Barrera, M., & Zamar, R. (2016). Rskc: An r package for a robust and sparse k-means clustering algorithm. *Journal of Statistical Software, 72*(5), 1–26.

Lan, X., Li, Q., & Zheng, Y. (2015). Density k-means: A new algorithm for centers initialization for k-means. In *2015 6th IEEE international conference on software engineering and service science (ICSESS)* (pp. 958–961). IEEE.

Lopuhaa, H. P., & Rousseeuw, P. J. (1991). Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *The Annals of Statistics, 19*(1), 229–248.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). Oakland, CA, USA

MATLAB. (2019). version 9.6.0 (R2019a). The MathWorks Inc., Natick, Massachusetts

Moret, B. M., & Shapiro, H. D. (1992). An empirical assessment of algorithms for constructing a minimum spanning tree. *Computational Support for Discrete Mathematics, 15,* 99–117.

Nidheesh, N., Nazeer, K. A., & Ameer, P. (2017). An enhanced deterministic k-means clustering algorithm for cancer subtype prediction from gene expression data. *Computers in Biology and Medicine, 91,* 213–221.

Numerical Algorithms Group (NAG). (2019). The NAG Toolbox for MATLAB®. https://www.nag.com/

Pena, J. M., Lozano, J. A., & Larranaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters, 20*(10), 1027–1040.

R Core Team. (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org/

Rendón, E., Abundez, I., Arizmendi, A., & Quiroz, E. M. (2011). Internal versus external cluster validation indexes. *International Journal of Computers and Communications, 5*(1), 27–34.

Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science, 344*(6191), 1492–1496.

van Rossum, G. (1995). Python tutorial. Technical Report CS-R9526. Centrum voor Wiskunde en Informatica (CWI), Amsterdam.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20,* 53–65.

Slonim, N., Aharoni, E., & Crammer, K. (2013). Hartigan's k-means versus lloyd's k-means-is it time for a change? In *IJCAI* (pp. 1677–1684).

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63*(2), 411–423.

Vouros, A., & Vasilaki, E. (2021). A semi-supervised sparse k-means algorithm. *Pattern Recognition Letters, 142,* 65–71.

Wang, Y., Miller, D., & Clarke, R. (2008). Approaches to working in high-dimensional data spaces: gene expression microarrays. *British Journal of Cancer, 98*(6), 1023.

Whelan, C., Harrell, G., & Wang, J. (2015). Understanding the k-medians problem. In *Proceedings of the international conference on scientific computing (CSC). The Steering Committee of The World Congress in Computer Science, Computer* (p. 219).

Witten, D. M., & Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistical Association, 105*(490), 713–726.

Yan, M., & Ye, K. (2007). Determining the number of clusters using the weighted gap statistic. *Biometrics, 63*(4), 1031–1037.