Noname manuscript No.
(will be inserted by the editor)

An efficient K-means clustering algorithm for tall data

Marco Capó · Aritz Pérez · Jose A. Lozano

Received: date / Accepted: date

Abstract The analysis of continously larger datasets is a task of major importance in a wide variety of scientific fields. Therefore, the development of efficient and parallel algorithms to perform such an analysis is a a crucial topic in unsupervised learning. Cluster analysis algorithms are a key element of exploratory data analysis and, among them, the K-means algorithm stands out as the most popular approach due to its easiness in the implementation, straightforward parallelizability and relatively low computational cost. Unfortunately, the K-means algorithm also has some drawbacks that have been extensively studied, such as its high dependency on the initial conditions, as well as to the fact that it might not scale well on massive datasets. In this article, we propose a recursive and parallel approximation to the K-means algorithm that scales well on the number of instances of the problem, without affecting the quality of the approximation. In order to achieve this, instead of analyzing the entire dataset, we work on small weighted sets of representative points that are distributed in such a way that more importance is given to those regions where it is harder to determine the correct cluster assignment of the original instances. In addition to different theoretical properties, which explain the reasoning behind the algorithm, experimental results indicate that our method outperforms the state-of-the-art in terms of the trade-off between number of distance computations and the quality of the solution obtained.

Keywords K-means problem · Lloyd's algorithm · K-means++ · Coresets · Unsupervised learning

Marco Capó
Basque Center of Applied Mathematics, Bilbao 48009, Spain
Tel.: +34 946 567 842
E-mail: mcapo@bcamath.org

Aritz Pérez
Basque Center of Applied Mathematics, Bilbao 48009, Spain

Jose A. Lozano
Intelligent Systems Group, Department of Computer Science and Artifitial Intelligence, University of the Basque Country UPV/EHU, San Sebastián 20018, Spain

1 Introduction

Partitional clustering is an unsupervised data analysis technique that intends to unveil the inherent structure of a set of points by partitioning it into a number of disjoint groups, called clusters. This is done in such a way that intra-cluster similarity is high and the inter-cluster similarity is low. Furthermore, clustering is a basic task in many areas, such as artificial intelligence, machine learning and pattern recognition (Jain and Dubes, 1988; Jain et al., 1999; Kanungo et al., 2002a). Even when there exists a wide variety of clustering methods, the K-means algorithm remains as one of the most popular (Berkhin et al., 2006; Jain, 2010). In fact, it has been identified as one of the top 10 algorithms in data mining (Wu et al., 2008).

1.1 K-means Problem

Given a set of n data points (instances) $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in $\mathbb{R}^d$ and an integer K, the **K-means problem** is to determine a set of K centroids $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ in $\mathbb{R}^d$, so as to minimize the following error function:

$$E^D(C) = \sum_{\mathbf{x} \in D} \|\mathbf{x} - \mathbf{c}_\mathbf{x}\|^2, \text{ where } \mathbf{c}_\mathbf{x} = \arg\min_{\mathbf{c} \in C} \|\mathbf{x} - \mathbf{c}\|^2 \tag{1}$$

This is a combinatorial optimization problem, since it is equivalent to finding the partition of the n instances in K groups whose associated set of centers of mass minimizes Eq.1. The number of possible partitions is a Stirling number of the second kind, $S(n, K) = \frac{1}{K!} \sum_{j=0}^{K} (-1)^{K-j} \binom{K}{j} j^n$ (Äyrämö and Kärkkäinen, 2006).

Since finding the globally optimal partition is known to be NP-hard (Aloise et al., 2009), even for instances in the plane (Mahajan et al., 2009), and exhaustive search methods are not useful under this setting, iterative refinement based algorithms are commonly used to approximate the solution of the K-means and similar problems (Äyrämö and Kärkkäinen, 2006; Kaufman and Rousseeuw, 1987; Lloyd, 1982). These algorithms iteratively relocate the data points between clusters until a locally optimal partition is attained. Among these methods, the most popular is the **K-means algorithm** (Jain, 2010; Lloyd, 1982).

1.2 K-means Algorithm

The K-means algorithm is an iterative refinement method that consists of two stages: Initialization, in which we set the starting set of K centroids, and an iterative stage, called **Lloyd's algorithm** (Lloyd, 1982). In the first step of Lloyd's algorithm, each instance is assigned to its closest centroid (assignment step), then the set of centroids is updated as the centers of mass of the instances assigned to the same centroid in the previous step (update step). Finally, a stopping criterion is verified. The most common criterion implies the computation of the error function (Eq.1): If the error does not change significantly with respect to the previous iteration, the algorithm stops (Manning et al., 2008), i.e., if C and C' are the set of centroids obtained at consecutive Lloyd's iterations, then the algorithm stops when

$$|E^D(C) - E^D(C')| \leq \varepsilon, \text{ for a fixed threshold } \varepsilon \ll 1. \tag{2}$$

Conveniently, every step of the K-means algorithm can be easily parallelized (Zhao et al., 2009), which is a major key to meet the scalability of the algorithm (Wu et al., 2008).

The time needed for the assignment step is $\mathcal{O}(n \cdot K \cdot d)$, while updating the set of centroids requires $\mathcal{O}(n \cdot d)$ computations and the stopping criterion, based on the computation of the error function, is $\mathcal{O}(n \cdot d)$. Hence, the assignment step is the most computationally demanding and this is due to the number of distance computations that needs to be done at this step. In fact, the computational costs of different variants to the K-means algorithm is usually compared in terms of the number of distances computed (Bachem et al., 2016; Capó et al., 2017; Elkan, 2003). Taking this into account, the main objective of our proposal is to define a variant of the K-means algorithm that controls the trade-off between the number of distance computations and the quality of the solution, oriented to problems with high volumes of data. Lately, this problem has gained special attention due to the exponential increase of the data volumes that scientists, from different backgrounds, face on a daily basis (Jordan, 2013).

1.2.1 Common initializations

As it is widely reported in the literature, the performance of Lloyd's algorithm highly depends upon the initialization stage in terms of the quality of the solution obtained and the number of Lloyd's iterations (Peña et al., 1999). A poor initialization, for instance, could lead to an exponential number of Lloyd's iterations, with respect to the number of instances, in the worst case scenario (Vattani, 2011).

Ideally, the selected seeding/initialization strategy should deal with different problems, such as outlier detection and cluster oversampling. A lot of research has been done on this topic: A detailed review of seeding strategies can be found in Redmond and Heneghan (2007); Steinley and Brusco (2007).

The standard initialization procedure consists of performing several re-initializati- ons via Forgy's method (Forgy, 1965) and keeping the set of centroids with the smallest error (Redmond and Heneghan, 2007; Steinley and Brusco, 2007). Forgy's technique defines the initial set of centroids as K instances selected uniformly at random from the dataset. The intuition behind this approach is that, by choosing the centroids uniformly at random, we are more likely to choose a point near an optimal cluster center, since such points tend to be where the highest density regions are located. The main disadvantage of this approach is that there is no guarantee that two, or more, of the selected seeds will not be near the center of the same cluster, especially when dealing with unbalanced clusters (Redmond and Heneghan, 2007).

More recently, different probabilistic based seeding techniques have been developed and, due to their simplicity and strong theoretical guarantees, they have become quite popular. Among these, the most relevant is the *K-means++* algorithm proposed by Arthur and Vassilvitskii (2007). K-means++ selects only the first centroid uniformly at random from the dataset. Each subsequent initial centroid is chosen with a probability proportional to the distance with respect to the previously selected set of centroids. The key idea of this cluster initialization technique is to preserve the diversity of seeds while being robust to outliers. The K-means++ algorithm leads to a $\mathcal{O}(\log K)$ factor approximation [1] of the optimal error after the initialization (Arthur and Vassilvitskii, 2007). The main drawbacks of this approach refer to its sequential nature, which hinders its parallelization, as well as to the fact that it requires K full scans of the entire dataset, which leads to a complexity of $\mathcal{O}(n \cdot K \cdot d)$.

In order to alleviate such drawbacks, different variants of K-means++ have been studied. In particular, in Bahmani et al. (2012), a parallel K-means++ type algorithm is presented. This parallel variant achieves a constant factor approximation to the optimal solution after a logarithmic number of passes over the dataset. Furthermore, in Bachem et al. (2016), an approximation to K-means++ with a sublinear time complexity with respect to the number of data points, is proposed. Such an approximation is obtained via a Markov Chain Monte Carlo sampling approximation of the K-means++ probability function. The proposed algorithm generates solutions of similar quality to those of K-means++, at a fraction of its cost.

1.2.2 Reducing the time complexity of Lloyd's algorithm

Regardless of the initialization, a large amount of work has also been done on reducing the overall computational complexity of Lloyd's algorithm. Mainly, two approaches can be distinguished:

- **The use of distance pruning techniques**: Lloyd's algorithm can be accelerated by avoiding unnecessary distance calculations, i.e., when it can be verified in advance that no cluster re-assignment is possible for a certain instance. As presented in Ding et al. (2015); Drake and Hamerly (2012); Elkan (2003); Hamerly (2010), this can be done with the construction of different pairwise distance bounds between the set of points and centroids and additional information, such as the displacement of every centroid after

[1] Algorithm A is an α factor approximation of the K-means problem, if $E^D(C') \leq \alpha \cdot \min_{C \subseteq \mathbb{R}^d, |C|=K} E^D(C)$, for any output C' of A.

a Lloyd's iteration. In particular, in Hamerly (2010), reductions of over 80% of the amount of distance computations are observed.

– **Applying Lloyd's algorithm over a smaller (weighted) set of points**: As previously commented, one of the main drawbacks of Lloyd's algorithm is that its complexity is proportional to the size of the dataset, meaning that it may not scale well for massive data applications. One way of dealing with this is to apply the algorithm over a smaller set of points rather than over the entire dataset. Such smaller sets of points are commonly extracted in two different ways:

  – *Via dataset sampling*: In Bottou and Bengio (1995); Bradley and Fayyad (1998); Davidson and Satyanarayana (2003); Sculley (2010), different statistical techniques are used with the same purpose of reducing the size of the dataset. Among these algorithms, we have the *Mini-batch K-means* proposed by Sculley in Sculley (2010). Mini-batch K-means is a very popular scalable variant of Lloyd's algorithm that proceeds as follows: Given an initial set of centroids obtained via Forgy's algorithm, at every iteration, a small fixed amount of samples is selected uniformly at random and assigned to their corresponding cluster. Afterwards, the cluster centroids are updated as the average of all samples ever assigned to them. This process continues until convergence. Empirical results, in a range of large web based applications, corroborate that a substantial saving of computational time can be obtained at the expense of some loss of cluster quality (Sculley, 2010). Moreover, very recently, in Newling and Fleuret (2016), an accelerated Mini-batch K-means algorithm, via the distance pruning approach of Elkan (2003), was presented.

  – *Via dataset partition*: The reduction of the dataset can also be generated as sets of representatives induced by partitions of the dataset. In particular, there have been a number of recent papers that describe $(1+\varepsilon)$-factor approximation algorithms and/or (K,ε)-coresets [2] for the K-means problem (Har-Peled and Mazumdar, 2004; Kumar et al., 2004; Matoušek, 2000). However, these variants are commonly exponential in K and/or ε^{-1}, and so, they might not be viable in practice (Arthur and Vassilvitskii, 2007). In the literature, other coreset constructions via sampling have been proposed as can be seen in Bachem et al. (2018); Balcan et al. (2013); Feldman et al. (2007); Lucic et al. (2016). Moreover, Kanungo et al. (2002b) also developed a $(9+\varepsilon)$-approximated algorithm for the K-means problem that is $\mathcal{O}(n^3 \cdot \varepsilon^{-d})$, thus it is not useful for massive data applications. In particular, for this kind of applications,

[2] A weighted set of points W is a (K, ε)-coreset if, for all set of centroids C, $|F^W(C) - E^D(C)| \leq \varepsilon \cdot E^D(C)$, where $F^W(C) = \sum_{\mathbf{y}\in W} w(\mathbf{y}) \cdot \|\mathbf{y} - \mathbf{c}_{\mathbf{y}}\|^2$ and $w(\mathbf{y})$ is the weight associated to a representative $\mathbf{y} \in W$.

another approach has been very recently proposed in Capó et al. (2017): The *Recursive Partition based K-means algorithm* (Algorithm 1).

Recursive Partition based K-means algorithm

Unlike the common coreset approach, the Recursive Partition based K-means algorithm (**RPKM**) approximates the solution of the K-means problem by recursively applying the weighted version of Lloyd's algorithm over a sequence of spatial-based thinner partitions of the dataset [3], rather than over a predefined weighted set of points.

Definition 1 (Dataset partition induced by a spatial partition) Given a dataset D and a spatial partition $\mathcal{B}$ of its smallest bounding box, the partition of the dataset D induced by $\mathcal{B}$ is defined as $\mathcal{P} = \mathcal{B}(D)$, where $\mathcal{B}(D) = \{B(D)\}_{B\in\mathcal{B}}$ and $B(D) = \{\ \mathbf{x} \in D : \mathbf{x} \text{ lies on } B \in \mathcal{B}\}$ [4].

Applying the weighted version of K-means algorithm over the dataset partition $\mathcal{P}$, consists of executing Lloyd's algorithm (Section 1.2) over the set of centers of mass (**representatives**) of $\mathcal{P}$, $\overline{P}$ for all $P \in \mathcal{P}$, considering their corresponding cardinality (**weight**), $|P|$, when updating the set of centroids. This means that we seek to minimize the weighted error function $E^{\mathcal{P}}(C) = \sum\limits_{P\in\mathcal{P}} |P| \cdot \|\overline{P} - \mathbf{c}_{\overline{P}}\|^2$, where $\mathbf{c}_{\overline{P}} = \arg\min_{\mathbf{c}\in C} \|\overline{P} - \mathbf{c}\|$. Afterwards, the same process is repeated over a thinner partition $\mathcal{P}'$ of the dataset, using as initialization the set of centroids obtained for $\mathcal{P}$.

Algorithm 1: RPKM algorithm

```
Input: Dataset $D$ and number of clusters $K$.
Output: Set of centroids $C$.
Step 1: Construct an initial partition of $D$, $\mathcal{P}$, and define an
initial set of $K$ centroids, $C$.
Step 2: $C =$ WeightedLloyd($\mathcal{P}, C, K$).
while not Stopping Criterion do
    Step 3: Construct a dataset partition $\mathcal{P}'$, thinner than $\mathcal{P}$.
    Set $\mathcal{P} = \mathcal{P}'$.
    Step 4: $C =$ WeightedLloyd($\mathcal{P}, C, K$).
end
return $C$
```

In general, the RPKM algorithm can be divided into three tasks: The construction of an initial partition of the dataset and set of centroids (`Step 1`), the update of the corresponding set of centroids via weighted

[3] A partition of the dataset $\mathcal{P}'$ is thinner than $\mathcal{P}$, if each subset of $\mathcal{P}$ can be written as the union of subsets of $\mathcal{P}'$.

[4] From now on, we will refer to each $B \in \mathcal{B}$ as a **block** of the spatial partition $\mathcal{B}$.

Lloyd's algorithm (`Step 2` and `Step 4`) and the construction of the sequence of thinner partitions (`Step 3`). At this point it must be highlighted that experimental results have shown the reduction of several orders of computations for RPKM with respect to both K-means++ and Mini-batch K-means, while obtaining competitive approximations to the solution of the K-means problem on low dimensional datasets (Capó et al., 2017).

1.3 Motivation and contribution

The experimental results presented in Capó et al. (2017) refer to a RPKM variant called **grid based RPKM**. In the case of the grid based RPKM, the initial spatial partition is defined by the grid obtained after dividing each side of the smallest bounding box of D by half, i.e., a grid with 2^d equally sized blocks. In the same fashion, at the i-th grid based RPKM iteration, the corresponding spatial partition is updated by dividing each of its blocks into 2^d new blocks, i.e., $\mathcal{P}$ can have up to $2^{i\cdot d}$ representatives (see Fig.1 in Capó et al. (2017)). It can be shown that this approach produces a (K,ε)-coreset with ε descending exponentially with respect to the number of iterations [5]. Taking this into consideration, three main problems arise for the grid based RPKM:

- **Problem 1.** *It is independent of the dataset D*: As we mentioned before, regardless of the analyzed dataset D, the sequence of partitions of the grid based RPKM is induced by an equally sized spatial partition of the smallest bounding box containing D. In this sense, the induced partition does not consider features of the dataset, such as its density, to construct the sequence of partitions: A large amount of computational resources might be spent on regions whose misclassification does not add a significant error to our approximation. Moreover, the construction of every partition of the sequence has a $O(n \cdot d)$ cost, which is expensive for massive data applications, as n is huge.
- **Problem 2.** *It is independent of the problem*: The partition strategy of the grid based RPKM does not explicitly consider the optimization problem that K-means seeks to minimize. Instead, it offers a simple/inefficient way of generating a sequence of spatial thinner partitions. The reader should note that each block of the spatial partition can be seen as a restriction over the K-means optimization problem, which enforces all the instances contained in it to belong to the same cluster. Therefore, it is of our interest to design smarter spatial partitions that focus most of the computational resources on those regions where the correct cluster affiliation is not clear, i.e., around the cluster boundaries associated to a given approximation to the solution of the K-means problem.

[5] See Theorem 6 in Appendix B.

– **Problem 3.** *It does not scale well on the dimension d*: Even when our target is related to tall data applications[6], it must be remarked that for a relatively low number of iterations, $i \simeq \log_2(n)/d$, and/or dimensionality $d \simeq \log_2(n)$, applying this RPKM version can be similar to applying Lloyd's algorithm over the entire dataset, i.e., no reduction of distance computations might be observed, as $|\mathcal{P}| \simeq n$. In fact, for the experimental section in Capó et al. (2017), $d, i \leq 10$.

In any case, independently of the partition strategy, RPKM algorithm offers some interesting properties such as the no clustering repetition. This is, none of the obtained groupings of the n instances into K groups can be repeated at the current RPKM iteration or for any thinner partition than the current one. This is an useful property since it can be guaranteed that the algorithm discards many possible clusterings at each RPKM iteration using a much reduced set of points than the entire dataset. Furthermore, this fact enforces the decrease of the maximum number of Lloyd iterations that we can have for a given partition. In practice, it is also common to observe a monotone decrease of the error for the entire dataset (Capó et al., 2017).

Bearing all these facts in mind, we propose a RPKM type approach called the *Boundary Weighted K-means* algorithm (**BWKM**). The idea behind it is to prioritize the use of resources on the cluster boundaries of our weighted approximation, which are constituted by those blocks that may not be well assigned.

Definition 2 (Well assigned blocks) Let C be a set of centroids and D be a given dataset. We say that a block B is well assigned, with respect to C and D, if every point $\mathbf{x} \in B(D)$ is assigned to the same centroid $c \in C$.

The notion of well assigned blocks is of our interest as RPKM associates all the instances contained in a certain block to the same cluster, which corresponds to the one that its center of mass belongs to. Hence, our goal is to divide those blocks that are not well assigned. In this sense, our proposal intends to approximate better those regions of the space where the correct cluster affiliation is not clear.

At this point, it should be remarked that not only fixing the set of representives around the cluster boundaries allows us to save a large amount of computational resources, but also lets us deduce different theoretical properties in terms of the weighted K-means error function that we are minimizing. Among other properties, that we discuss in Section 2, it can be observed that if all the instances in a set of points, P, are correctly assigned for two sets of centroids, C and C', then the difference between the error of both sets of centroids is equivalent to the difference of their weighted error, i.e., $E^P(C) - E^P(C') = E^{\{P\}}(C) - E^{\{P\}}(C')$ [7]. If this occurs for each subset of a

[6]Data sets with an enormous number of instances and low number of dimensions.

[7] See Lemma 1 in Appendix B.

dataset partition, $\mathcal{P}$, and the centroids are generated after consecutive weighted K-means iterations, then we retain some properties of the K-means algorithm when applied over the entire dataset: i) We can guarantee a monotone decrease of the error for the entire dataset [8], ii) We can compute the reduction of the error for the newly obtained set of centroids, without computing the error function for the entire dataset, as in this case $E^D(C) - E^D(C') = E^{\mathcal{P}}(C) - E^{\mathcal{P}}(C')$ and iii) If every block contains instances belonging to the same cluster, then the solution obtained by our weighted approximation is actually a local optima of Eq.1 [9].

In particular, our proposal can be mainly divided into three tasks:

- **Task 1**: Design of a partition criterion that decides whether or not to divide a certain block, using only information obtained from the weighted Lloyd's algorithm.
- **Task 2**: Construct an initial partition of the dataset with a predefined number of blocks, which are mostly placed on the cluster boundaries.
- **Task 3**: Once a certain block is decided to be cut, guarantee a low increase on the number of representatives without affecting, if possible, the quality of the approximation. In particular, we propose a criterion that, in the worst case, has a linear growth in the number of representatives after an iteration.

The goal of **Task 1** and **Task 2** is to generate partitions of the dataset that, ideally, contain well assigned subsets, i.e., all the instances contained in a certain subset of the partition belong to the same cluster (**Problem 1** and **Problem 2**). As we previously commented, this fact implies additional theoretical properties in terms of the quality of our approximation. On the other hand, observe that both **Task 2** and **Task 3** alleviate the scalability of the algorithm with respect to the dimensionality of the problem, d (**Problem 3**).

The rest of this article is organized as follows: In Section 2, we describe the proposed algorithm, introduce some notation and discuss some theoretical properties of our proposal. In Section 3, we present a set of experiments in which we analyze the effect of different factors, such as the size of the dataset and the dimension of the instances over the performance of our algorithm. Additionally we compare these results with the ones obtained by the state-of-the-art. Finally, in Section 4, we define the next steps and possible improvements to our current work.

2 BWKM algorithm

In this section, we present the Boundary Weighted K-means algorithm. As we already commented, BWKM is a scalable improvement of the grid based

[8] See Theorem 2 in Appendix B.

[9] See Theorem 3 in Appendix B.

RPKM algorithm [10], that generates competitive approximations to the K-means problem, while reducing the amount of computations that the state-of-the-art algorithms require for the same task. BWKM reuses all the information generated at each weighted Lloyd run to construct a sequence of thinner partitions that solves **Problem 1**, **Problem 2** and alleviates **Problem 3**.

Our new approach makes major changes in all the steps in Algorithm 1 except in `Step 2` and `Step 4`. In these steps, the weighted version of Lloyd's algorithm is applied over the set of representatives and weights of the current dataset partition, $\mathcal{P}$. This process has a $O(|\mathcal{P}| \cdot K \cdot d)$ cost, hence it is of our interest to control the growth of $|\mathcal{P}|$, which is highlighted in both **Task 2** and **Task 3**.

In the following sections, we will describe in detail each step of BWKM. In Section 2.1, Section 2.2 and Section 2.3 we elaborate on **Task 1**, **Task 2** and **Task 3**, respectively.

2.1 A cheap criterion for detecting well assigned blocks

BWKM tries to efficiently determine the set of well assigned blocks in order to update the dataset partition. In order to do so, we firstly introduce a criterion, the misassignment function, that will help us verify this using mostly information generated in advanced by our weighted approximation:

Definition 3 Given a set of K centroids, C, a set of points $D \subseteq \mathbb{R}^d$ and $P = B(D) \neq \emptyset$ the subset of points contained in a block B. We define the misassignment function for B, given C and D, as:

$$\epsilon_{C,D}(B) = \max\{0, 2 \cdot l_B - \delta_P(C)\}, \tag{3}$$

where $\delta_P(C) = \min_{\mathbf{c} \in C \setminus \mathbf{c}_{\overline{P}}} \|\overline{P} - \mathbf{c}\| - \|\overline{P} - \mathbf{c}_{\overline{P}}\|$ and l_B is the length of the diagonal of B. In the case $P = B(D) = \emptyset$, we set $\epsilon_{C,D}(B) = 0$.

In the following result, Theorem 1, we show that if the misassignment function of a block is zero, then the block is well assigned. Otherwise, the block may not be well assigned. Taking this into account and, in order to control the size of the spatial partition and the number of distance computations, BWKM generates the sequence of spatial partitons by splitting only blocks from the boundary:

Definition 4 Let D be a dataset, C be a set of K centroids and $\mathcal{B}$ be a spatial partition. We define the boundary of $\mathcal{B}$, given C and D, as

$$\mathcal{F}_{C,D}(\mathcal{B}) = \{B \in \mathcal{B} : \epsilon_{C,D}(B) > 0\} \tag{4}$$

[10] From now on, we assume each block $B \in \mathcal{B}$ to be a hyperrectangle aligned with the coordinate axes.

Theorem 1 *Given a set of K centroids, C, a dataset, $D \subseteq \mathbb{R}^d$, and a block B, if $\epsilon_{C,D}(B) = 0$, then $\boldsymbol{c_x} = \boldsymbol{c}_{\overline{P}}$ for all $\boldsymbol{x} \in P = B(D) \neq \emptyset$.*[11]

Even though the condition in Theorem 1 is a sufficient condition, we will use the following heuristic rule in the construction of both the initial and the sequence of thinner partitions: The larger the misassignment function of a certain block is, then the more likely it is to contain instances with different cluster memberships.

It should be remarked that Theorem 1 offers an efficient and effective way of verifying that all the instances contained in a block B belong to the same cluster, using only information related to the structure of B and the set of centroids, C, as can be seen in Fig.1. In particular, we do not require any information associated to the individual instances in the dataset, $\mathbf{x} \in P$, but instead the criterion just uses some distance computations with respect to the representative of P, $\overline{P}$, that are already obtained from the weighted Lloyd's algorithm: In particular, in the example presented in Fig.1, we only set two cluster centroids (blue stars) and the representative of the instances in the block, $\overline{P}$, given by the purple diamond. In order to compute the misassignment function of the block, we require the length of the **three** segments: Distance between the representative with respect to its two closest centroids in C (blue dotted lines) and the diagonal of the block (purple dotted line). If the misassignment function is zero, then we know that all the instances contained in the block belong to the same cluster. Observe that, in this example, there are instances in both clusters, then the block is included in the boundary.

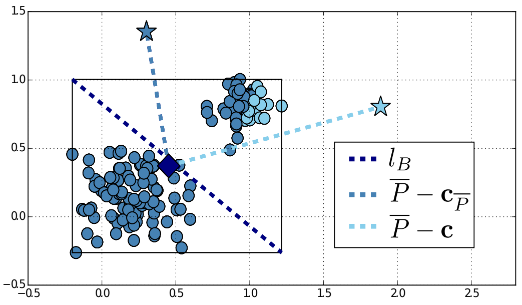

Fig. 1: Information required for computing the misassignment function of the block B, $\epsilon_{C,D}(B)$, for $K = 2$.

In Theorem 2, we present a bound for the weighted error function for a given partition of the dataset, $\mathcal{P}$, with respect to the K-means error function over the entire dataset, using information associated to the misassignment function and the corresponding spatial partition.

[11]The proof of Theorem 1 is in Appendix B.

Theorem 2 *Given a dataset, D, a set of K centroids C and a spatial partition $\mathcal{B}$ of the dataset D, the following inequality is satisfied:*

$$|E^D(C) - E^{\mathcal{P}}(C)| \leq \sum_{B \in \mathcal{B}} 2 \cdot |P| \cdot \epsilon_{C,D}(B) \cdot (2 \cdot l_B + \|\overline{P} - \boldsymbol{c}_{\overline{P}}\|) + \frac{|P|-1}{2} \cdot l_B^2,$$

where $P = B(D)$ and $\mathcal{P} = \mathcal{B}(D)$. Furthermore, for a well assigned partition $\mathcal{P}$, if $C^{\mathcal{P}}_{OPT} = \arg\min_{C \subset \mathbb{R}^d, |C|=K} E^{\mathcal{P}}(C)$ and $C_{OPT} = \arg\min_{C \subset \mathbb{R}^d, |C|=K} E^D(C)$, then

$$E^D(C^{\mathcal{P}}_{OPT}) \leq E^D(C_{OPT}) + (n - |\mathcal{P}|) \cdot l^2,$$

where $l = \max_{B \in \mathcal{B}} l_B$.[12]

According to this result, we must increase the amount of well assigned blocks and/or reduce the diagonal lengths of the blocks of the spatial partition, so that our weighted error function approximates better the K-means error function, Eq.1. Observe that, by reducing the diagonal of the blocks, i) the condition of Theorem 1 is more likely to be satisfied, ii) both additive terms of the bound in Theorem 2 are reduced. This last point gives the intuition for our new partition strategy: i) split only those blocks in the boundary and ii) split them on their largest side. This partition strategy alleviates **Problem 3**. It should be noted that, in order to reduce the diagonal length in high dimensional domains, the new strategy could require a linear number of iterations with respect to the dimensionality of the dataset, d. This suggests that BWKM is particularly advisable for tall data domains.

2.2 Initial Partition

In this section, we elaborate on the construction of the initial dataset partition used by the BWKM algorithm (see `Step 1` of Algorithm 5, where the main pseudo-code of BWKM is). Starting with the smallest bounding box of the dataset, the proposed procedure iteratively divides subsets of blocks of the spatial partition with high probabilities of not being well assigned. In order to determine these blocks, in this section we develop a heuristic based on the misassignment function, Eq.3.

As our new cutting criterion is mostly based on the evaluation of the misassignment function associated to a certain block, we firstly need to construct a starting spatial partition of size $m' \geq K$, from where we can select the set of K centroids with respect to which the misassignment function is computed (`Step 1`).

From then on, multiple sets of centroids C are selected via a weighted K-means++ run over the set of representatives of the dataset partition, for different subsamplings. The subsampling strategy allows to control the trade off

[12]The proof of Theorem 2 is in Appendix B.

between the computational cost of Algorithm 2 and the quality of the obtained initialization. In general, this process allows us to estimate a probability distribution that quantifies the chances of each block of not being well assigned (`Step 2`). Then, according to this distribution, we randomly select the most promising blocks to be cut (`Step 3`), and divide them until reaching a predefined number of blocks m (`Step 4`). In Algorithm 2, we show the pseudo-code of the algorithm proposed for generating the initial spatial partition.

Algorithm 2: Construction of the initial partition

```
Input: Dataset $D$, number of clusters $K$, integer $m' > K$, size of the
initial spatial partition $m > m'$.
Output: Initial spatial partition $\mathcal{B}$ and its induced dataset partition,
$\mathcal{P} = \mathcal{B}(D)$.
Step 1: Obtain a starting spatial partition of size $m'$, $\mathcal{B}$ (Algorithm 3).
while $|\mathcal{B}| < m$ do
    Step 2: Compute the cutting probability, $Pr(B)$ for $B \in \mathcal{B}$
     (Algorithm 4).
    Step 3: Sample $\min\{|\mathcal{B}|, m - |\mathcal{B}|\}$ blocks from $\mathcal{B}$, with replacement,
     according to $\Pr(\cdot)$ to determine a subset of blocks $\mathcal{A} \subseteq \mathcal{B}$.
    Step 4: Split each $B \in \mathcal{A}$ and update $\mathcal{B}$.
end
Step 5: Construct $\mathcal{P} = \mathcal{B}(D)$.
return $\mathcal{B}$ and $\mathcal{P}$.
```

In `Step 1`, a partition of the smallest bounding box containing the dataset D, B_D, of size $m' > K$ is obtained by splitting recursively the blocks according to the pseudo-code shown in Algorithm 3 (see Section 2.2.1). Once we have the spatial partition of size m', we iteratively produce thinner partitions of the space as long as the number of blocks is lower than m. At each iteration, the process is divided into three steps: In `Step 2`, we estimate the cutting probability $Pr(B)$ for each block B in the current space partition $\mathcal{B}$ using Algorithm 4 (see Section 2.2.2). Then, in `Step 3`, we randomly sample (with replacement) $\min\{|\mathcal{B}|, m - |\mathcal{B}|\}$ blocks from $\mathcal{B}$ according to $Pr(\cdot)$ to construct the subset of blocks $\mathcal{A} \subseteq \mathcal{B}$, i.e., $|\mathcal{A}| \leq \min\{|\mathcal{B}|, m - |\mathcal{B}|\}$. Afterwards, each of the selected blocks in $\mathcal{A}$ is replaced by two smaller blocks obtained by splitting B in the middle point of its longest side. Finally, the obtained spatial partition $\mathcal{B}$ and the induced dataset partition $\mathcal{B}(D)$ (of size lower or equal to m) are returned.

2.2.1 Construction of the starting spatial partition in Algorithm 2 (`Step 1`)

Algorithm 3 generates the starting spatial partition of size m' of the dataset D. This procedure recursively obtains thinner partitions by splitting a subset of up to $\min\{|\mathcal{B}|, m' - |\mathcal{B}|\}$ blocks selected by a random sampling with replacement according to a probability proportional to the product of the diagonal of the block B, l_B, by its weight, $|B(S)|$. At this step, as we can not estimate how likely it is for a given block to be well assigned with respect to a set of K representatives. The goal is to control both weight and size of the generated spatial partition, in order to reduce the possible number of cluster misassignments, as this cutting procedure prioritizes those blocks that might be large and dense. Ultimately, as we reduce this factor, we improve the accuracy of our approximation, see Theorem 2.

This process is repeated until a spatial partition with the desired number of blocks, $m' \geq K$, is obtained. Such a partition is later used to determine the sets of centroids which we use to verify how likely it is for a certain block to be well assigned.

Algorithm 3: Step 1 of Algorithm 2

```
Input: Dataset $D$, partition size $m' > K$, sample size $s < n$.
Output: A spatial partition of size $m'$, $\mathcal{B}$.
- Set $\mathcal{B} = \{B_D\}$.
while $|\mathcal{B}| < m'$ do
    - Take a random sampling of size $s$, $S \subset D$ .
    - Obtain a subset of blocks, $\mathcal{A} \subseteq \mathcal{B}$, by sampling, with replacement,
      $\min\{|\mathcal{B}|, m' - |\mathcal{B}|\}$ blocks according to a probability proportional to
      $l_B \cdot |B(S)|$, for each $B \in \mathcal{B}$.
    - Split the selected blocks $\mathcal{A}$ and update $\mathcal{B}$.
end
return $\mathcal{B}$.
```

2.2.2 Cutting probability function, $\Pr(\cdot)$ *(`Step 2`)*

In Algorithm 4, we show the pseudo-code used in `Step 2` of Algorithm 2 for computing the cutting probabilities associated to each block $B \in \mathcal{B}$, $Pr(B)$. Such a probability function depends on the misassignment function associated to each block with respect to multiple K-means++ based set of centroids. To generate these sets of centroids, r subsamples of size s, with replacement, are extracted from the dataset, D. In particular, the **cutting probabilities** are expressed as follows:

$$\Pr(B) = \frac{\sum_{i=1}^{r} \epsilon_{C^i,S^i}(B)}{\sum_{B' \in \mathcal{B}} \sum_{i=1}^{r} \epsilon_{C^i,S^i}(B')} \tag{5}$$

for each $B \in \mathcal{B}$, where S^i is the subset of points sampled and C^i is the set of K centroids obtained via K-means++ for $i = 1, ..., r$. As we commented before, the larger the misassignment function is, then the more likely it is for the corresponding block to contain instances that belong to different clusters. It should be highlighted that a block B with $Pr(B) = 0$ is well assigned for all S^i and C^i, with $i = 1, .., r$.

Algorithm 4: Step 2 of Algorithm 2

```
Input: A spatial partition $\mathcal{B}$ of size higher than $K$, dataset $D$, number
of clusters $K$, sample size $s$, number of repetitions $r$.
Output: Cutting probability $\Pr(B)$ for each $B \in \mathcal{B}$.
for $i = 1, \dots, r$ do
    -Take subsample $S^i \subseteq D$ of size $s$ and construct $\mathcal{P} = \mathcal{B}(S^i)$.
    -Obtain a set of centroids $C^i$ by applying $K$-means++ over the
      representatives of $\mathcal{P}$.
    - Compute $\epsilon_{S^i,C^i}(B)$ for all $B \in \mathcal{B}$ (Eq.3).
end
Step 4: Compute $Pr(B)$ for every $B \in \mathcal{B}$ (Eq.5).
return $\Pr(\cdot)$.
```

Even when cheaper seeding procedures, such as a Forgy type initialization, could be used, K-means++ avoids cluster oversampling, and so one would expect the corresponding boundaries not to divide subsets of points that are supposed to have the same cluster affiliation. Additionally, as previously commented, this initialization also tends to lead to competitive solutions. Later on, in Appendix A.1, we will comment on the selection of the different parameters used in the initialization (m, m', r and s) for the empirical comparison in Section 3. It is important to remark that these parameters, with especial focus on the size of the subsampling (s), controls the trade-off between the quality and the computational cost of the initialization process.

2.3 Construction of the sequence of thinner partitions

In this section, we provide the pseudo-code of the BWKM algorithm and introduce a detailed description of the construction of the sequence of thinner partitions, which is the basis of BWKM. In general, once the initial partition is constructed via algorithm 2, BWKM progresses iteratively by alternating i) a run of weighted Lloyd's algorithm over the current partition and ii) the creation of a thinner partition using the information provided by the weighted Lloyd's algorithm. The pseudo-code of the BWKM algorithm can be seen in Algorithm 5.

In `Step 1`, the initial spatial partition $\mathcal{B}$ and the induced dataset partition, $\mathcal{P} = \mathcal{B}(D)$, are generated via Algorithm 2. Then, the initial set of centroids is obtained through a weighted version of K-means++ over the set of representatives of $\mathcal{P}$. Given the current set of centroids C and the partition of the dataset $\mathcal{P}$, the set of centroids is updated in `Step 2` and `Step 4` by applying the weighted Lloyd's algorithm. It must be commented that the only difference between these two tasks is the fact that `Step 2` is initialized with a set of centroids obtained via weighted K-means++ run, while `Step 4` utilizes the set of centroids generated by the weighted Lloyd's algorithm over the previous dataset partition. In addition, in order to compute the misassignment function $\epsilon_{C,D}(B)$ for all $B \in \mathcal{B}$ in `Step 3` (see Eq.3), we store the following information provided by the last iteration of the weighted Lloyd's algorithm: for each $P \in \mathcal{P}$, the two closest centroids to the representative $\overline{P}$ in C are saved (see Figure 1).

In `Step 3`, a spatial partition thinner than $\mathcal{B}$ and its induced dataset partition are generated. For this purpose, the misassignment function, $\epsilon_{C,D}(B)$ for all $B \in \mathcal{B}$ is computed and the boundary $\mathcal{F}_{C,D}(\mathcal{B})$ is determined using the information stored at the last iteration of `Step 2`. Next, as the misassignment criterion in Theorem 1 is just a sufficient condition, instead of dividing all the blocks that do not satisfy it, we prioritize those blocks that are less likely to be well assigned: A set $\mathcal{A}$ of blocks is selected by sampling with replacement $|\mathcal{F}_{C,D}(\mathcal{B})|$ blocks from $\mathcal{B}$ with a (cutting) probability proportional to $\epsilon_{C,D}(B)$. Note that the size of $\mathcal{A}$ is at most $|\mathcal{F}_{C,D}(\mathcal{B})|$. Afterwards, in order to reduce as much as possible the length of the diagonal of the newly generated blocks and control the size of the thinner partition, each block in $\mathcal{A}$ is divided in the middle point of its largest side. Each block is split once into two equally shaped hyper-rectangles and it is replaced in $\mathcal{B}$ to produce the new thinner spatial partition. Finally, given the new spatial partition $\mathcal{B}$, its induced dataset partition is obtained $\mathcal{P} = \mathcal{B}(D)$.

Algorithm 5: BWKM Algorithm

```
**Input:** Dataset $D$, number of clusters $K$ and initialization parameters
$m'$, $m$, $s$, $r$.
**Output:** Set of centroids $C$.
Step 1: Initialize $\mathcal{B}$ and $\mathcal{P}$ via Algorithm 2, with input $m'$, $m$, $s$, $r$, and
  obtain $C$ by applying a weighted $K$-means++ run over the set of
  representatives of $\mathcal{P}$.
Step 2: $C = \texttt{WeightedLloyd}(\mathcal{P}, C, K)$.
while not Stopping Criterion do
    Step 3: Update dataset partition $\mathcal{P}$:
    - Compute $\epsilon_{C,D}(B)$ for all $B \in \mathcal{B}$.
    - Select $\mathcal{A} \subseteq \mathcal{F}_{C,D}(\mathcal{B}) \subseteq \mathcal{B}$ by sampling, with replacement, $|\mathcal{F}_{C,D}(\mathcal{B})|$
      blocks according to $\epsilon_{C,D}(B)$, for all $B \in \mathcal{B}$.
    - Cut each block in $\mathcal{A}$ and update $\mathcal{B}$ and $\mathcal{P}$.
    Step 4: $C = \texttt{WeightedLloyd}(\mathcal{P}, C, K)$.
end
return $C$
```

In Fig.2, we show an example of 6 iterations of the BWKM algorithm on a mixture of three 2D Gaussians. In this figure, the green circles represent the initialization of the current BWKM iteration and, the black circles, the obtained approximation, while the brown dots stand for the set of representatives. As expected, it can be seen that as we increase the number of iterations most of the representatives tend to be placed in those areas around the cluster boundaries. Analogously, the furthest we are from such frontiers the largest the cells associated to the spatial partition are. Furthermore, after the applying the weighted version of Lloyd's algorithm over the initial spatial partition, both the initialization and the obtained approximation tend to overlap, meaning that BWKM using a very small amount of representatives is able to generate a competitive approximation and possible local minima of the original K-means problem.

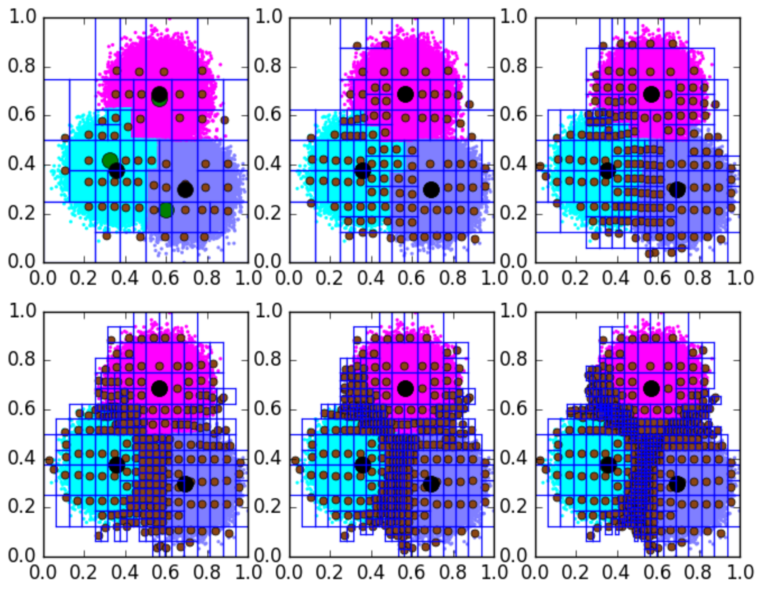

Fig. 2: Clusters and spatial partitions obtained for six consecutives iterations of BWKM on a mixture of three 2D Gaussians: The set representatives generated by BWKM tend to accumulate around the cluster boundaries, as we increase the number of iterations of the algorithm.

It should be noted that the cutting criterion, Eq.3, is more accurate, i.e., it detects more well assigned blocks, as long as we evaluate it over the smallest bounding box of each block of the spatial partition, since we minimize the maximum distance (diagonal) between any two points in the block. Therefore, when updating the data partition in `Step 3`, we also recompute the diagonal of the smallest bounding box of each subset. `Step 2` and `Step 3` are then repeated until a certain stopping criterion is satisfied, see Appendix A.2.

2.3.1 Computational complexity of the BWKM algorithm

In this section, we provide some insights on the computational complexity of each step of BWKM, in the worst case. It should be noted that the construction of the initial spatial partition, the corresponding induced dataset partition and the set of centroids of BWKM (`Step 1`) has the following computational cost: $\mathcal{O}(\max\{r\cdot s\cdot m^2,\ r\cdot K\cdot d\cdot m^2,\ \mathcal{O}(n\cdot\max\{m,\ d\})\})$. Each of the previous terms corresponds to the complexity of `Step 1`, `Step 2` and `Step 5` in Algorithm 2, respectively, which are the most computationally demanding procedures of the initialization. Even when these costs are deduced from the worst possible scenario, which is overwhelmingly improbable, in Appendix A.1, we will elaborate on the selection of the initialization parameters in such a way that the cost of this step is not more expensive than that of the K-means algorithm. In particular, assuming that r is a predefined small integer, satisfying $r \ll n/s$, we propose the use of $m = \mathcal{O}(\sqrt{K\cdot d})$ and $s = \mathcal{O}(\sqrt{n})$. More specifically, in Section 3, the values $m = 10\cdot\sqrt{K\cdot d}$, $s = \sqrt{n}$ and $r = 5$ showed a good trade off between the quality of the found solution and the number of distances computed.

As mentioned at the beginning of Section 1.2.2, `Step 2` of Algorithm 5 (the weighted Lloyd's algorithm) has a computational complexity of $\mathcal{O}(|\mathcal{P}|\cdot K\cdot d)$. In addition, `Step 3` executes $\mathcal{O}(|\mathcal{P}|\cdot K)$ computations to verify the cutting criterion, since all the distance computations are obtained from the previous weighted Lloyd iteration. Moreover, assigning each instance to its corresponding block and updating the bounding box for each subset of the partition is $\mathcal{O}(n\cdot d)$. In summary, since $|\mathcal{P}| \leq n$, then BWKM has an overall cost of $\mathcal{O}(n\cdot K\cdot d)$ in the worst case.

3 Experiments

In this section, we perform a set of experiments so as to analyze the trade-off between the computational performance and the quality of the approximation obtained by the **BWKM** algorithm proposed in Section 2 and different methods known for the quality of their approximations[13]: Lloyd's algorithm initialized via i) Forgy (**FKM**) and ii) K-means++ (**KM++**) [14]. We also consider the Minibatch K-means, with batches $b = \{100, 500, 1000\}$ [15] (**MB b**), which is particularly known for its efficiency due to the small amount of resources needed to generate its approximation, as well as the Markov chain Monte Carlo sampling based approximation of the K-means++ (**AFKMC2**) with a further MB 100 run as recommended by its authors (Bachem et al., 2016).

As previously commented, the computational load of the methods considered in our experimental setting is dominated by the number of distance computations. Therefore, as has been used in many articles related to the

[13]Additionally, in Appendix C, we comment on the grid based RPKM.

[14]The output of such an initialization is presented as **KM++_init**.

[15]Similar values were used in the original paper (Sculley, 2010).

K-means problem, such as Bachem et al. (2016); Elkan (2003), we use the number of distances computed to measure the computational performance of all these approaches. On the other hand, the quality of the approximation is measured via the K-means error function, Eq.1.

To have a better understanding of BWKM, we analyze its performance on a wide variety of well known real datasets (see Table 1) with different scenarios of the clustering problem. For each dataset, we have considered a different number of clusters, $K = \{3, 5, 10, 25, 50\}$. Given the random nature of the algorithms, each experiment has been repeated 50 times for each dataset and each K value.

Dataset	n	d
Corel Image Features (CIF)	$68,037$	17
3D Road Network (3RN)	$434,874$	3
Household Power Consumption (HPC)	$2,049,280$	7
Gas Sensor (GS)	$4,208,259$	19
SUSY	$5,000,000$	19
Web Users Yahoo! (WUY)	$45,811,883$	5

Table 1: Information of the datasets.

As stopping criterion, we have fixed the maximum number of BWKM iterations to 100. However it might converge before if the corresponding boundary is empty, in which case, we can guarantee that the obtained set of centroids is a fixed point of the weighted Lloyd's algorithm for any thinner partition of the dataset, therefore, it is also a fixed point of Lloyd's algorithm on the entire dataset D (see Theorem 3). Moreover, in order to compare the performance of the algorithms for different settings of the clustering problem, we decided to use the average of the relative error with respect to the best solution found at each repetition of the experiment, i.e., $\hat{E}_M = \frac{E_M - \min\limits_{M' \in \mathcal{M}} E_{M'}}{\min\limits_{M' \in \mathcal{M}} E_{M'}}$, where $\mathcal{M}$ is the set of algorithms being compared and E_M stands for the K-means error obtained by method $M \in \mathcal{M}$. Analogously, in terms of the amount of computational resources required, we show the proportion of distances computed by each method with respect to the method that computed the largest number of distances, i.e., $\hat{DC}_M = \frac{DC_M}{\max\limits_{M' \in \mathcal{M}} DC_{M'}}$, where DC_M is the number of distances computed by $M \in \mathcal{M}$.

In Fig. 3-8 and Table2, we show the trade-off between the average relative number of distances computed *vs* the average relative error for all the algorithms. Observe that a single symbol is used for each algorithm, except for BWKM, in which we compute the trade-off at each iteration so as to observe the evolution of the quality of its approximation as the number of computed distances increases.

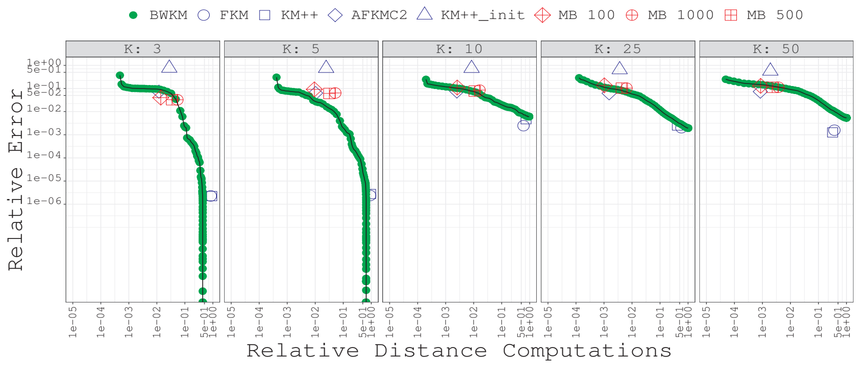

Fig. 3: Relative distance computations *vs* relative error on the *CIF* dataset.

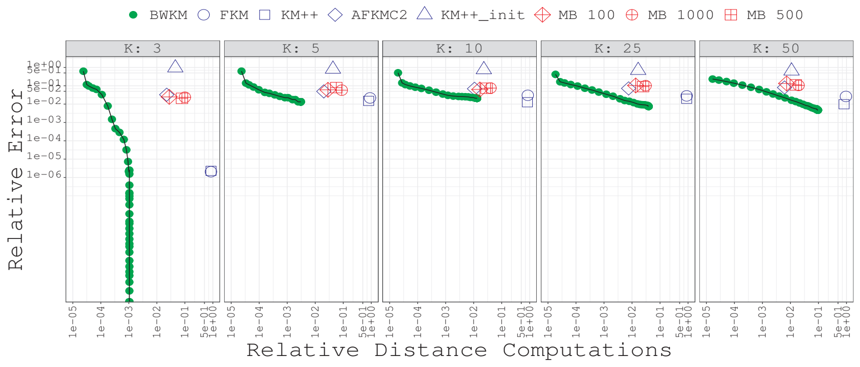

Fig. 4: Relative distance computations *vs* relative error on the *3RN* dataset.

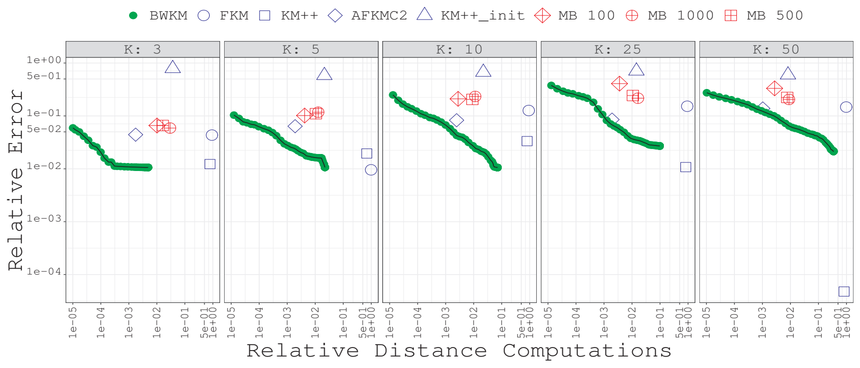

Fig. 5: Relative distance computations *vs* relative error on the *HPC* dataset.

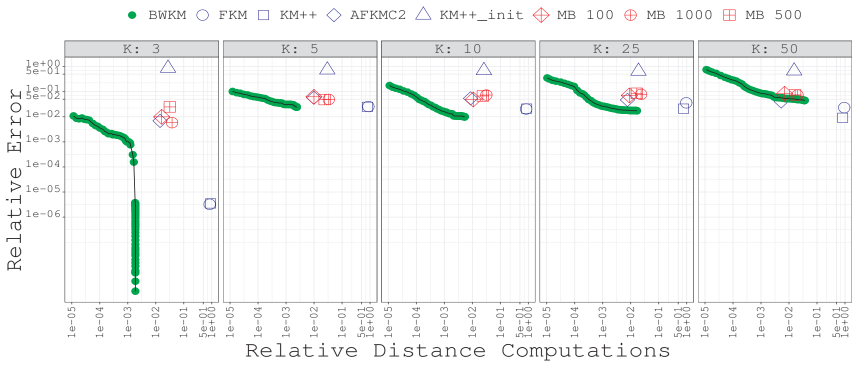

Fig. 6: Relative distance computations *vs* relative error on the *GS* dataset.

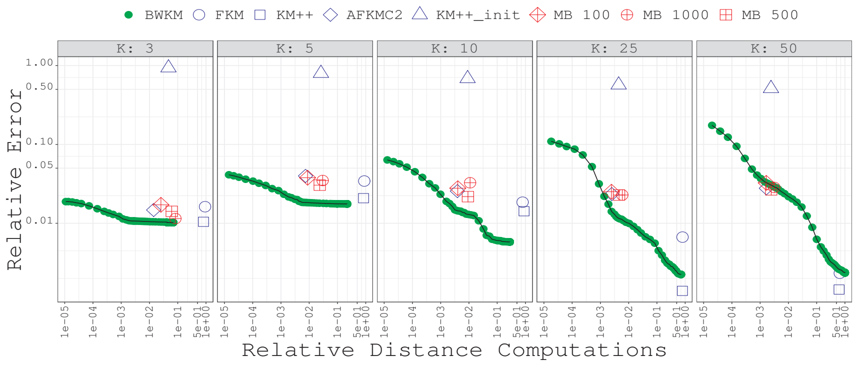

Fig. 7: Relative distance computations *vs* relative error on the *SUSY* dataset.

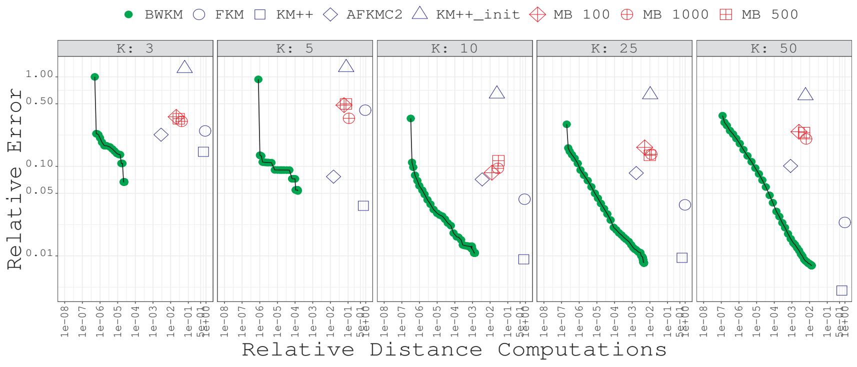

Fig. 8: Relative distance computations *vs* relative error on the *WUY* dataset.

Method	K	*CIF*		*3RN*		*HPC*		*GS*		*SUSY*		*WUY*	
		5%	1%	5%	1%	5%	1%	5%	1%	5%	1%	5%	1%
FKM	**3**	2	2	4	4	5	5	6	5	5	5	7	6
	5	3	2	4	4	4	3	5	3	6	6	7	7
	10	2	1	4	4	5	5	4	4	5	3	7	6
	25	2	1	4	3	4	4	4	4	3	3	6	6
	50	1	1	4	3	4	4	3	*	3	1	5	4
KM++	**3**	2	2	4	4	5	4	6	5	5	5	6	5
	5	3	2	4	4	5	3	5	4	6	4	6	6
	10	2	1	4	2	4	3	4	4	5	3	6	5
	25	2	1	4	3	3	*	4	4	3	2	5	4
	50	1	0	3	2	2	*	3	*	3	1	4	3
AFKMC2	**3**	2	0	3	3	3	3	4	4	4	4	4	4
	5	2	2	3	3	2	2	3	3	4	4	5	5
	10	1	0	3	2	2	1	3	2	3	1	4	4
	25	0	0	3	2	1	1	2	2	1	1	4	4
	50	0	1	2	1	1	1	1	0	1	1	3	3
KM++ init	**3**	2	2	4	4	4	4	4	4	4	4	6	6
	5	2	2	4	4	4	4	4	4	4	4	5	5
	10	2	2	3	3	4	4	4	4	4	4	5	5
	25	2	2	3	3	4	4	4	4	4	4	4	4
	50	2	2	3	3	4	4	4	3	4	4	5	5
MB 100	**3**	1	0	3	3	3	3	4	4	4	4	5	5
	5	2	2	3	3	3	3	3	3	4	4	5	5
	10	1	1	3	2	3	2	3	2	4	2	5	5
	25	1	1	3	3	3	3	2	2	1	1	5	5
	50	1	0	2	2	3	3	2	1	1	1	4	4
MB 500	**3**	1	0	3	3	4	4	4	4	4	4	5	5
	5	2	2	4	3	4	3	4	3	4	4	5	5
	10	2	1	3	3	3	3	3	3	4	2	5	5
	25	2	1	3	3	3	2	3	2	1	1	5	5
	50	1	0	3	2	3	3	2	2	1	1	5	5
MB 1000	**3**	1	0	4	3	4	4	4	4	4	4	5	5
	5	2	2	4	3	4	4	4	3	4	4	6	6
	10	2	1	3	3	3	3	3	3	4	2	5	5
	25	2	1	3	3	3	2	3	3	2	1	5	5
	50	2	1	3	2	3	3	2	2	1	1	5	4

Table 2: Orders of reduction of distances computed by BWKM with respect to the considered methods for reaching under 5% and 1% of their relative error.

At first glance, we observe that, in 18 out of 35 different configurations of datasets and K values, BWKM obtained the best (average) solution among the considered methods. Furthermore, in Table 2 we observe that BWKM quite frequently (in 206 out of 210 cases) converged to sets of centroids that reached on average, at least, 1% of error with respect to all the considered methods and clustering configurations. If we increase such a threshold to 5%, BWKM reaches it in every case. It must be highlighted that such clusterings were generated while computing a massively reduced number of distances: Up to 5 and 7 orders of magnitude of distances less than the *Minibatch based methods* (MB 100, MB 500, MB 1000 and AFKMC2) and the *Lloyd's based methods* (FKM and KM++), respectively. In particular and as expected, the best performance of BWKM seems to occur on large datasets with small dimensions (*WUY*). On one hand, the decrease in the amount of distances computed is mainly due to the reduction in the number of representatives that BWKM uses in comparison to the actual size of the dataset. On the other hand, given a set of points as the dimension decreases, the number of blocks required to obtain a partition completely well assigned tends to decrease (*WUY* and *3RN*).

Regardless of this, even when considering the most unfavorable setting for BWKM (smallest dataset size with large dimension, i.e., *CIF*), for small K values, our proposal still managed to converge to competitive solutions (under 1% of error when compared to the competition) at a fast rate (reducing on average up to 2 orders of distance computations). Note that for small K values, since the number of centroids is small, one may not need to reduce the diagonal of the blocks so abruptly to verify the well assignment criterion. On the other hand, for the largest numbers of clusters, BWKM still generated solutions of the same quality but struggled to reduce the number of computations: BWKM computed the same order of distances as the Minibatch based methods to generate approximations with a similar error.

In the case of small datasets with low dimensionality (*3RN*), BWKM performs much better in comparison to the previous case: In 4 out of 5 values of K, BWKM actually generates the most competitive solutions. In particular, in order to achieve a relative error of under 1% of error, BWKM reduces between 1 to 3 orders of magnitude of distances with respect to the Minibatch based methods, and 2 to 4 orders of magnitude against the Lloyd's based methods. Furthermore, for the medium size datasets with low dimensionality (*HPC*), BWKM has a similar performance, leading on average to the most qualitative solution in 3 out of 5 values of K, while reducing between 1 to 4 orders of magnitude of distances with respect to the Minibatch based methods, and 2 to 5 orders of magnitude against the Lloyd's based methods to generate solutions under 5% of their error.

If we consider the case of the medium to large datasets with larger dimensionality (*GS* and *SUSY*), in order to reach a 5% relative error, BWKM decreases between 1 to 4 orders of magnitude with respect to the Minibatch based methods and 3 to 6 orders in comparison to the Lloyd's based methods. Moreover, BWKM obtains the solutions with the lowest errors in 5 out of 10 configurations.

For the largest dataset (*WUY*), BWKM got its best performance. Again, it usually generated the most competitive solutions (in 4 out of 5 cases), however, in this case, as expected BWKM computes an amount of distance from 4 to 6 and 3 to 7 orders of magnitude lower than the Minibatch and the Lloyd's based algorithms to reach a relative error of under 1% with respect to them, respectively.

Finally, we would like to highlight that BWKM, already at its first iterations, reaches a relative error much lower than KM++_init in all the configurations requiring to compute an amount of distances from 2 to 6 order of magnitude lower. This fact strongly motivates the use of BWKM as a competitive initialization strategy for Lloyd's algorithm.

Undoubtedly, BWKM achieves its best results, in terms of the trade-off between number of distance computations and the quality of the solution obtained, when dealing with large datasets with small dimensions (*tall data*) and not a large number of clusters, since in this scenario the ratio between the number of representatives and instances is reduced. Therefore, the use of BWKM is mostly recommended for this setting. In any case, and in spite of considered configuration, BWKM shows quite a competitive performance when compared to the state-of-the-art, leading to reductions of several orders of distance computations while converging to sets of centroids with a similar and/or lower error than the one achieved by the considered algorithms. Furthermore, it must be highlighted that the modifications made in the cutting criterion allowed our algorithm to scale to dimensions that were intractable for the previous grid based RPKM in Capó et al. (2017).

4 Conclusions

In this work, we have presented an alternative to the K-means algorithm, that is especially advisible for tall data domains, called the Boundary Weighted K-means algorithm (BWKM). This approach recursively applies a weighted version of the K-means algorithm over a sequence of spatial based partitions of the dataset that ideally contains a large amount of *well assigned blocks*, i.e., cells of the spatial partition that only contain instances with the same cluster affiliation. It can be shown that BWKM produces better approximations to the K-means error function as the number of well assigned blocks increases (see Theorem 2). In the practice, we have observed that the number of points that fall into the well asigned blocks tends to increase with respect to the number of iterations of BWKM. Thus, the approximation is more accurate. Ultimately, if all the blocks of a spatial partition are well assigned at the end of a BWKM step, then the obtained clustering is actually a fixed point of the K-means algorithm. By Theorem 2, if two consecutive weighted Lloyd's iterations only contain well assigned blocks, then the error of our approximation also decreases monotonically. Furthermore, in the practice, we have observed that such a monotonic error descend occurs with high probability.

In order to achieve this, in Section 2.1, we designed a criterion to determine those blocks that may not be well assigned. One of the major advantages of the criterion is its low computational cost: It only uses information generated by the weighted K-means algorithm -distances between the center of mass of each block and the set of centroids- and a feature of the corresponding spatial partition -diagonal length of each block-. This allows us to guarantee that, even in the worst possible case, BWKM does not have a computational cost higher than that of the K-means algorithm. In particular, the criterion is presented in Theorem 1 and states that, if the diagonal of a certain block is smaller than half the difference of the two the smallest distances between its center of mass and the set of centroids, then the block is well assigned.

In addition to all the theoretical guarantees that motivated and justify our algorithm (see Section 2 and Appendix B), in practice, we have also observed its competitiveness with respect to the state-of-the-art (Section 3). BWKM has been compared to Lloyd's algorithm initialized with Forgy's approach and K-means++, the AFKMC2 algorithm and the Minibatch K-means.

The results, on different well known real datasets, show that BWKM in several cases (18 out of 35 configurations) has generated the most competitive solutions. Furthermore, in 206 out of 210 cases, BWKM has converged to solutions with a relative error of under 1% with respect to the considered methods, while using a much smaller amount of distance computations (up to 7 orders of magnitude lower).

As for the next steps, we plan to exploit different benefits of BWKM. First of all, observe that the proposed algorithm is embarrassingly parallel up to the K-means++ seeding of the initial partition (over a very tiny amount of representatives when compared to the dataset size), hence we could implement this approach in different platforms such as *Apache Spark*. Moreover, we must point out that BWKM is also compatible with the distance pruning techniques presented in Ding et al. (2015); Drake and Hamerly (2012); Elkan (2003); Hamerly (2010), therefore, we could also implement these techniques within the weighted Lloyd framework of BWKM and reduce, even more, the number of distance computations. In addition, we could develop partition strategies that speed up the construction of well assigned blocks while still scaling well on the dimensionality of the problem. One step in this direction could be using the current cluster boundaries to determine those blocks that are well assigned and construct the sequence of thinner partitions, while dividing the current blocks on multiple dimensions at once. On the same token, we could also analyze the performance of different dimensionality reduction techniques, such as Principal Component Analysis (Cohen et al., 2015; Ding and He, 2004), Singular Value Decomposition (Boutsidis et al., 2009; Cohen et al., 2015), Random Projections (Boutsidis et al., 2010; Cohen et al., 2015) and representation via binary codes (Shen et al., 2017), that have successfully used in other heuristics for the K-means problem.

It should be remarked that we can easily extend the use of BWKM in a streaming manner for the K-means problem. In particular, we can update a given BWKM approximation by placing the new set of instances in their corresponding blocks of the current spatial partition and updating the set of representatives (centers of mass). From a theoretical standpoint, one could exploit the geometric features of the spatial partition to, for instance, deduce conditions under which a well assigned block remains the same, after receiving the new subset of points in streaming, as well as for obtaining quality guarantees of the seeding for the entire data set. Finally, as it occurs for different coreset-type algorithms, we could extend the use of BWKM to other clustering techniques such as K-median and K-medoids (Balcan et al., 2013; Har-Peled and Mazumdar, 2004).

Acknowledgements Marco Capó and Aritz Pérez are partially supported by the Basque Government through the BERC 2014-2017 program and the ELKARTEK program, and by the Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa excelence accreditation SVP-2014-068574 and SEV-2017-0718, and through the project TIN2017-82626-R funded by (AEI/FEDER, UE). Jose A. Lozano is partially supported by the Basque Government (IT1244-19), and Spanish Ministry of Economy and Competitiveness MINECO (TIN2016-78365-R).

A Additional Remarks on BWKM

In this section, we discuss additional features of the BWKM algorithm, such as the selection of the initialization parameters for BWKM, we also comment on different possible stopping criteria, with their corresponding computational costs and theoretical guarantees.

A.1 Parameter selection

The construction of the initial space partition and the corresponding induced dataset partition of BWKM (see Algorithm 2 and `Step 1` of Algorithm 5) depends on the parameters m, m', r, s, K and D, while the core of BWKM (`Step 2` and `Step 3`) only depends on K and D. In this section, we propose how to select the parameters m, m', r and s, keeping in mind the following objectives: i) to guarantee BWKM having a computational complexity equal to or lower than $\mathcal{O}(n \cdot K \cdot d)$, which corresponds to the cost of Lloyd's algorithm, and ii) to obtain an initial spatial partition with a large amount of well assigned blocks.

In order to ensure that the computational complexity of BWKM's initialization is, even in the worst case, $\mathcal{O}(n \cdot K \cdot d)$, we must take m, m', r and s such that $r \cdot s \cdot m^2$, $r \cdot m^2 \cdot K \cdot d$ and $n \cdot m$ are $\mathcal{O}(n \cdot K \cdot d)$. On the other hand, as we want such an initial partition to minimize the number of blocks that may not be well assigned, we must consider the following facts: i) the larger the diagonal for a certain block $B \in \mathcal{B}$ is, then the more likely it is for B not to be well assigned, ii) as the number of clusters K increases, then any block $B \in \mathcal{B}$ has more chances of containing instances with different cluster affiliations, and iii) as s increases, the cutting probabilities become better indicators for detecting those blocks that are not well assigned.

Taking into consideration these observations, and assuming that r is a predefined small integer, satisfying $r \ll n/s$, we propose the use of $m = \mathcal{O}(\sqrt{K \cdot d})$ and $s = \mathcal{O}(\sqrt{n})$. Not only does such a choice satisfy the complexity constraints that we just mentioned (See Theorem 5 in Appendix B), but also, in this case, the size of the initial partition increases with respect to both dimensionality of the problem and number of clusters: Since at each iteration, we divide a block only on one of its sides, then, as we increase the dimensionality, we need more cuts (number of blocks) to have a sufficient reduction of its diagonal (observation i)). Analogously, the number of blocks and the size of the sampling increases with respect to the number of clusters and the actual size of the dataset, respectively (observation ii) and iii)). In particular, in the experimental section, Section 3, we used $m = 10 \cdot \sqrt{K \cdot d}$, $s = \sqrt{n}$ and $r = 5$.

A.2 Stopping Criterion

As we commented in Section 1.3, one of the advantages of constructing spatial partitions with only well assigned blocks is that our algorithm, under this setting, converges to a local minima of the K-means problem over the entire dataset and, therefore, there is no need to execute any further run of the

BWKM algorithm as the set of centroids will remain the same for any thinner partition:

Theorem 3 *If C is a fixed point of the weighted K-means algorithm for a spatial partition $\mathcal{B}$, for which all of its blocks are well assigned, then C is a fixed point of the K-means algorithm on D.* [16]

To verify this criterion, we can make use of the concept of boundary of a spatial partition (Definition 4). In particular, observe that if $\mathcal{F}_{C,D}(\mathcal{B}) = \emptyset$, then one can guarantee that all the blocks of $\mathcal{B}$ are well assigned with respect to both C and D. To check this, we just need to scan the misassignment function value for each block, i.e., it is just $\mathcal{O}(|\mathcal{P}|)$. In addition to this criterion, in this section we will propose three other stopping criteria:

- *A practical computational criterion*: We could set, in advance, the amount of computational resources that we are willing to use and stop when we exceed them. In particular, as the computation of distances is the most expensive step of the algorithm, we could set a maximum number of distances as a stopping criterion.
- *A Lloyd's algorithm type criterion*: As we mentioned in Section 1.2, the common practice is to run Lloyd's algorithm until the reduction of the error, after a certain iteration, is small, see Eq 2. As in our weighted approximation we do not have access to the error $E^D(C)$, a similar approach is to stop the algorithm when the obtained set of centroids, in consecutive iterations, is smaller than a fixed threshold, ε_w. We can actually set this threshold in a way that the stopping criterion of Lloyd's algorithm is satisfied. For instance, for $\varepsilon_w = \sqrt{l^2 + \frac{\varepsilon^2}{n^2}} - l$, if $\|C - C'\|_\infty \leq \varepsilon_w$, then the criterion in Eq.2 is satisfied[17]. However, this would imply additional $\mathcal{O}(K \cdot d)$ computations at each iteration.
- *A criterion based on the accuracy of the weighted error*: We could also consider the bound obtained at Theorem 2 and stop when it is lower than a predefined threshold. This will let us know how accurate our current weighted error is with respect to the error over the entire dataset. All the information in this bound is obtained from the weighted Lloyd iteration and the information of the block and its computation is just $\mathcal{O}(|\mathcal{P}|)$.

B Proofs

In Theorem 1, we prove the cutting criterion that we use in BWKM. It consists of an inequality that, only by using information referred to the partition of the dataset and the weighted Lloyd's algorithm, helps us guarantee that a block is well assigned.

[16]The proof of Theorem 3 is in Appendix B.

[17]See Theorem 4 in Appendix B.

Theorem 1 *Given a set of K centroids, C, a dataset, $D \subseteq \mathbb{R}^d$, and a block B, if $\epsilon_{C,D}(B) = 0$, then $\boldsymbol{c_x} = \boldsymbol{c}_{\overline{P}}$ for all $\boldsymbol{x} \in P = B(D) \neq \emptyset$.*

Proof From the triangular inequality, we know that $\|\mathbf{x} - \mathbf{c}_{\overline{P}}\| \leq \|\mathbf{x} - \overline{P}\| + \|\overline{P} - \mathbf{c}_{\overline{P}}\|$. Moreover, observe that $\overline{P}$ is contained in the block B, since B is a convex polytope. Then $\|\mathbf{x} - \overline{P}\| \leq l_B$.

For this reason, $\|\mathbf{x}-\mathbf{c}_{\overline{P}}\| \leq l_B - \delta_P(C) + \|\overline{P}-\mathbf{c}\| \leq (2\cdot l_B - \delta_P(C)) + \|\mathbf{x}-\mathbf{c}\|$ holds. As $\epsilon_{C,D}(B) = \max\{0, 2\cdot l_B - \delta_P(C)\} = 0$, then $2\cdot l_B - \delta_P(C) \leq 0$ and, therefore, $\|\mathbf{x} - \mathbf{c}_{\overline{P}}\| \leq \|\mathbf{x} - \mathbf{c}\|$ for all $\mathbf{c} \in C$. In other words, $\mathbf{c}_{\overline{P}} = \arg\min_{\mathbf{c}\in C} \|\mathbf{x} - \mathbf{c}\|$ for all $\mathbf{x} \in P$.

The two following results show some properties of the error function when having well assigned blocks.

Lemma 1 *If $\boldsymbol{c_x} = \boldsymbol{c}_{\overline{P}}$ and $\boldsymbol{c}'_{\boldsymbol{x}} = \boldsymbol{c}'_{\overline{P}}$ for all $\boldsymbol{x} \in P$, where $P \subseteq D$ and C, C' are a pair of sets of centroids, then $E^P(C) - E^{\{P\}}(C) = E^P(C') - E^{\{P\}}(C')$.*

Proof From Lemma 1 in Capó et al. (2017), we can say that the following function is constant $f(\mathbf{c}) = |P| \cdot \|\overline{P} - \mathbf{c}\|^2 - \sum_{\mathbf{x}\in P} \|\mathbf{x} - \mathbf{c}\|^2$, for $\mathbf{c} \in \mathbb{R}^d$. In particular, since $f(\overline{P}) = -\sum_{\mathbf{x}\in P} \|\mathbf{x} - \overline{P}\|^2$, we have that $|P| \cdot \|\overline{P} - \mathbf{c}_{\overline{P}}\|^2 = \sum_{\mathbf{x}\in P} \|\mathbf{x} - \mathbf{c}_{\overline{P}}\|^2 - \sum_{\mathbf{x}\in P} \|\mathbf{x} - \overline{P}\|^2$ and so we can express the weighted error of a dataset partition, $\mathcal{P}$, as follows

$$E^{\mathcal{P}}(C) = \sum_{P\in\mathcal{P}} \sum_{\mathbf{x}\in P} (\|\mathbf{x} - \mathbf{c}_{\overline{P}}\|^2 - \|\mathbf{x} - \overline{P}\|^2) \tag{6}$$

In particular, for $P \in \mathcal{P}$, we have

$$\begin{aligned} E^P(C) - E^{\{P\}}(C) &= \sum_{\mathbf{x}\in P} (\|\mathbf{x} - \mathbf{c}_{\mathbf{x}}\|^2 - \|\mathbf{x} - \mathbf{c}_{\overline{P}}\|^2 + \|\mathbf{x} - \overline{P}\|^2) \\ &= \sum_{\mathbf{x}\in P} \|\mathbf{x} - \overline{P}\|^2 \\ &= \sum_{\mathbf{x}\in P} (\|\mathbf{x} - \mathbf{c}'_{\mathbf{x}}\|^2 - \|\mathbf{x} - \mathbf{c}'_{\overline{P}}\|^2 + \|\mathbf{x} - \overline{P}\|^2) \\ &= E^P(C') - E^{\{P\}}(C') \end{aligned}$$

In the previous result we observe that, if all the instances are correctly assigned in each block, then the difference of the weighted and the entire dataset error, of both sets of centroids, is the same. In other words, if all the blocks of a given partition are correctly assigned, not only can we then actually guarantee a monotone descend of the entire error function for our approximation, a property that can not be guaranteed for the typical coreset type approximations of K-means, but we know exactly the reduction of such an error after a weighted Lloyd iteration.

Lemma 2 *Given two set of centroids C, C', where C' is obtained after a weighted Lloyd's iteration (on a partition $\mathcal{P}$) over C and $\boldsymbol{c_x} = \boldsymbol{c}_{\overline{P}}$ and $\boldsymbol{c}'_{\boldsymbol{x}} = \boldsymbol{c}'_{\overline{P}}$ for all $\boldsymbol{x} \in P$ and $P \in \mathcal{P}$, then $E^D(C') \leq E^D(C)$.*

Proof Using Lemma 1 over all the subsets $P \in \mathcal{P}$, we know that $E^D(C') - E^D(C) = \sum_{P\in\mathcal{P}}(E^P(C') - E^P(C)) = \sum_{P\in\mathcal{P}}(E^{\{P\}}(C') - E^{\{P\}}(C)) = E^{\mathcal{P}}(C') - E^{\mathcal{P}}(C)$. Moreover, from the chain of inequalities $A.1$ in Capó et al. (2017), we know that $E^{\mathcal{P}}(C') \leq E^{\mathcal{P}}(C)$ at any weighted Lloyd iteration over a given partition $\mathcal{P}$, thus $E^D(C') \leq E^D(C)$.

Up to this point, most of the quality results assume the case when all the blocks are well assigned. However, in order to achieve this, many BWKM iterations might be required. In the following result, we provide a bound to the weighted error with respect to the full error. This result shows that our weighted representation improves as more blocks of our partition satisfy the criterion in Algorithm 1 and/or the diagonal of the blocks are smaller.

Theorem 2 *Given a dataset, D, a set of K centroids C and a spatial partition $\mathcal{B}$ of the dataset D, the following inequality is satisfied:*

$$|E^D(C) - E^{\mathcal{P}}(C)| \leq \sum_{B\in\mathcal{B}} 2 \cdot |P| \cdot \epsilon_{C,D}(B) \cdot (2 \cdot l_B + \|\overline{P} - \mathbf{c}_{\overline{P}}\|) + \frac{|P|-1}{2} \cdot l_B^2,$$

where $P = B(D)$ and $\mathcal{P} = \mathcal{B}(D)$. Furthermore, for a well assigned partition $\mathcal{P}$, if $C^{\mathcal{P}}_{OPT} = \arg\min_{C\subset\mathbb{R}^d,|C|=K} E^{\mathcal{P}}(C)$ and $C_{OPT} = \arg\min_{C\subset\mathbb{R}^d,|C|=K} E^D(C)$, then

$$E^D(C^{\mathcal{P}}_{OPT}) \leq E^D(C_{OPT}) + (n - |\mathcal{P}|) \cdot l^2,$$

where $l = \max_{B\in\mathcal{B}} l_B$.

Proof Using Eq.6 in Lemma 1, we know that $|E^D(C) - E^{\mathcal{P}}(C)| \leq \sum_{P\in\mathcal{P}} \sum_{\mathbf{x}\in P} \|\mathbf{x} - \mathbf{c}_{\overline{P}}\|^2 - \|\mathbf{x} - \mathbf{c_x}\|^2 + \|\mathbf{x} - \overline{P}\|^2$.

Observe that, for a certain instance $\mathbf{x} \in P$, where $\epsilon_{C,D}(B) = \max\{0, 2 \cdot l_B - \delta_P(C)\} = 0$, $\|\mathbf{x} - \mathbf{c}_{\overline{P}}\|^2 - \|\mathbf{x} - \mathbf{c_x}\|^2 = 0$, as $\mathbf{c_x} = \mathbf{c}_{\overline{P}}$ by Theorem 1. On the other hand, if $\epsilon_{C,D}(B) > 0$, we have the following inequalities:

$$\begin{aligned}\|\mathbf{x} - \mathbf{c}_{\overline{P}}\| - \|\mathbf{x} - \mathbf{c_x}\| &\leq 2 \cdot \|\mathbf{x} - \overline{P}\| - (\|\overline{P} - \mathbf{c_x}\| - \|\overline{P} - \mathbf{c}_{\overline{P}}\|) \\ &\leq \epsilon_{C,D}(B)\end{aligned}$$

$$\begin{aligned}\|\mathbf{x} - \mathbf{c}_{\overline{P}}\| + \|\mathbf{x} - \mathbf{c_x}\| &\leq 2 \cdot \|\mathbf{x} - \overline{P}\| + \|\overline{P} - \mathbf{c_x}\| + \|\overline{P} - \mathbf{c}_{\overline{P}}\| \\ &< 2 \cdot l_B + (2 \cdot l_B + \|\overline{P} - \mathbf{c}_{\overline{P}}\|) \\ &+ \|\overline{P} - \mathbf{c}_{\overline{P}}\| \\ &= 2 \cdot (2 \cdot l_B + \|\overline{P} - \mathbf{c}_{\overline{P}}\|)\end{aligned}$$

Using both inequalities, we have $\|\mathbf{x}-\mathbf{c}_{\overline{P}}\|^2-\|\mathbf{x}-\mathbf{c}_{\mathbf{x}}\|^2 \leq 2\cdot\epsilon_{C,D}(B)\cdot(2\cdot l_B+\|\overline{P}-\mathbf{c}_{\overline{P}}\|)$. On the other hand, observe that $\sum\limits_{\mathbf{x}\in P}\|\mathbf{x}-\overline{P}\|^2 = \frac{1}{|P|}\cdot\sum\limits_{\mathbf{x},\mathbf{y}\in P}\|\mathbf{x}-\mathbf{y}\|^2 \leq \frac{1}{|P|}\cdot\frac{|P|\cdot(|P|-1)}{2}\cdot l_B^2 = \frac{|P|-1}{2}\cdot l_B^2$.

Furthermore, if the partition is well assigned, then $\epsilon_{C,D}(B)=0$ for all $B\in\mathcal{B}$ and so,

$$\begin{aligned}
E^D(C_{OPT}^{\mathcal{P}}) &\leq E^{\mathcal{P}}(C_{OPT}^{\mathcal{P}}) + \sum_{B\in\mathcal{B}}\frac{|P|-1}{2}\cdot l_B^2\\
&\leq E^D(C_{OPT}) + 2\cdot\sum_{B\in\mathcal{B}}\frac{|P|-1}{2}\cdot l_B^2\\
&\leq E^D(C_{OPT}) + (n-|\mathcal{P}|)\cdot l^2
\end{aligned}$$

In Theorem 3, we show an interesting property of the BWKM algorithm. We verify that a fixed point of the weighted Lloyd's algorithm, over a partition with only well assigned blocks, is also a fixed point of Lloyd's algorithm over the entire dataset D.

Theorem 3 *If C is a fixed point of the weighted K-means algorithm for a spatial partition $\mathcal{B}$, for which all of its blocks are well assigned, then C is a fixed point of the K-means algorithm on D.*

Proof $C=\{\mathbf{c}_1,\ldots,\mathbf{c}_K\}$ is a fixed point of the weighted K-means algorithm, on a partition $\mathcal{P}$, if and only if when applying an additional iteration of the weighted K-means algorithm on $\mathcal{P}$, the generated clusterings $\mathcal{G}_1(\mathcal{P}),\ldots,\mathcal{G}_K(\mathcal{P})$, i.e., $\mathcal{G}_i(\mathcal{P}) := \{P\in\mathcal{P} : \mathbf{c}_i = \arg\min_{\mathbf{c}\in C}\|\overline{P}-\mathbf{c}\|\}$, satisfies $\mathbf{c}_i = \frac{\sum_{P\in\mathcal{G}_i(\mathcal{P})}|P|\cdot\overline{P}}{\sum_{P\in\mathcal{G}_i(\mathcal{P})}|P|}$ for all $i=\{1,\ldots,K\}$ (1).

Since all the blocks $B\in\mathcal{B}$ are well assigned, then the clusterings of C in D, $\mathcal{G}_i(D) := \{\mathbf{x}\in D : \mathbf{c}_i = \arg\min_{\mathbf{c}\in C}\|\mathbf{x}-\mathbf{c}\|\}$, satisfy $|\mathcal{G}_i(D)| = \sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}|P|$ (2) and $\sum\limits_{\mathbf{x}\in\mathcal{G}_i(D)}\mathbf{x} = \sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}\sum\limits_{\mathbf{x}\in P}\mathbf{x}$ (3). From (1), (2) and (3), we have

$$\begin{aligned}
\mathbf{c}_i &= \frac{\sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}|P|\cdot\overline{P}}{\sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}|P|} = \frac{\sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}|P|\cdot\sum\limits_{\mathbf{x}\in P}\frac{\mathbf{x}}{|P|}}{\sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}|P|}\\
&= \frac{\sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}\sum\limits_{\mathbf{x}\in P}\mathbf{x}}{\sum\limits_{P\in\mathcal{G}_i(\mathcal{P})}|P|} = \frac{\sum\limits_{\mathbf{x}\in\mathcal{G}_i(D)}\mathbf{x}}{|\mathcal{G}_i(D)|}\forall\ i\in 1,\ldots,K,
\end{aligned}$$

this is, C is a fixed point of K-means algorithm on D.

As we do not have access to the error for the entire dataset, $E^D(C)$, since its computation is expensive, in Algorithm 5 we propose a possible stopping criterion that bounds the displacement of the set of centroids. In the following result, we show a possible choice of this bound in a way that, if the proposed criterion is verified, then the common Lloyd's algorithm stopping criterion is also satisfied.

Theorem 4 *Given two sets of centroids* $C = \{\boldsymbol{c}_k\}_{k=1}^K$ *and* $C' = \{\boldsymbol{c}'_k\}_{k=1}^K$ *, if* $\|C - C'\|_\infty = \max_{k=1,\dots,K} \|\boldsymbol{c}_k - \boldsymbol{c}'_k\| \le \varepsilon_w$*, where* $\epsilon_w = \sqrt{l^2 + \frac{\epsilon^2}{n^2}} - l$*, then* $|E^D(C) - E^D(C')| \le \varepsilon$*.*

Proof Initially, we bound the following terms: $\|\mathbf{x} - \mathbf{c}_\mathbf{x}\| + \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\|$ and $|\|\mathbf{x} - \mathbf{c}_\mathbf{x}\| - \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\||$ for any $\mathbf{x} \in D$.

If we set j and t as the indexes satisfying $\mathbf{c}_j = \mathbf{c}_\mathbf{x}$ and $\mathbf{c}'_t = \mathbf{c}'_\mathbf{x}$, then we have $\|\mathbf{x} - \mathbf{c}_\mathbf{x}\| + \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\| = \|\mathbf{x} - \mathbf{c}_j\| + \|\mathbf{x} - \mathbf{c}'_t\| \le \|\mathbf{x} - \mathbf{c}_t\| + \|\mathbf{x} - \mathbf{c}'_t\| \le 2 \cdot \|\mathbf{x} - \mathbf{c}'_t\| + \varepsilon_w = 2 \cdot \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\| + \varepsilon_w$ (1). Analogously, applying the triangular inequality, we have $|\|\mathbf{x} - \mathbf{c}_\mathbf{x}\| - \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\|| \le \varepsilon_w$ (2). In the following chain of inequalities, we will make use of (1) and (2):

$$
\begin{aligned}
|E^D(C) - E^D(C')| &\le |\sum_{\mathbf{x}\in D} \|\mathbf{x} - \mathbf{c}_\mathbf{x}\|^2 - \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\|^2| \\
&\le \sum_{\mathbf{x}\in D} |\|\mathbf{x} - \mathbf{c}_\mathbf{x}\|^2 - \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\|^2| \\
&\le \sum_{\mathbf{x}\in D} (\|\mathbf{x} - \mathbf{c}_\mathbf{x}\| + \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\|) \cdot \\
&\quad |\|\mathbf{x} - \mathbf{c}_\mathbf{x}\| - \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\|| \\
&\le \sum_{\mathbf{x}\in D} \varepsilon_w \cdot (2 \cdot \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\| + \varepsilon_w) \\
&\le n \cdot \varepsilon_w^2 + 2 \cdot n \cdot \max_{\mathbf{x}\in D} \|\mathbf{x} - \mathbf{c}'_\mathbf{x}\| \cdot \varepsilon_w \\
&\le n \cdot \varepsilon_w^2 + 2 \cdot n \cdot l \cdot \varepsilon_w = \varepsilon
\end{aligned}
$$

As can be seen in Section 2.2, there are different parameters that must be tuned. In the following result, we set a criterion to choose the initialization parameters of Algorithm 2 in a way that its complexity, even in the worst case scenario, is still the same as that of Lloyd's algorithm.

Theorem 5 *Given an integer* r*, if* $m = \mathcal{O}(\sqrt{K \cdot d})$ *and* $s = \mathcal{O}(\sqrt{n})$*, then Algorithm 2 is* $\mathcal{O}(n \cdot K \cdot d)$*.*

Proof It is enough to verify the conditions presented before. Firstly, observe that $r \cdot s \cdot m^2 = \mathcal{O}(\sqrt{n} \cdot K \cdot d)$ and $n \cdot m = \mathcal{O}(n \cdot \sqrt{K \cdot d})$. Moreover, as $K \cdot d = \mathcal{O}(n)$, then $r \cdot m^2 = \mathcal{O}(n)$.

Finally, we present a complimentary property of the grid based RPKM proposed in Capó et al. (2017). Each iteration of the RPKM can be proved to be a coreset with an exponential decrease in the error with respect to the number of iterations. This result could actually bound the BWKM error, if we fix i as the minimum number of cuts that a block, of a certain partition generated by BWKM, $\mathcal{P}$, has.

Theorem 6 *Given a set of points D in $\mathbb{R}^d$, the i-th iteration of the grid based RPKM produces a (K, ε)-coreset with $\varepsilon = \frac{1}{2^{i-1}} \cdot (1 + \frac{1}{2^{i+2}} \cdot \frac{n-1}{n}) \cdot \frac{n \cdot l^2}{OPT}$, where $OPT = \min\limits_{C \subseteq \mathbb{R}^d, |C| = K} E^D(C)$ and l the length of the diagonal of the smallest bounding box containing D.*

Proof Firstly, we denote by $\mathbf{x}'$ to the representative of $\mathbf{x} \in D$ at the i-th grid based RPKM iteration, i.e., if $\mathbf{x} \in P$ then $\mathbf{x}' = \overline{P}$, where P is a block of the corresponding dataset partition $\mathcal{P}$ of D. Observe that, at the i-th grid based RPKM iteration, the length of the diagonal of each cell is $\frac{1}{2^i} \cdot l$ and we set a positive constant, c, as the positive real number satisfying $\frac{1}{2^i} \cdot l = \sqrt{c \cdot \frac{OPT}{n}}$. By the triangular inequality, we have

$$\begin{aligned} |E^D(C) - E^{\mathcal{P}}(C)| &\leq \sum_{\mathbf{x} \in D} |\|\mathbf{x} - \mathbf{c_x}\|^2 - \|\mathbf{x}' - \mathbf{c_{x'}}\|^2| \\ &\leq \sum_{\mathbf{x} \in D} |(\|\mathbf{x} - \mathbf{c_x}\| - \|\mathbf{x}' - \mathbf{c_{x'}}\|)(\|\mathbf{x} - \mathbf{c_x}\| + \|\mathbf{x}' - \mathbf{c_{x'}}\|)| \end{aligned}$$

Analogously, observe that the following inequalities hold $\|\mathbf{x}' - \mathbf{c_{x'}}\| + \|\mathbf{x} - \mathbf{x}'\| \geq \|\mathbf{x} - \mathbf{c_x}\|$ and $\|\mathbf{x} - \mathbf{c_x}\| + \|\mathbf{x} - \mathbf{x}'\| \geq \|\mathbf{x}' - \mathbf{c_{x'}}\|$. Thus, $\|\mathbf{x} - \mathbf{x}'\| \geq |\|\mathbf{x} - \mathbf{c_x}\| - \|\mathbf{x}' - \mathbf{c_{x'}}\||$:

$$|E^D(C) - E^{\mathcal{P}}(C)| \leq \sum_{\mathbf{x} \in D} \|\mathbf{x} - \mathbf{x}'\| \cdot (2 \cdot \|\mathbf{x} - \mathbf{c_x}\| + \|\mathbf{x} - \mathbf{x}'\|)$$

On the other hand, we know that $\sum\limits_{\mathbf{x} \in D} \|\mathbf{x} - \mathbf{x}'\|^2 \leq \frac{n-1}{2^{2i+1}} \cdot l^2$ and that, as both $\mathbf{x}$ and $\mathbf{x}'$ must be located in the same cell, $\|\mathbf{x} - \mathbf{x}'\| \leq \frac{1}{2^i} \cdot l$. Therefore, as $\mathbf{d}(\mathbf{x}, C) \leq l$,

$$\begin{aligned} |E^D(C) - E^{\mathcal{P}}(C)| &\leq (\frac{n-1}{2^{2i+1}} + \frac{n}{2^{i-1}}) \cdot l^2 \\ &\leq (\frac{n-1}{2^{2i+1}} + \frac{n}{2^{i-1}}) \cdot 2^{2i} \cdot c \cdot \frac{OPT}{n} \\ &\leq (\frac{1}{2^{i+2}} \cdot \frac{n-1}{n} + 1) \cdot 2^{i+1} \cdot c \cdot E^D(C) \end{aligned}$$

In other words, the i-th RPKM iteration is a (K, ε)- coreset with $\varepsilon = (\frac{1}{2^{i+2}} \cdot \frac{n-1}{n} + 1) \cdot 2^{i+1} \cdot c = \frac{1}{2^{i-1}} \cdot (1 + \frac{1}{2^{i+2}} \cdot \frac{n-1}{n}) \cdot \frac{n \cdot l^2}{OPT}$.

C About the grid based RPKM

In the experimental section in Capó et al. (2017), the partition sequence used (grid based RPKM) consisted on sequentially constructing a new spatial partition by dividing each block of the previous partition into 2^d new blocks, i.e., $\mathcal{P}$ can have up to $2^{i \cdot d}$ representatives. In this section, we provide some additional results in which we compare the performance of the grid based RPKM with respect to the methods and datasets presented in Section 3 and $K \in \{3, 5, 10, 25, 50\}$.

As in Capó et al. (2017), we fix the maximum number of iterations, M, as the stopping criterion for the grid based RPKM. Initially, we considered $M = 10$, however just for the *CIF* and *3RN* datasets -case i)- the grid based RPKM managed to converge before reaching the limit running time (24 hours). Moreover, for the *HPC* and *WUY* datasets -case ii)-, we obtained results for $M = 5$ and, unfortunately, for the datasets with the largests dimensionality (*GS* and *SUSY*), the grid based RPKM failed to provide any output. The obtained results are summarized in the following figures:

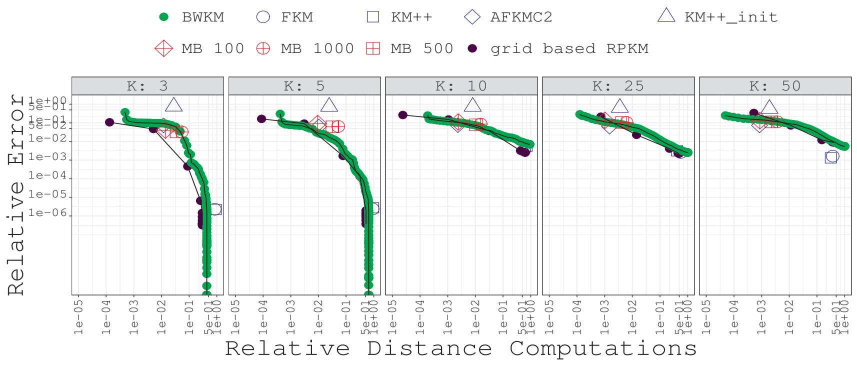

Fig. 9: Relative distance computations *vs* relative error on the *CIF* dataset.

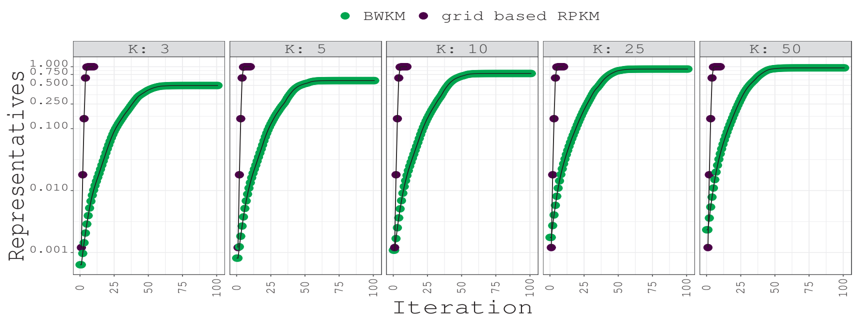

Fig. 10: Proportion representatives/instances with respect to the number of iterations on the *CIF* dataset.

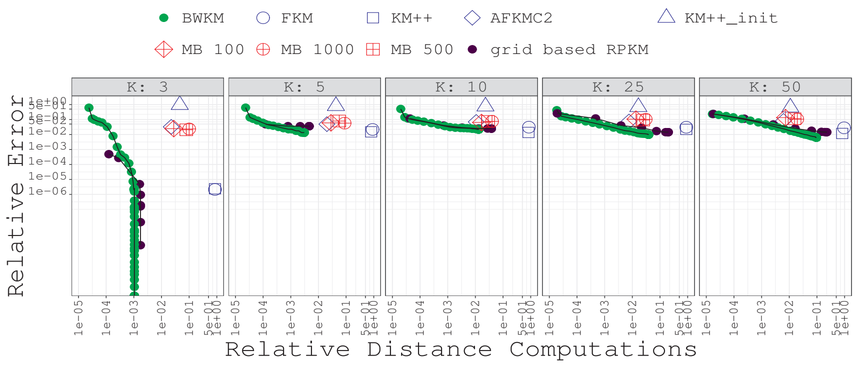

Fig. 11: Relative distance computations *vs* relative error on the *3RN* dataset.

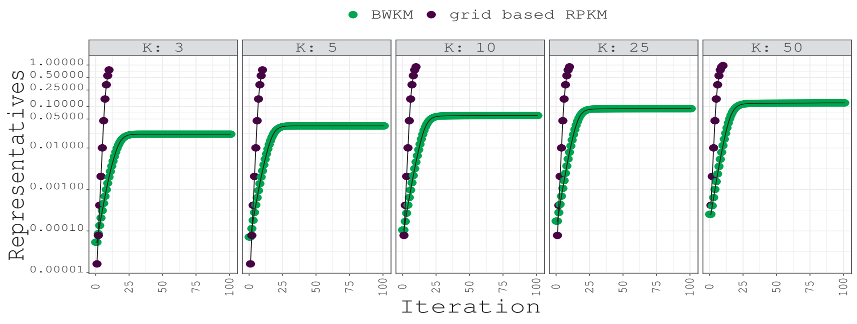

Fig. 12: Proportion representatives/instances with respect to the number of iterations on the *3RN* dataset.

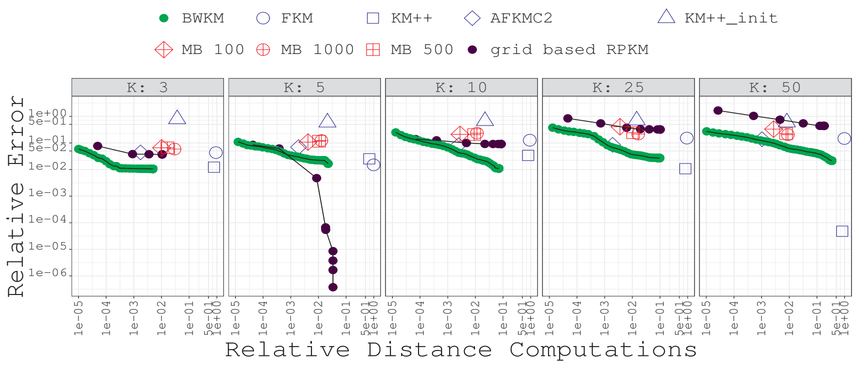

Fig. 13: Relative distance computations *vs* relative error on the *HPC* dataset.

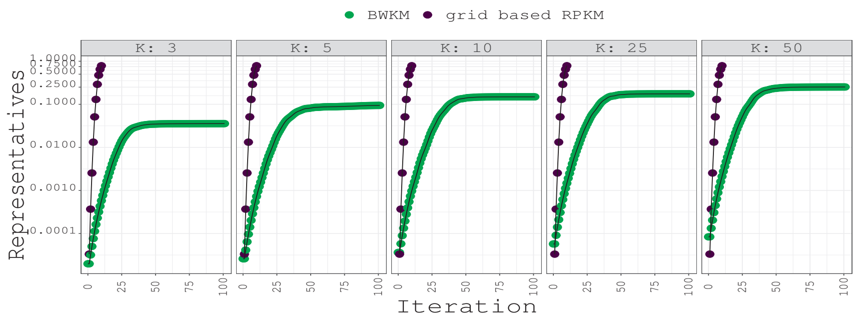

Fig. 14: Proportion representatives/instances with respect to the number of iterations on the *HPC* dataset.

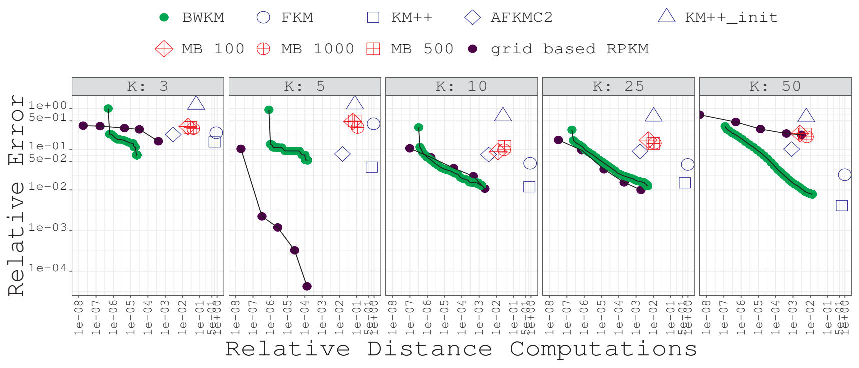

Fig. 15: Relative distance computations *vs* relative error on the *WUY* dataset.

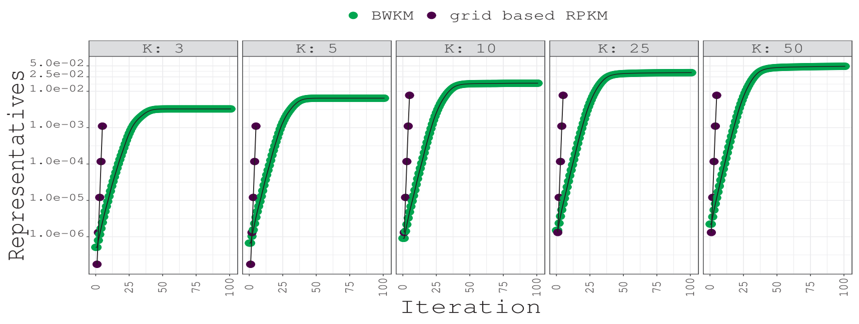

Fig. 16: Proportion representatives/instances with respect to the number of iterations on the *WUY* dataset.

Dataset	K	**BWKM**	**grid based RPKM**
CIF	3	0.498 ± 0.042	0.999 ± 0.000
	5	0.597 ± 0.021	0.999 ± 0.000
	10	0.779 ± 0.013	0.999 ± 0.000
	25	0.917 ± 0.005	0.999 ± 0.000
	50	0.958 ± 0.002	0.999 ± 0.000
3RN	3	0.021 ± 0.000	0.760 ± 0.000
	5	0.034 ± 0.001	0.760 ± 0.000
	10	0.060 ± 0.004	0.900 ± 0.000
	25	0.090 ± 0.008	0.900 ± 0.000
	50	0.123 ± 0.009	0.967 ± 0.000
HPC	3	0.038 ± 0.014	0.785 ± 0.000
	5	0.095 ± 0.014	0.785 ± 0.000
	10	0.148 ± 0.008	0.785 ± 0.000
	25	0.175 ± 0.009	0.785 ± 0.000
	50	0.253 ± 0.011	0.785 ± 0.000
WUY	3	0.003 ± 0.001	0.001 ± 0.000
	5	0.006 ± 0.000	0.001 ± 0.000
	10	0.017 ± 0.004	0.007 ± 0.000
	25	0.033 ± 0.004	0.007 ± 0.000
	50	0.049 ± 0.003	0.007 ± 0.000

Table 3: Proportion final number of representatives / instances for the different datasets and number of clusters.

In the datasets of case i), we have better view of the evolution of the number of representatives of the grid based RPKM with respect to the number of iterations. In Fig.10, Fig.12 and Table 4, we observe that the number of representatives of the grid based RPKM, after 10 iterations, is about the number of instances in both the *CIF* and *3RN* datasets, while, for the BWKM, we observe a much slower growth in the number of representatives. In particular, for the *3RN* dataset, the number of representatives, for the different number of clusters and after 100 iterations, is still under 13% the number of instances, while generating approximations of similar and/or better quality than those of the grid based RPKM. Furthermore, we observe that, for all the datasets, the number of representatives of BWKM reaches a *plateau* way before the final number of iterations, meaning that, for a small number of iterations, most of the blocks generated by BWKM are well assigned. On the other hand, as the number of representatives for the grid based RPKM, for the datasets in case ii), is smaller than in the previous case, we observe in Fig.13 and Fig.15, that the quality of the approximation of the grid based RPKM is commonly much less competitive than the solutions obtained via BWKM: the grid based RPKM commonly has over 10% of relative error with respect to BWKM.

In Table 4, we present the proportion of cases in which BWKM generates a well assigned partition verified via the assignment function of Theorem 1. As we commented in Section A.2, this is a sufficient condition to verify that the solution obtained via BWKM is also a fixed point of Lloyd's algorithm for the entire dataset. We observe that, specially for a low number of clusters, BWKM is very likely to converge to a local minima of the K-means problem.

For instance, for *WUY* dataset and $K \in \{3, 5\}$, BWKM always generated well assigned partitions, which is quite remarkable as the number of representatives in these cases is under 0.6% of the number of instances. As expected, as the number of cluster increases, it is harder to verify such a condition, however we must remember that this is just a sufficient condition since we are using Theorem 1, rather than computing all the pairwise distances instance-centroid.

K	***CIF***	***3RN***	***HPC***	***WUY***
3	0.92	1.00	0.78	1.00
5	1.00	1.00	0.70	1.00
10	0.44	0.98	0.46	0.84
25	0.40	0.96	0.54	0.08
50	0.48	0.84	0.02	0.00

Table 4: Proportion of cases in which the spatial partition obtained by BWKM satisifes Theorem 3 verified via the misassigment function of Theorem 1.

From the results presented in this section, it is clear that BWKM alleviates the main drawback of the grid based RPKM, as it also controls the growth of the number of representatives, which, in the worst case scenario, only has a linearly growth in this case. This is an important factor, as it allows BWKM to scale better with respect to both the dimensionality and the number of iterations. Furthermore, BWKM is still a RPKM type approach, meaning that, besides the theoretical guarantees that we have developed during article and the results that just commented on, BWKM also has the guarantees of the grid based RPKM.

References

Aloise D, Deshpande A, Hansen P, Popat P (2009) NP-hardness of Euclidean sum-of-squares clustering. Machine Learning 75(2):245–248

Arthur D, Vassilvitskii S (2007) k-means++: The Advantages of Careful Seeding. In: Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms, pp 1027–1035

Äyrämö S, Kärkkäinen T (2006) Introduction to Partitioning-Based Clustering Methods with a Robust Example. Reports of the Department of Mathematical Information Technology Series C, Software engineering and computational intelligence 1/2006

Bachem O, Lucic M, Hassani H, Krause A (2016) Fast and Provably Good Seedings for K-means. In: Advances in Neural Information Processing Systems, pp 55–63

Bachem O, Lucic M, Krause A (2018) Scalable K-means Clustering via Lightweight Coresets. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 1119–1127

Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S (2012) Scalable K-means++. Proceedings of the VLDB Endowment 5(7):622–633

Balcan MFF, Ehrlich S, Liang Y (2013) Distributed K-means and K-median Clustering on General Topologies. In: Advances in Neural Information Processing Systems, pp 1995–2003

Berkhin P, et al. (2006) A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data 25:71

Bottou L, Bengio Y (1995) Convergence Properties of the K-means Algorithms. In: Advances in Neural Information Processing Systems, pp 585–592

Boutsidis C, Drineas P, Mahoney MW (2009) Unsupervised Feature Selection for the K-means Clustering Problem. In: Advances in Neural Information Processing Systems, pp 153–161

Boutsidis C, Zouzias A, Drineas P (2010) Random Projections for K-means Clustering. In: Advances in Neural Information Processing Systems, pp 298–306

Bradley PS, Fayyad UM (1998) Refining Initial Points for K-means Clustering. In: Proceedings of the 15th International Conference on Machine Learning, vol 98, pp 91–99

Capó M, Pérez A, Lozano JA (2017) An Efficient Approximation to the K-means Clustering for Massive Data. Knowledge-Based Systems 117:56–69

Cohen MB, Elder S, Musco C, Musco C, Persu M (2015) Dimensionality Reduction for K-means Clustering and Low Rank Approximation. In: Proceedings of the forty-seventh annual ACM symposium on Theory of computing, ACM, pp 163–172

Davidson I, Satyanarayana A (2003) Speeding up K-means Clustering by Bootstrap Averaging. In: IEEE Data Mining Workshop on Clustering Large Data Sets

Ding C, He X (2004) K-means Clustering via Principal Component Analysis. In: Proceedings of the twenty-first international conference on Machine learning, ACM, p 29

Ding Y, Zhao Y, Shen X, Musuvathi M, Mytkowicz T (2015) Yinyang k-means: A Drop-In Replacement of the Classic K-means with Consistent Speedup. In: International Conference on Machine Learning, pp 579–587

Drake J, Hamerly G (2012) Accelerated K-means with Adaptive Distance Bounds. In: 5th NIPS Workshop on Optimization for Machine Learning, pp 42–53

Elkan C (2003) Using the Triangle Inequality to Accelerate K-means. In: Proceedings of the 20th International Conference on Machine Learning, pp 147–153

Feldman D, Monemizadeh M, Sohler C (2007) A PTAS for K-means Clustering Based on Weak Coresets. In: Proceedings of the twenty-third annual symposium on Computational geometry, pp 11–18

Forgy EW (1965) Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications. Biometrics 21:768–769

Hamerly G (2010) Making K-means Even Faster. In: Proceedings of the 2010 SIAM International Conference on Data Mining, pp 130–140

Har-Peled S, Mazumdar S (2004) On Coresets for K-means and K-median Clustering. In: Proceedings of the 36th ACM Symposium on Theory of Computing, pp 291–300

Jain AK (2010) Data Clustering: 50 Years Beyond K-means. Pattern Recognition Letters 31(8):651–666

Jain AK, Dubes RC (1988) Algorithms for Clustering Data. Prentice-Hall, Inc.

Jain AK, Murty MN, Flynn PJ (1999) Data Clustering: A Review. ACM computing surveys 31(3):264–323

Jordan M (2013) Committee on the analysis of massive data, committee on applied and theoretical statistics, board on mathematical sciences and their applications, division on engineering and physical sciences, council, nr. frontiers in massive data analysis. Frontiers in Massive Data Analysis

Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY (2002a) An Efficient K-means Clustering Algorithm: Analysis and Implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):881–892

Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY (2002b) A Local Search Approximation Algorithm for K-means Clustering. In: Proceedings of the 18th annual Symposium on Computational Geometry, pp 10–18

Kaufman L, Rousseeuw P (1987) Clustering by Means of Medoids. North-Holland

Kumar A, Sabharwal Y, Sen S (2004) A Simple Linear Time $(1 + \varepsilon)$-Approximation Algorithm for K-means Clustering in Any Dimensions. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp 454–462

Lloyd S (1982) Least Squares Quantization in PCM. IEEE Transactions on Information Theory 28(2):129–137

Lucic M, Bachem O, Krause A (2016) Strong Coresets for Hard and Soft Bregman Clustering with Applications to Exponential Family Mixtures. In: Artificial Intelligence and Statistics, pp 1–9

Mahajan M, Nimbhorkar P, Varadarajan K (2009) The Planar k-means Problem is NP-hard. In: International Workshop on Algorithms and Computation, pp 274–285

Manning CD, Raghavan P, Schütze H (2008) Evaluation in Information Retrieval. Introduction to Information Retrieval pp 151–175

Matoušek J (2000) On Approximate Geometric K-Clustering. Discrete & Computational Geometry 24(1):61–84

Newling J, Fleuret F (2016) Nested Mini-Batch K-means. In: Advances in Neural Information Processing Systems, pp 1352–1360

Peña JM, Lozano JA, Larrañaga P (1999) An Empirical Comparison of Four Initialization Methods for the K-means algorithm. Pattern Recognition Letters 20(10):1027–1040

Redmond SJ, Heneghan C (2007) A Method for Initialising the K-means Clustering Algorithm Using KD-trees. Pattern Recognition Letters 28(8):965–973

Sculley D (2010) Web-Scale K-means Clustering. In: Proceedings of the 19th International conference on World Wide Web, pp 1177–1178

Shen X, Liu W, Tsang I, Shen F, Sun QS (2017) Compressed K-means for Large-Scale Clustering. In: Thirty-First AAAI Conference on Artificial Intelligence

Steinley D, Brusco MJ (2007) Initializing K-means Batch Clustering: A Critical Evaluation of Several Techniques. Journal of Classification 24(1):99–121

Vattani A (2011) K-means Requires Exponentially Many Iterations Even in the Plane. Discrete & Computational Geometry 45(4):596–616

Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Philip SY, et al. (2008) Top 10 Algorithms in Data Mining. Knowledge and Information Systems 14(1):1–37

Zhao W, Ma H, He Q (2009) Parallel K-means Clustering Based on MapReduce. In: IEEE International Conference on Cloud Computing, pp 674–679